


Article

Retaliation against Ransomware in Cloud-Enabled PureOS System

Atef Ibrahim ^{1,2,*}, Usman Tariq ³, Tariq Ahamed Ahanger ³, Bilal Tariq ⁴ and Fayez Gebali ²

¹ Computer Engineering Department, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 16278, Saudi Arabia

² Electrical and Computer Engineering Department, University of Victoria, Victoria, BC V8P 5C2, Canada

³ Management Information System Department, College of Business Administration, Prince Sattam Bin Abdulaziz University, Al-Kharj 16278, Saudi Arabia

⁴ Department of Management Sciences, COMSATS University Islamabad, Vehari Campus, Vehari 61010, Pakistan

* Correspondence: aa.mohamed@psau.edu.sa; Tel.: +966-546825905

Abstract: Ransomware is malicious software that encrypts data before demanding payment to unlock them. The majority of ransomware variants use nearly identical command and control (C&C) servers but with minor upgrades. There are numerous variations of ransomware, each of which can encrypt either the entire computer system or specific files. Malicious software needs to infiltrate a system before it can do any real damage. Manually inspecting all potentially malicious file types is a time-consuming and resource-intensive requirement of conventional security software. Using established metrics, this research delves into the complex issues of identifying and preventing ransomware. On the basis of real-world malware samples, we created a parameterized categorization strategy for functional classes and suggestive features. We also furnished a set of criteria that highlights the most commonly featured criteria and investigated both behavior and insights. We used a distinct operating system and specific cloud platform to facilitate remote access and collaboration on files throughout the entire operational experimental infrastructure. With the help of our proposed ransomware detection mechanism, we were able to effectively recognize and prevent both state-of-art and modified ransomware anomalies. Aggregated log revealed a consistent but satisfactory detection rate at 89%. To the best of our knowledge, no research exists that has investigated the ransomware detection and impact of ransomware for PureOS, which offers a unique platform for PC, mobile phones, and resource intensive IoT (Internet of Things) devices.

Keywords: ransomware detection; malicious software; file monitoring

MSC: 68M25



Citation: Ibrahim, A.; Tariq, U.; Ahamed Ahanger, T.; Tariq, B.; Gebali, F. Retaliation against Ransomware in Cloud-Enabled PureOS System. *Mathematics* **2023**, *11*, 249. <https://doi.org/10.3390/math11010249>

Academic Editor: Marjan Mernik

Received: 25 November 2022

Revised: 19 December 2022

Accepted: 24 December 2022

Published: 3 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Anticipate that the inoperability of your most important systems brings your enterprise to a grinding halt. As a result, the perpetrators seek a ransom to restore access to your computer systems. Spam, phishing, ransomware, and denial-of-service (DoS) cyberattacks are only a few examples of the malevolent behavior that plagues the Internet and affiliate systems. A substantial portion of the risk/threat depends on armies of infected computers, or botnets, which are dispersed all over the Internet. Hostile operations may not be uniformly dispersed across the network. Some connections may not have enough security measures, which could lead to numerous infected workstations. Other systems, on the other hand, may have strict security measures and see no malicious activity at all. In addition, there is the possibility that some networks have been created for the explicit purpose of engaging in malevolent behavior.

The following are some of the most typical entry points for attacks that we have observed:

- (a) Servers supporting the Remote Desktop Protocol (RDP) or the Virtual Private Network (VPN) were breached, giving adversaries access to users' data and allowing them to eavesdrop on their communications. Intruders used brute-force logon to gain access to systems without two-factor authentication. In other cases, hackers used weak privileges to get into the system through VPNs.
- (b) An intruder used a flaw in a server or program that had not been fixed.
- (c) To acquire access to the target system, an attacker used spear phishing to contact specific individuals.

We have found that as soon as a malicious entity entered into the victim's system by using an RDP server, it started a self-directed search for important systems in the user's infrastructure. This allowed the attacker to discover the assets that needed to be managed. Once a hacker knew how the network worked, he or she could use that knowledge to install backdoors on the most important systems from afar. After the backdoors were installed, the attacker could go around to the company's internal backup systems, where they could delete all copies of the enterprise's data to stop them from being recovered. As a decisive step, intruders are eligible to deploy the ransomware to cripple the entire infrastructure and make demands from the needful.

By creating an atmosphere of panic, the creators of ransomware want to coerce their targets into clicking on a link or paying a ransom, both of which may result in further malware infection of the user's system. There are many different kinds of ransomware, and each one has its own set of characteristics. Table 1 illustrates commonly used ransomware variants that were differentiated based on criteria such as computing methodology, learning paradigm, and anomaly analysis.

Since the famous WannaCry attack in 2017 brought ransomware to the forefront of the news, many attacks on organizations and systems have caught the attention of people all over the world. Malwarebytes [6] research shows that the United States is still a popular place for ransomware attacks. In August 2022, 43 countries, including Luxembourg, Qatar, and Gabon, reported 175 attacks. In 2021, when hackers broke into the Colonial Pipeline systems, an oil pipeline in the United States had to be shut down and a \$4.4 million ransom payment was lost. Then, hackers in the UK attacked a supplier of information technology for the National Health Service. This made it impossible to access patient records and put an important health service at risk.

In September 2022, we surveyed the top business leaders in the Gulf region to find out how much they knew about ransomware and how prepared they were. Our key findings are as follows:

- Ransomware attacks are causing an increasing amount of stress for sixty-six percent of all cybersecurity experts.
- Ransomware attacks have caused substantial disruption to twenty percent of all enterprises.
- Ninety-two percent of cybersecurity experts agree that additional expense is necessary to tackle ransomware, and these leaders estimate that an extra budget rise of twenty-two percent is essential on average.
- A ransomware cyberattack has resulted in a reduction of profit for twenty-four percent of firms in the preceding twelve months.
- Overall, forty-four percent of intrusions cost well over USD 100,000.

For companies and their employees, the effects of a cyber incident can be catastrophic. Even though many years have passed, the volume and severity of cyberattacks have not decreased, and ransomware remains a major participant in the difficulties encountered by cybersecurity leaders and their employees. Considering the potentially disastrous effects of ransomware risk, this research made the following contributions:

- (1) With the help of standard metrics, we characterized the broad challenges of combating ransomware.

- (2) With real-world malware samples as a baseline, we developed a classification system of parameters for function classes and indication attributes.
- (3) In addition to this, we detailed a set of behavior-based and metadata-based criteria, both of which include previously recommended elements.
- (4) We used a Tonido cloud platform [7] for the functionality, such as enterprise file sharing, sync, backup, and remote access. Cloud storage was enhanced by a program for detecting ransomware, which added a crucial usability feature in the form of simple data recovery via the built-in file source control mechanisms.
- (5) We demonstrated that ransomware can be reliably detected, and that the functional user can quickly adapt to the transition interface.

Table 1. Ransomware overview.

Ransomware	Description	Attack Vector	Operating OS and Software
Ryuk [1]	Ryuk is one type of ransomware that only attacks selected devices. Common forms of infection include spear phishing and unauthorized RDP connections to corporate networks. After Ryuk has successfully infected a system, it will encrypt predefined types of data (but not those that are necessary for the computer to work) and then ask for money.	Network Analysis	Windows OS
Maze [2]	The Maze ransomware is groundbreaking because it is the first type of ransomware specifically designed to steal data in addition to encrypting files. The Maze encrypts victims' sensitive data after they refuse to pay the ransom.	Files Monitoring	Word or Excel files
LockBit [3]	LockBit, a relatively new Ransomware-as-a-Service, has been active as of September 2019. This ransomware was programmed to encrypt data repositories of large organizations quickly so that security devices would not find it right away.	Files Monitoring, System Honeypot	Windows Powershell and Server Message Block (SMB)
DearCry [4]	When it comes to files, the DearCry ransomware is selective. After the process of encrypting the data is done, DearCry shows a message asking for money before giving instructions on how to decrypt the data.	Keys Backup	Microsoft Exchange servers
WannaCry [5]	The WannaCry ransomware infection spread quickly across the internet and into many different computer networks. Once it gets into a Windows computer, it encrypts the data on the hard drive, making it hard for the user to access the data. Then, it will ask for a ransom payment in cryptocurrency to unlock the contents.	Files Monitoring, System Honeypot	Windows OS

Challenges in Countering Ransomware

The infrequency of ransomware attacks combined with the effectiveness of automated recovery creates a dilemma for detection rates: If we assume that a ransomware predictive algorithm has a false-positive rate of 0.5%, then it would need to track every single one of the 500,000 file actions that occur daily in the file system. A total of 250 false restorations would be initiated, and 250 pointless user engagements would be required to undo them.

Moreover, the work done to cut down on false positives has led to a clear set of signs that ransomware developers can use to make malware that cannot be found.

Since people have become more aware of malware, and anti-virus software has become better at recognizing it through pattern recognition, cybercriminals have decided to make it harder to find malicious applications when they attempt to access target computers. To evade detection by anti-malware tools, hackers began utilizing a variety of masking techniques. For example, a polymorphic virus is one that can change its behavior in each new variant without altering its configuration. Because of this, simple pattern matching becomes less effective in identifying it.

Even though most enterprises have been using firewalls, intrusion detection systems (IDS), and a wide range of other system-centric preventative and detective defensive measures for a long time, it is still important to use the right defensive layers for a complete defense. When security measures such as a firewall, IDS, web content filter, and others are bypassed, an endpoint can be attacked. Thus, false positives and automatic real-time restoration go together because there are fewer valid indicators, and each one only works for the victim system. Therefore, in order to make use of the behavior-based indicators, it is essential to keep track of the functional actions of the systems for a considerable amount of time.

A similar dilemma exists for the files that are stored on victim devices. Encrypted files with unfamiliar file extensions, such as a cipher file container with an unknown file format, can be the result of perfectly safe file activities. The information and log file detectors might notice these general traits and place the data in the wrong category.

We organize the paper in the following sections: The literature review is presented in Section 2. The research gap is discussed in Section 3. In Section 4, a recommended methodology is presented, whereas Section 5 presents the performance analysis of our proposed ransomware analysis and identification architecture. We conclude in Section 6.

2. Literature Review

Here, we will review what has already been accomplished in the study of malicious programs, especially ransomware. There are two main foci here: malware analysis and ransomware detection. We begin with analysis since before learning how to identify cyberattacks, it is necessary to grasp what they entail and how they function. Even though some of the literature is about malware in general, it is thought that this is relevant to our study because ransomware is a type of malicious activity. We will begin by reviewing research that aims to analyze ransomware. Before considering ransomware, we will take a look at the research that investigated the runtime environment of these cyberattacks. Afterwards, we will share published papers that investigate how malware operates.

Agrawal [8] made prototypes of several types of ransomware so that we could learn about their source code, features, and the way their cryptosystem data structures are set up. The author concluded that the code quality is usually “very rudimentary”, that both symmetric and asymmetric encryption methods are employed, and that ransomware instances are not tailored to any one organization but rather are made for general use. Even though they were able to get useful information from the malware metrics, they could not reverse-engineer each version. Kara et al. [9] looked at numerous ransomware segments from many different families. The study’s authors concluded that intrusions belonging to the same malware family might use different execution techniques but still share several commonalities. There are also differences in the complexity of encoding, locking, and deleting data that can be seen across ransomware datasets. The results showed that useful information can be gathered even without a deep understanding of how the new datasets work. However, due to the growing number of ransomware variants, this process still requires manual labor and could be automated in the future.

Sharma et al. [10] implemented RansomDroid, the novel system built on an unconstrained machine learning method, to make it easy to find Android ransomware. This architecture does not require conventional anti-virus providers to pre-label attribute values

as malicious or benign. RansomDroid does a comprehensive study of the services used by Android ransomware, allowing it to identify and remove malicious privileges, ambitions, strings in native languages derived from imagery and text, locking, encrypting, and encoding techniques. Even though the study claimed that it improved the accuracy of malware detection, no dynamic analysis was conducted in environments like CuckooDroid Sandbox to find dynamic features.

Machine learning is effectively detecting and classifying cybersecurity threats. Using elaborate learning techniques and many attributes, detectors can be simulated. Classified parameters provide high performance but understanding the knowledge taught is difficult. To overcome this challenge, machine learning algorithms for information security applications are being studied for accuracy. In particular, relying on interpretations could improve the ability to find and stop malicious activity by helping human experts determine the unique traits that really define malware activity. Scalas et al. [11] discovered a novel way to use state-of-the-art reasoning techniques to find ransomware’s key exclusionary traits. Their insights help shed some light on the origins and evolution of ransomware families. The authors described ransomware families and how they have changed over time. They also suggested ways to use explanation techniques effectively. Since the research only looked at well-known malware signatures to find ransomware, it may not be possible to show how well it works, especially in cases where the behavior of the malware is unknown, or the executable is stealth.

Furthermore, keeping data safe on Internet of Things devices without resorting to root access is a complex task at present. This highlights the critical necessity of prevention and mitigation solutions on ROOT-less IoT devices. Wang et al. [12] analyzed cryptographic ransomware and outlined its distinguishing features. They advocate for a decoy-based technique to safeguard files from ransomware. KRProtector was created and developed to identify ransomware and safeguard files using decoys, satisfying the demand for file security on computers that lacked root access.

In Table 2, the performance matrix was generated based on accuracy ratio, errors, false-positive ratio, true-positive ratio, false-negative ratio, true-negative ratio, precision, recall, and F-measure. In accordance with the criteria, the analytical sensitivity of a test that is otherwise similar may end up being noticeably different even when the sample matrix that it is being tested on is unchanged. As shown by our findings, the F1 score considers both precision and recall about the positive class, while the accuracy score takes into account correctly recognized observations regardless of whether or not they belong to the positive or negative class.

Table 2. Research Survey Synopsis on Literature Evaluation.

Ref#	Methods of Analysis			Analysis Methods	The Analysis’s Distinctive Features	Performance Indicators	Analytical Sensitivity
	Static	Dynamic	Hybrid				
[13]			✓	deep learning model	binary file analysis	heuristic approach	99.93% accuracy
[14]			✓	evolutionary-based machine learning approach	evolutionary-based machine learning approach	sensitivity, specificity, No. of features	96.4% accuracy
[15]	✓	✓	✓	bidirectional long short-term memory model	behavior and process memory analysis	EfficientNet-B3 architecture	94.7% accuracy
[16]	✓	✓		pre-trained convolutional neural network (CNN) model	binary classification of Android malware images	EfficientNet-B4 architecture	95.7% accuracy
[17]		✓		one-dimensional convolutional neural networks (CNN) model	Windows Portable Executable (PE) malware dataset	ExtraTrees classifier using XGBoost	98.62% accuracy

3. Research Gap

No studies have addressed this devastating risk (i.e., ransomware) for PureOS [17] as far as we are aware. Even though innovative academic studies such as [10] were able to find and identify non-ransomware malware in the auditable PureOS environment (i.e., of Librem 5 (mobile phones) and Purism’s laptops) with almost perfect accuracy, they were only able to find and identify 50% of our ransomware dataset. The rationale for this is that ransomware techniques are fundamentally imitation attacks, which means that the entire callousness of the operation is only observable as a set of behaviors that are considered to be genuine. While cryptography and screen blocking on their own are not malicious, when used together they certainly are. Because of this, it should come as no surprise that general methods for detecting malware have a low recall rate.

3.1. Applied Ransomware Dataset Overview

A total of 669 ransomware samples from among the most prominent families and variations is included in the collection. The collection contains not just ransomware samples but also information from 197 normal features that are representative of the most widely used programs on Windows. Similar benign and anomalous vulnerabilities were tested for PureOS. The whole dataset occupies 457 GB of storage space.

3.2. Dataset Structure

In the primary directory that is created when a dataset zip file is extracted, we encountered subdirectories labeled with the family names of the samples that were studied, such as “HiddenWasp”, “Kryptik”, etc. The “Benign” subfolder contains information on the programs’ behavior. Each sample was assigned its own subfolder inside the family directory, which held all of the sample’s data. When analyzing a sample, a unique identifier was generated and allocated to each subfolder. For each ransomware and benign sample evaluated, various behavioral data were obtained. These details were organized and kept in a variety of files and folders. Each sample was tested in our research facility, where we could confirm that it encrypts user data without the interference of the Internet or any active C&C servers. The information is further summarized in Table 3.

Table 3. Extraction of features from the PureOS ransomware (selected sample).

Files/Folders	Characteristics	Investigated Variables	Applied Methodologies	Sample Features
PureOS*.xml	Permissions Intent/Behavior Text	{Access-authorization} {response}	Access Rights Actions Content_View	276
*.src	Lock Hash Cipher	{suspiciously slower} {Intensive CPU usage percentage}	Methods()	449
Malvertising	Lock Hash Cipher	{Processor & Graphics Card Overheating}	malware_code()	583
Agent Tesla (RAT), AZORult, MOUSEISLAND embedded macros, NanoCore (RAT), Qakbot (embedded), GootKit (loader), Remcos (backdoor), Address Resolution Protocol (ARP), ASPXSpy, HAFNIUM, Active Scanning, Aggregate Victim Information (APT32, and Magic Hound), Boot Integrity, Command & Scripting Interpreter (CHOPSTICK, DarkComet, FIN5-7, FIVEHANDS, Get2, Matryoshka, and Zeus Panda), etc.				Sample Features’ Example

4. Proposed Methodology

Ransomware is malicious software that encrypts user files on infected computers. A ransomware attack can start with a hacked website, a download of an infected file, the use of a security hole, a coordinated operation, or malware. Once ransomware is installed on a computer, it will often spread to other computers on the network and communicate with a C&C (command and control) server that the person who installed it controls. As soon as

the ransomware receives a command (such as “encrypt files”) from the adversary, it begins its encryption process.

Based on our experimental dataset (i.e., pre-labeled attributes), most ransomware-infected files were encrypted with asymmetric encryption, which is a secure form of cryptography that uses two keys (a secret key and a session key) to cipher and decipher data. The adversary has possession of a secret key and will provide the targeted system with a key pair. The ransomware immediately starts ciphering data using the information contained in the cryptographic key. In most cases, ransomware will cipher data that are not mission-critical dependent on the file type (for example, .docx, .xlsx, png, or jpeg) in order to ensure that the defendant’s machine continues to operate continuously for the victim to be able to pay the ransom. Once the malicious cryptographic functions have begun, it can rapidly propagate across the network and into all shared files.

There are many varieties of crypto-ransomware. It is crucial to consider both the type of encryption used (symmetric or asymmetric encryption) and how it is implemented. Ransomware’s payload, symbolized by its encryption method, is the source of its malevolent intention. The only way for an adversary and a target to successfully transact (key-money) is if the defendant’s data cannot be recovered via any other means. As a result, the likelihood of successfully extracting ciphered files decreases as their level of security increases.

Based on our thorough analysis of a number of different types of PureOS-focused ransomware from all the available categories, our main finding was that there are specific unique parts of ransomware methods that are different from other malware families and, of course, from legitimate software. To be more precise, our strategy was to evaluate whether or not a mobile application seeks to intimidate the operator, restrict the device, or encode data—or whether or not it intends to do all of these things simultaneously. Therefore, we classified several goals of PureOS security analysis techniques:

- (a) By manipulating access controls, malicious programs launched a wide variety of attacks, including denial-of-service (DoS) and distributed denial-of-service (DDoS) assaults, as well as attacks that destroy data or compromise data integrity. Thus, the system’s resources cannot be used unless authorized users have legitimate access to them.
- (b) PureOS anomaly detector was particularly focused about data security. Phone numbers, Wi-Fi hotspot credentials, GPS coordinates, etc., all pose security risks.
- (c) For any given software to function properly, code verification was necessary.
- (d) Exploitation of cryptography led to access control flaws in the system, such as when SSL (Secure Sockets Layer)/TLS (Transport Layer Security) validation failed and MITM (Man-in-the-middle) attacks occurred.
- (e) Among PureOS’s many security flaws, intent infiltration and resource hijacking ranked among the most prevalent. By tampering with user data, unauthorized code was executed, resulting in intent infiltration. Unauthorized access to restricted data was labeled as resource hijacking, and it happened when malicious actors exploited security flaws in applications by exploiting exported features.
- (f) Explored a correlation between malware infection and the amount of power consumed by devices with location awareness.
- (g) Investigated a phenomenon about how features such as ‘Control Flow Graph (CFG), Inter-procedural Control Flow Graph (ICFG)’ have an impact on malware identification and detection.

Thus, developing a reliable method to identify ransomware entails creating classifiers for the many malware families. Existing ransomware families’ behaviors, as well as potential but as yet unrealized implementations, were used to identify indicators. Analyzed malware could:

- (a) rewrite the contents of a file in place. This was achieved by reading the file’s contents, encrypting them, rewriting it with the encrypted versions, and then closing the file. The file could be given a new name afterward, if desired.

- (b) generate a new file and copy its contents, then delete the source thereafter. This was achieved by obtaining the original file's contents, encrypting them, and then putting the encrypted data into the new file.
- (c) In order to dispose of files in a way that was unique, the actions were conducted in a variety of ways that varied from each preceding transaction. A few actions to exemplify this are:
 - i. Overwriting and erasing data in batches (write) and deleting or renaming (batch).
 - ii. "Write" followed by "delete" or "rename", and vice versa.
- (d) When ransomware encrypted a file, it would either give it a completely arbitrary file extension (such as .r5a), a ransomware-specific file extension (such as .aesir, .file0locked, .AngleWare, etc.), or a similar file extension to the file that was encrypted (e.g., .xls). Due to the encrypted data, the file system could not tell what format the file was based on the file identification header. As a result, the file became corrupted and showed a version that did not exist (i.e., xls) in the last element. We were able to determine what kind of data were in a file by looking at their extension, which is a set of bytes in the file's header that tells the investigator what kind of file it was. Metadata are often disrupted when a file is encrypted; therefore, the ransomware must append it. In cases where relevant data were absent or when the file type was observed to have changed over the file's lifespan, we had reason to suspect malicious intent, in particular, if this characteristic holds for a large number of files.
- (e) Files infected with ransomware would have their names
 - i. altered to something completely arbitrary,
 - ii. a cryptographic hash of the perpetrator's file name (often micro, cerber3, legion, onion, etc.), or
 - iii. the same name as the infected file.

Following the steps in Figure 1, we assessed the 'information gain' through each set of attributes and then combined the required data. This explains the GTK+ message digest and checksum estimator application (Gtckhash) running on the Tonido cloud platform through the Nautilus file manager plugin. Gnome-43 of the PureOS desktop uses Nautilus as its primary file manager. Gtckhash's principal uses were in file management and file viewing. Hereby, the attribute relation files were examined in consideration with malicious indicators, such as privileged access controls, major and minor linker version attributes, size attributes (i.e., size of code, size of initialized data, size of stack and heap reserve/commit), base of code and data, loader flag, number of relocations, pointer to data, etc. For initial evaluation, we considered the following files (ext.: .avi, .csv, .dbase3, .eps, .mp3, .log, .sgml, .troff, etc.). Given that the header is just 8 bytes in size, we have used it as a significant metric in our research. By not altering the 8-byte sequence, sophisticated ransomware might evade the detection method. To satisfy analysis requirements, the 'attribute relation files' were evaluated periodically. Examined criteria were based on the following attributes: read-only, hidden, system, directory, archive, execute, permission indicators, changes, preserve-root, chmod, version, etc.

We observed multiple infection scenarios:

- (a) There were no distinguishing features between file types.
- (b) Each filename extension was unique and unfamiliar. File names were often safe to trust; however, some types of ransomware (such as Phobos ransomware, etc. [18,19]) tend to hash the source file name, thereby rendering the new file name incomprehensible to users.
- (c) There was a random assortment of files with unknown filename extensions.
- (d) Each file extension correctly identified; however, the files were compromised.

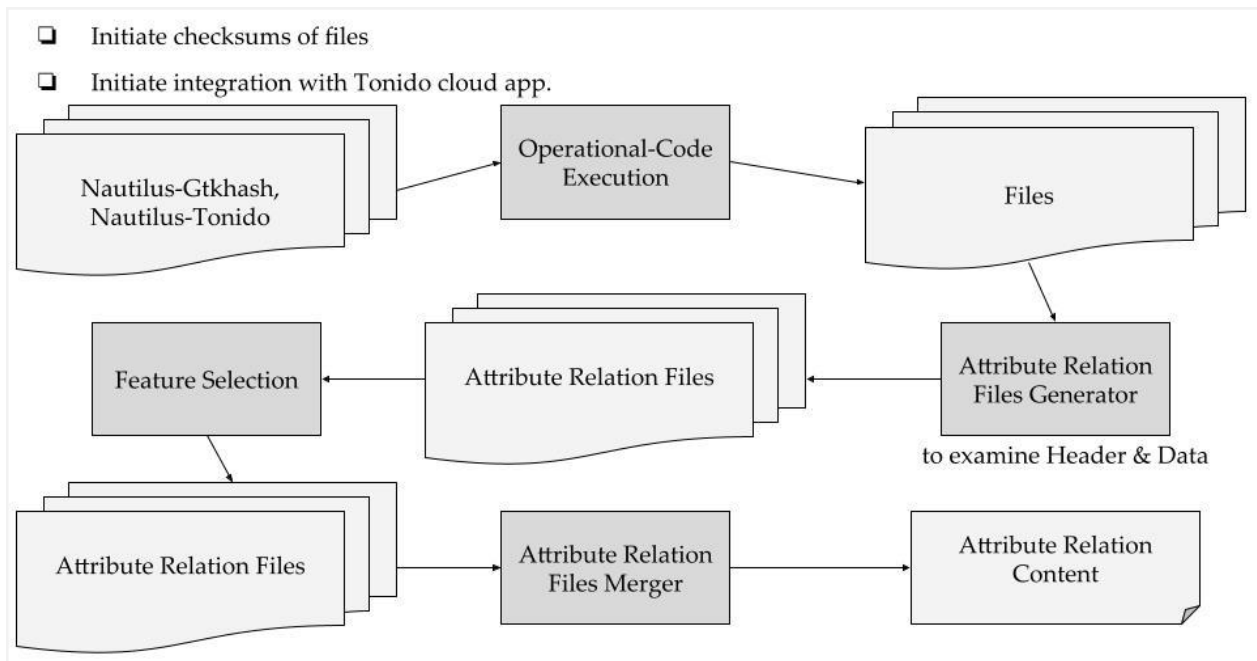


Figure 1. An outline of our methodology for the analysis.

The steps for collecting and analyzing function calls are depicted in Figure 2. The applied application is used to run each operation for predefined intervals (i.e., 90 s) in order to monitor the engagement between system calls as the transaction is being carried out. The acquired log file contains variables and return values, both of which are normalized in order to retrieve the signature of the function call without the attributes and subprograms.

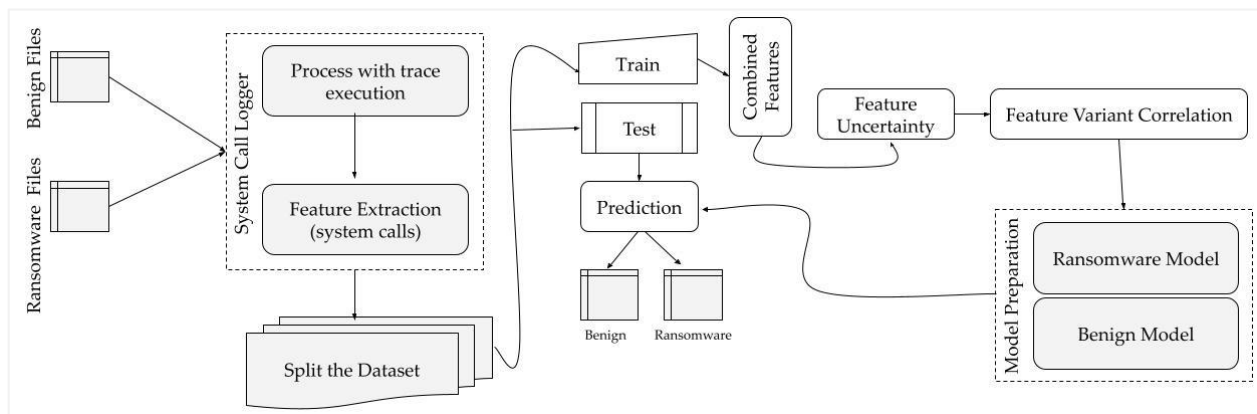


Figure 2. The procedures for collecting function calls are as follows: step one: extract each instance from the feature set; step two: run employing reference and feed the logfile; step three: support for hypothesis, analyze the similarities, and predict unknown patterns as safe or malicious.

In accordance with Table 3, we have assigned threat levels depending on the following criteria by using the setup preparation that is outlined in Algorithm 1:

- (a) *Extremely unusual*: The file was stored on disk space, its content was encoded, and its name and extension seemed unusual.
- (b) *Unusual*: After the file was saved to the disk or drive, its representation was analyzed to determine whether or not it should be treated as a zipped or hashed file. If the file is encoded, only the filename or extension needs to be suspect; if it is zipped, both must be.

- (c) *Usual*: The data were written to disk and were either encoded or zipped using common file naming conventions and file extension patterns, or they were completely random files with no discernible pattern.
- (d) *No data*: Access was denied due to file renaming or deletion.

Threat landscape was marked on the basis of ‘trust boundaries of the system, information elements & classification, attack vectors, malware abuse use case (i.e., anomalous file encryption (partial or full dataset), file deletion, extensive file extension modifications, intensity of file/folder relocation, etc.), threat impact/possibility/ease-of-exploitation’.

Algorithm 1. A basic method for preparation of the proposed setup

1. **while** Failed to meet terminal condition criterion **do**
 2. Collect a small dataset that includes both malicious and safe applications.
 3. Generate ransomware prediction results and calculate their value.
 4. Acquire the ransomware’s outputs that are predicted to be substituted by fairly benign system-calls.
 5. Collect the victim’s performance from running the malicious and benign applications.
 6. By adjusting the replacement weights, reduce the error rate on benign and malicious datasets.
 7. **end while**
-

In order to diminish the basis of multi-valued feature selection, we have evaluated “Information Gain” entropy. The term “information gain” refers to the difference in entropy that exists between a dataset’s original state and its new state after undergoing a transformation. Here, entropy is defined as the smallest set (i.e., S) of bits that encodes the label for each random element in S with uniform probability. In this context, we defined entropy as the smallest collection (i.e., S) of bits that can encode the label for each random element in S with the same probability.

The entropy of a dataset can be approximated, for instance, in a task involving binary detection (two modules):

$$Entropy = -(k(A) * \log(K(A)) + k(B) * \log(K(B))) \tag{1}$$

Here ‘A’ and ‘B’ refer to the values associated to each selected module, whereas $K = (k_1, k_2, \dots, k_n)$ and the likelihood that an entity belongs in a given category is denoted by the variable k. When working with categorical data variables, the acquired index just does binary separating and returns a “malicious” or “legitimate” verdict, whereas “information gain” considers pre- and post-splitting entropy differences to reflect heterogeneity in base classifiers.

To understand this concept, consider two attributes, ‘D’ and ‘E’, from different classes, then use the difference between the two to determine the information gain. When the value of one property ‘D’ is known, the value of another class attribute ‘E’ can be estimated with less uncertainty. The entropy of ‘E’, denoted by Z(E), is used as a proxy for its worth. Probabilistic likelihood of ‘E’ given ‘D’, Z(E|D), gives the degree of doubt regarding ‘E’ given the value of ‘D’.

$$Y(E:D) = Z(E) - Z(E|D) \tag{2}$$

If we assume that ‘E’ and ‘D’ are two independent parameters with ranges of $[e_1 \dots e_w]$ and $[d_1 \dots d_w]$, respectively, we can evaluate the entropy of ‘E’ as:

$$Z(E) = \sum_{y=1}^{y=w} K(E = e_y) \log_2 K(E = e_y) \tag{3}$$

Here ‘K’ referred as a probabilistic analysis coefficient of a class. Assuming ‘D’, the implied entropy of ‘E’ is

$$Z(E|D) = - \sum_{x=1}^v K(D = d_y) Z(E|D = d_y) \tag{4}$$

Preferably, the “information gain” can be

$$Y(E : D) = Z(D) + Z(E) - Z(D, E) \tag{5}$$

This is the case when ‘D’ and ‘E’ have a joint entropy, denoted by $Z(D, E)$:

$$Z(D, E) = - \sum_{y=1}^w \sum_{x=1}^v K(D = d_x, E = e_y) \log_2 K(D = d_x, E = e_y) \tag{6}$$

When the predictor variable ‘D’ is consistent, the information gain of ‘D’ with class attribute ‘E’ was evaluated by comparing all alternative continuous descriptors when we set a criterion. The value of the attribute was derived from each of the values of ‘D’. Thus, the information gain was simply:

$$Y = (E : D) = \operatorname{argmax}_{D_\theta} Y(E, D_\theta) \tag{7}$$

Here θ refers to the threshold that is applicable to ‘D’, whereas the ‘argmax’ input regulates the maximum size of the function’s output in the context of its parameters.

In order to diminish the consideration of dump or non-essential data during evaluation, we measured the dataset impurity as:

$$\text{Entropy} = K_y \log_2 K_y \tag{8}$$

Here, K_y referred as probability of class ‘y’. Hence, when entropy increases, so does the amount of available information.

We looked into Correlation-based feature selection to comprehend symmetrical ambiguity (SA). The significance of an attribute in relation to a classifier is evaluated using symmetrical ambiguity. We measured how numerous factors interacted with the rest of the variables in the feature subset by averaging the measured interactivity gain of all the other factors and the classifier. We measured correlation as:

$$SA = 2.0 \times \frac{Z(D) + X(E) - Z(D + E)}{Z(D) + Z(E)} \tag{9}$$

Symmetrical ambiguity could take on a relevance between 0 and 1. If D or E were 1, then it was 100% predictive of the other. When both parameters were at their default settings of 0, they were considered to be unrelated. After a fixed collection of trustworthy attributions had been created, we devised a method to systematically examine them and drew out relevant data. Particularly, by inspecting the criterion values of the estimations, we grasped the typical function of the characteristics onto the samples’ classification. Because of the attribution approaches we adopted, we were able to provide a comprehensive justification for each forecast. We observed that considerable bias existed in our context with regards to the dissemination of pertinent information. It was possible that the inference values were zero in some cases, while in others they were nestled at a precise assessment (positive if the attribute conforms towards the ransomware group; negative otherwise). Choosing an appropriate synthetic measure allowed us to determine the central tendency of such a distribution. The median was also considered in our analyses. Here, we emphasized a feature as important if it did not display a relevance of “0” in the vast majority of trials. Although it might seem restricting to utilize a single synthetic measure, we argued that informative insights could be gleaned through studying representative samples.

5. Experimental Analysis

After providing context for the experiments in section A, feature selection is described in section B, and we conducted preliminary tests in section C that will allow us to evaluate the hypotheses that are created using a variety of criteria.

5.1. A. Settings

We used a binary classification system, wherein the classification methods were taught to distinguish between ransomware samples and trustworthy samples. The operation's settings and execution are described in this section here.

We employed the data source in [20], which comprised 474 total samples that included 31 trustworthy instances, 180 crypto miners, 44 memory dumps, 50 .RAT rating files, and 169 ransomware samples, and whose latest update date was identified as 29 July 2022. A simulation model in the VMware NSX sandbox [21] yielded ransomware sample strings. When combined with Full-system Mirroring, NSX Sandbox provided the proposed method with the ability to accurately recognize and evade even the most sophisticated of attacks. We also ran our script in Cuckoo Sandbox [22] to understand how the file acts when being run in a realistic yet alternate contained setting. To better understand these associated settings, we investigated two vastly different sandboxes to identify the totality of conditions that make a particular intervention possible. A total of 550 features make up the feature vector. About 215 of those are instances of calling API packages. All of PureOS's application programming interfaces are compiled here. Table 4 illustrates a brief outline of the ransomware variants used for testing.

Table 4. Generic overview of analyzed ransomware variants.

Ransomware	Encoding	Lock	Remote Access Trojan	Samples
Cloud Snooper	✓	✓	✓	10
HiddenWasp	✓	✓	✓	15
Hajime		✓	✓	10
JISUT-variant	✓	✓	✓	20
Kapuser		✓	✓	22
Kryptik	✓			35
Lockerpin ⁺	✓	✓	✓	25
NightSky-Log4j2	✓			20
QNAPCrypt	✓		✓	12

5.2. B. Feature Selection

During evaluation, we experienced that including more characteristics in the dataset improved the accuracy of classification. However, maintaining a large number of unnecessary features might slow down the learning process and lower the precision of the final classification performance. Classifiers might become confused by unnecessary or duplicate features, lowering their recognition accuracy. Because of this, it is crucial to reduce the high dimensionality of feature occurrences by discarding superfluous features. We conducted a phased process for selecting features to isolate the predictor variables' properties that proved most useful in determining the type of process being tested. We eliminated superfluous features from our information source before running the classifier. Typically, it is the platform that determines whether or not a given set of attributes will be retained or discarded. As a result, we have given the PureOS platform's attributes greater weight during the process of selecting features. A selection of the feature criteria we used to evaluate are highlighted in Table 5.

5.3. C. Hypothesis and Testing

It was not a simple process to develop a predictive model that could anticipate the attributes of ransomware dissemination for all families, as distinct malware developers would likely use diverse approaches to developing their script. In addition, there is a significant possibility that the malicious software will evolve and adjust over time. This is

due to the fact that some ransomware content may be made accessible to other malware producers and exchanged among them. Nonetheless, it is logical to assume that there is a consistent order to the steps taken by different types of ransomware.

Table 5. Outline of distinct static features (sample set).

Feature Name	Description
Suspend/resume issues	The problem originates from the middleware's bypassing of sibling threads and its interaction with suspend/resume. The hyper-threading in the BIOS is vulnerable to malware.
Kernel Driver Callback	This function's log shows details about the 'register' function that generated the callback. The operator seeking information is responsible for learning the callback object's identifier and the meaning of the parameters it receives. Anomalies can be triggered when using commands such as InitializeObjectAttributes and ExRegisterCallback that have the potential to misinterpret the CallbackContext parameter.
systemd	The log of this feature reveals the information about 'preserved visual memory', 'capabilities assigned to manage power', and 'system configuration'.
systemd-suspend.service	The process involved in suspending the platform was handled by the "systemd-suspend.service" service, which was triggered into action by the "suspend.target" system file.
suspend-to-RAM (STR)	When a mobility-aware computer goes into a low-power mode, STR happens. System calls invoke RAM to store system settings, running programs, and currently open files when the rest of the computer is idle. Malicious actors now can easily trigger a buffer-overflow anomaly to restrict operators to use computing resources. Moreover, in the event of a power interruption (i.e., raised by a malevolent action), the device would restart automatically.
Disabling Application-Isolation	User access control rights were employed to circumvent 'micro-segmentation' by turning off application-isolation mechanisms. An active application-isolation feature can hinder hackers exploiting remote access. Once a user has been authorized provision on a network, whether through a VPN (virtual private network) service or a proxy server, the user could easily browse across the network without further authentication or authorization. This means that if an anomaly was able to breach the system's perimeter protections, it would gain access to the entire system.
CRYPTSETUP	This feature made it easier to deploy the cryptographic DMCCrypt kernel module.

We analyzed the API (Application Programming Interface) component calls prompted by various threat vectors in PureOS using the anomalous malwares named in Table 4 to learn more about the possible steps and processes involved in ransomware propagation. Due to the fact that different aspects of the PureOS API were called upon quite frequently in the datasets that we investigated, the process of configuring ransomware took a considerable amount of time. Because of this, we decided to produce a complete analysis of all the dataset factors that were incorporated into the categorization strategy of the model. The following hypothesis guided our investigation:

- (a) The extent of damage caused by ransomware varies according to the size of the targeted enterprise.
- (b) The intensity of a ransomware strike on a honeypot-enterprise-setup varied depending on its tolerance-group.
- (c) The magnitude of a network intrusion was impacted by the state of security governance within an entity.
- (d) A malware attack's intensity was affected by the crypto-propagation ransomware's type.
- (e) The intensity of the disruption caused by ransomware varied according to the type of intrusion, which can be either assertive or tailored specifically to target an exclusive entity.

Upon closer inspection, ransomware showed itself in a variety of forms. Keep in mind that not all ransomware will unfold in a specific manner and that the order in which the processes transition could also vary. The proposed strategy for predicting the spread of ransomware used the following models.:

- (a) The creation of a fingerprint, which includes credentials of the operating system's attributes and the assessment of whether or not it is suitable for payload distribution.
- (b) In order to transmit, defensive method must investigate the likelihood of linear distribution of payload inside a system or between interconnected devices.
- (c) Transmitting and receiving information from the command and control server controlled by the adversary.
- (d) Tracing the defendant's actions by examining the data stored in relevant files.
- (e) Protecting sensitive information stored on the defendant's workstation by encoding it.
- (f) Limiting or blocking the defendant's access to the operating system.
- (g) Configuring the features of the information relating to the defendant by either resetting them or disconnecting them.
- (h) Use of a software-driven threatening message to coerce a payment from the target.

Hence, if a platform keeps track of every time a file is viewed or written without authorization, an applied system could easily stop ransomware in its tracks. Yet doing so would significantly degrade the device engagement by increasing the likelihood of the misclassification of legitimate programs, including indexing applications. Finding the right middle ground is essential, ideally with a fairly low false outcome for participants.

In order to better understand the creation and response to ransomware binaries, four separate experiments were carried out. Half of the trial batch was used for training purposes and the other half was used to evaluate the proposed technique in each session. The results from each experiment are shown in Figure 3. Two of the most important goals of these detection trials were: (a) to analyze directories using signature-based identification to discover sensitive data and tendencies connected with malware; and (b) to use policy data analysis to discover previously unseen ransomware variants. As a means of enhancing the platform's ability to detect anomalies, the transaction reaction record was gathered, analyzed, and validated following each interaction. The ransomware under study was eligible to conduct a system-wide search to locate the important files and folders before settling on a strong ciphering, locking and/or RAT configuration. The duration of each iteration was unique because it was based on the input synthesis used for testing. After testing the runtime utility with multiple applications, we were able to validate the necessary timeframe.

We conducted a number of studies to determine how effective our proposed strategy was at detecting ransomware. By comparing the checksum of a ransomware executable with attack signature records, we could determine its accuracy, precision, f-measure, false/positive and false/negative ration. This allowed relatively speedy transient scanning of data in the context, which was helpful for identifying malicious ransomware variants. In our context, precision referred to the proportion of empirical evidence across all extracted feature occurrences, whereas recall signified the fraction of all correctly identified that were actually derived. For this reason, the importance of relevance was evident in both accuracy and recall.

$$precision = \frac{|\{relevant Sample_{set}\} \cap \{retrived Sample_{set}\}|}{retrived Sample_{set}} \quad (10)$$

Hence, precision was estimated with data outcome presented in Table 6.

$$precision_{estimation} = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (11)$$

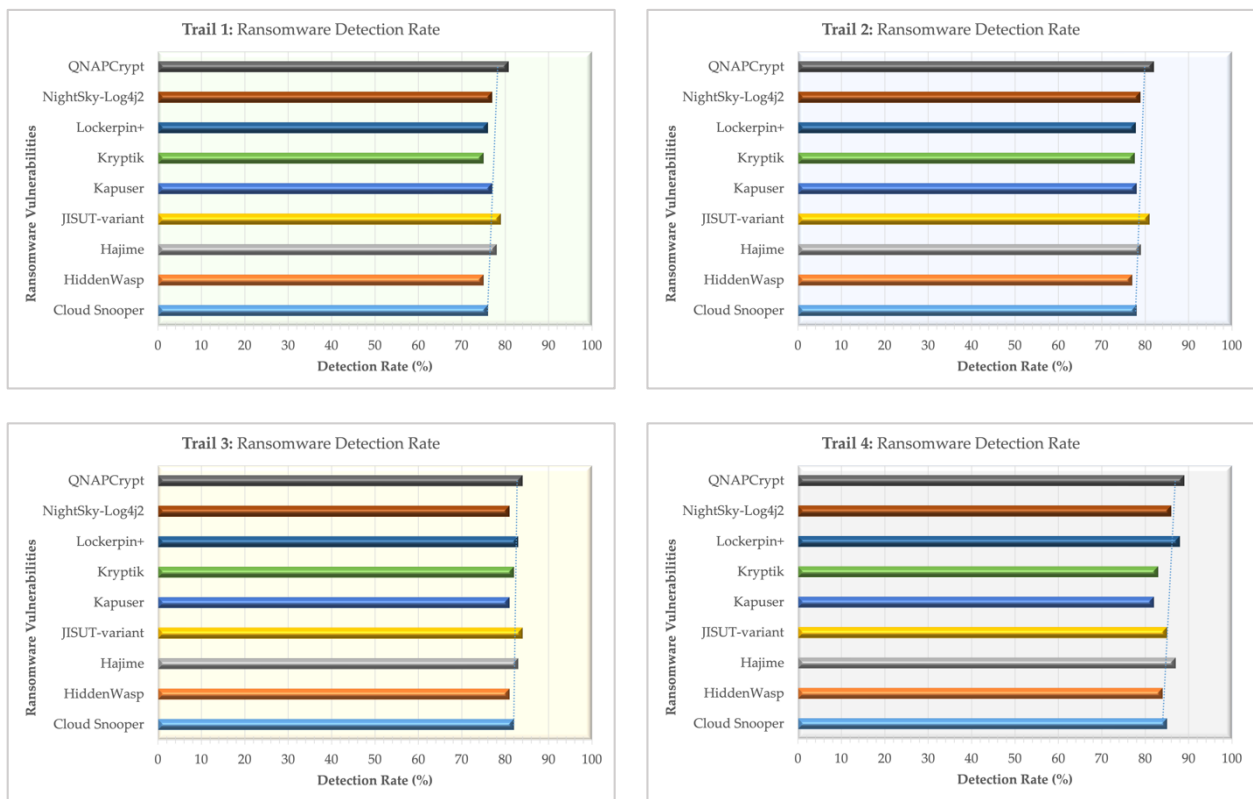


Figure 3. Detection percentage for ransomware (Trail 1, 2, 3 & 4).

Table 6. The performance of Proposed Method on Ransomware anomalies (sample data).

Anomalies (All Samples)	Accuracy	Precision	Recall	F-Measure	False/Positive	False/Negative
Cloud Snooper	85	0.853	0.841	0.851	3.51	3.47
HiddenWasp	84	0.848	0.839	0.842	2.47	2.35
Hajime	87	0.872	0.868	0.871	3.58	3.41
JISUT-variant	85	0.851	0.831	0.849	3.16	4.11
Kapuser	82	0.822	0.802	0.811	2.79	2.95
Kryptik	83	0.835	0.833	0.825	2.45	2.21
Lockerpin ⁺	88	0.889	0.866	0.875	4.13	3.94
NightSky-Log4j2	86	0.862	0.834	0.855	1.16	2.01
QNAPCrypt	89	0.897	0.859	0.804	0.6	1.1

Whereas,

$$Recall = \frac{|\{relevant Sample_{set}\} \cap \{retrived Sample_{set}\}|}{relevent Sample_{set}} \tag{12}$$

$$Recall_{estimation} = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{13}$$

In order to establish a balance between precision and recall, F-Measure is estimated.

$$F_{Score} = \frac{Precision \times Recall}{Precision + Recall} \tag{14}$$

During the training phase, a false positive happened when the output implied the presence of an anomalous status when it did not exist, and a false negative occurred when the outcome implied the absence of a criterion when it did exist. We integrated the Naïve Bayes classifier [23], which is a great fit for our learning paradigm to determine whether an indicator is legitimate or malicious and has been employed to further validate the proposed scheme. As opposed to dealing with empirical system parameters, Naive Bayes performed better while dealing with conditional model parameters. In addition, this technique facilitates the formation of hypotheses and estimates.

We revisited the effectiveness of our feature selection criteria by comparing information gain and chi-square channelled by the Naïve Bayes classifier. Ten, twenty, thirty, forty, and fifty features were used to create the feature sets. Table 7 illustrates the qualities shared by the two approaches (information gain and chi-square). Four metrics including True Positive, False Positive, Precision, and F-Measure were used to examine the classifier’s effectiveness. Table 7 displays the classification performance. Outcome data showed the results from a detection procedure employing a Bayesian predictor with attributes chosen via the information gain and chi-square techniques. There was a positive correlation between the amount of characteristics used and anomaly detection across both techniques. It appears that accuracy improves when the attributes are optimized. These strategies performed similarly inadequately on 20–25 characteristics, though. Chi-squared was evaluated as:

$$Chi - Squared = \sum \frac{(Observed\ value - Expected\ Value)^2}{Expected\ Value} \tag{15}$$

Table 7. The detection effectiveness using the Naive Bayes method.

Features Optimization	Adopted Feature Count	True Positive Rate (%)	False Positive Rate	Precision	F-Measure
Information Gain	50	85	3.50	0.85	0.84
	40	84	3.45	0.84	0.84
	30	82	2.7	0.92	0.82
	20	81	2.37	0.95	0.81
	10	80	1.97	0.94	0.80
Chi-Square	50	92	3.45	0.83	0.84
	40	91.7	3.41	0.82	0.82
	30	91.65	2.96	0.80	0.81
	20	91.61	2.91	0.78	0.79
	10	91.01	1.94	0.77	0.77

Comparative analysis, as we understand it, is the process of recognizing different methodologies and identifying the aspects in which they are identical to and distinct from each other. The methodological comparison is depicted in Figure 4 in which parameters such as logical access, reinforcement, etc. were highlighted. Our evaluations were predicated on a non-active examination of eight security mechanisms, thirty targets, and tens of security measures. We found that all of these ransomware collectives employ a similar tactic: they use previously exploited vulnerabilities, such as those in RDP platforms or via spear phishing, to launch their strikes. During the course of our experiment, we gathered PureOS performance metrics and discovered that several ransomware strikes made greater use of the additional computational resources, whereas other ransomware was remarkably resource-light. No clear relationship could be established between the proportion of resources (i.e., computation, memory, energy, bandwidth, etc.) a sample used and how quickly it could cipher its data. When tested on the more powerful hardware, certain ransomware strains underperformed and sometimes even failed.

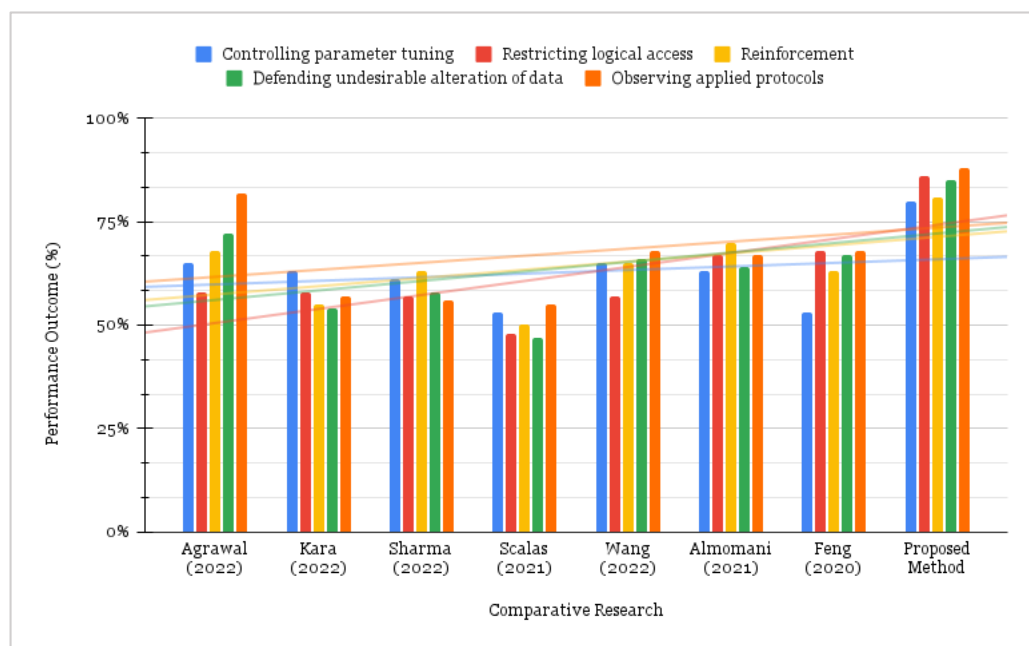


Figure 4. Comparative Analysis [8–12,14,15].

The results of the study also showed that the acceptable assessment of reported data is impossible due to the correlation between dataset size and rated scanning performance. Additionally, it is precarious to draw final judgments regarding whether a technique is “better” without understanding the nature of the selected data size-accuracy curve for the given experimental data. The results also indicated that shifting standards from accuracy to recall at a corrected false positive rate may not be sufficient on their own to provide a realistic comparison, owing to the similarity between dataset size and “real-world” outcomes. Even when using a bigger data set and more lenient criteria, a class imbalance proportion of at least 0.5 is required for the comparison to be meaningful. Taking into account the information in Table 6, we came to the conclusion that accuracy should not be used as a performance criterion for any approach if the dataset is highly imbalanced.

Nonetheless, the proposed approach allowed the malfunction to be discovered quickly, and the propagation of malware was halted within minutes. The extent and frequency of vulnerabilities are only expected to rise in future years, making ransomware a major danger to businesses of all kinds. According to the findings of this study, the most effective course of action is proactive intervention. Having the appropriate security policies and practices in place can help reduce the likelihood of a ransomware attack and even aid in the recovery process by allowing one to restore critical systems and data from servers in the event that an intrusion does occur.

6. Conclusions and Future Work

Malicious software known as ransomware encrypts data and then demands payment to decrypt it. Most ransomware subtypes are based on slightly different versions of the malware’s command and control software. Malicious software must first get access to a target system in order to make a significant impact. Various variants of ransomware encrypt either the entire system or individual files. Traditional security applications require time-consuming and resource-intensive manual analysis of all dangerous file types in the system. In this study, we used industry standard criteria to explain the formidable challenges of countering ransomware. We developed a parameterized classification scheme for function classes and suggestive features using real-world malware samples as a basis. We also detailed a set of criteria that considers both behavior and insights and emphasizes those that are frequently featured. We implemented a Tonido cloud platform that enabled centralized and decentralized file sharing, synchronization, backup, and remote access

across the applied experimental setup. As a result of implementing a novel ransomware detection mechanism, data storage now has the added benefit of enabling users to effectively identify and prevent existing and future ransomware anomalies with a suitable accuracy of 89%.

A surprising, yet clear, message emerged from the statistics. When the proportion of benign-ware to malware attacks increased beyond 1:1, the performance of certain testing iterations appeared to drop dramatically. This study provided the research team with confidence that the performance indicators we developed accurately reflected reality after the imbalance exceeded 2:4, while even balanced (1:1) testing sets may be suitable for evaluation if the training dataset is sufficiently large.

In the future, we hope to be able to analyze unstructured data in near real-time to spot and stop prospective threats as soon as they emerge. We intend to test the full capabilities of the extended system across any public cloud environment without compromising security features or allowing unauthorized access to sensitive information. Instead of paying a ransom to an adversary, we want to test the ease with which files can be restored by enforcing the immutability feature of the data.

In addition, we plan on including a convolutional neural network (CNN) into our proposed strategy to improve malware family classification and decrease false positive detections.

Author Contributions: Conceptualization, A.I. and U.T.; methodology, A.I., U.T. and F.G.; software, U.T. and A.I.; validation, U.T. and T.A.A.; formal analysis, A.I.; investigation, A.I. and U.T.; resources, U.T. and T.A.A.; data curation, A.I. and U.T.; writing—original draft preparation, A.I., U.T. and B.T.; writing—review and editing, A.I., U.T. and B.T.; visualization, B.T. and U.T.; supervision, A.I.; project administration, A.I. and F.G.; funding acquisition, A.I. All authors have read and agreed to the published version of the manuscript.

Funding: Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia, project number (IF2-PSAU-2022/01/21637).

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Deanship of Scientific Research, Prince Sattam Bin Abdulaziz University, Saudi Arabia.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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 (IF2-PSAU-2022/01/21637).

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

References

1. Masid, A.G.; Higuera, J.B.; Higuera, J.-R.B.; Montalvo, J.A.S. Application of the SAMA methodology to Ryuk malware. *J. Comput. Virol. Hacking Tech.* **2022**, *1*–34. [[CrossRef](#)]
2. Yamany, B.; Elsayed, M.S.; Jurcut, A.D.; Abdelbaki, N.; Azer, M.A. A New Scheme for Ransomware Classification and Clustering Using Static Features. *Electronics* **2022**, *11*, 3307. [[CrossRef](#)]
3. Eliando, E.; Purnomo, Y. LockBit 2.0 Ransomware: Analysis of infection, persistence, prevention mechanism. *CogITo Smart J.* **2022**, *8*, 232–243. [[CrossRef](#)]
4. Pitney, A.M.; Penrod, S.; Foraker, M.; Bhunia, S. A Systematic Review of 2021 Microsoft Exchange Data Breach Exploiting Multiple Vulnerabilities. In Proceedings of the 2022 7th International Conference on Smart and Sustainable Technologies (SpliTech), Split/Bol, Croatia, 5–8 July 2022; pp. 1–6. [[CrossRef](#)]
5. Turner, A.; McCombie, S.; Uhlmann, A.J. Ransomware-Bitcoin Threat Intelligence Sharing Using Structured Threat Information Expression. *IEEE Secur. Priv.* **2022**, *2*–12. [[CrossRef](#)]
6. Threat Intelligence Team. Ransomware Review: August 2022. 8 September 2022. Available online: <https://www.malwarebytes.com/blog/threat-intelligence/2022/09/ransomware-review-august-2022> (accessed on 29 October 2022).
7. Cloud. Tonido—Run Your Personal Cloud. A Free Private Cloud Server. 25 October 2022. Available online: <https://www.tonido.com/> (accessed on 29 October 2022).

8. Preeti; Agrawal, A.K. A Comparative Analysis of Open Source Automated Malware Tools. In Proceedings of the 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 23–25 March 2022; pp. 226–230. [CrossRef]
9. Kara, I.; Aydos, M. The rise of ransomware: Forensic analysis for windows based ransomware attacks. *Expert Syst. Appl.* **2021**, *190*, 116198. [CrossRef]
10. Sharma, S.; Krishna, C.R.; Kumar, R. RansomDroid: Forensic analysis and detection of Android Ransomware using unsupervised machine learning technique. *Forensic Sci. Int. Digit. Investig.* **2021**, *37*, 301168. [CrossRef]
11. Scalas, M.; Rieck, K.; Giacinto, G. Explanation-Driven Characterization of Android Ransomware. In *International Conference on Pattern Recognition*; Springer: Cham, Switzerland, 2021; pp. 228–242. [CrossRef]
12. Wang, S.; Zhang, H.; Qin, S.; Li, W.M.; Tu, T.; Shen, A.; Liu, W. KRProtector: Detection and Files Protection for IoT Devices on Android Without ROOT Against Ransomware Based on Decoys. *IEEE Internet Things J.* **2022**, *9*, 18251–18266. [CrossRef]
13. Shah, I.A.; Mehmood, A.; Khan, A.N.; Elhadeif, M.; Khan, A.U.R. HeuCrip: A malware detection approach for internet of battlefield things. *Clust. Comput.* **2022**, 1–16. [CrossRef]
14. Almomani, I.; Qaddoura, R.; Habib, M.; Alsoghyer, S.; Al Khayer, A.; Aljarah, I.; Faris, H. Android Ransomware Detection Based on a Hybrid Evolutionary Approach in the Context of Highly Imbalanced Data. *IEEE Access* **2021**, *9*, 57674–57691. [CrossRef]
15. Feng, J.; Shen, L.; Chen, Z.; Wang, Y.; Li, H. A Two-Layer Deep Learning Method for Android Malware Detection Using Network Traffic. *IEEE Access* **2020**, *8*, 125786–125796. [CrossRef]
16. Yadav, P.; Menon, N.; Ravi, V.; Vishvanathan, S.; Pham, T.D. EfficientNet convolutional neural networks-based Android malware detection. *Comput. Secur.* **2022**, *115*, 102622. [CrossRef]
17. Azeez, N.; Odufuwa, O.; Misra, S.; Oluranti, J.; Damaševičius, R. Windows PE Malware Detection Using Ensemble Learning. *Informatics* **2021**, *8*, 10. [CrossRef]
18. Community. A Fully-Convergent, User Friendly, Secure and Freedom Respecting OS for Your Daily Usage. PureOS. 30 October 2022. Available online: <https://pureos.net/> (accessed on 2 November 2022).
19. Davies, S.R.; Macfarlane, R.; Buchanan, W.J. Comparison of Entropy Calculation Methods for Ransomware Encrypted File Identification. *Entropy* **2022**, *24*, 1503. [CrossRef]
20. Dataset, V.S. Tau-Research/2022-H1-Exposing-Malware-in-Linux-based-Multi-Cloud-Environments at Main Vmware-Samples/Tau-Research. GitHub: VMware Threat Report 2022: Dataset Metadata. 29 July 2022. Available online: <https://github.com/vmware-samples/tau-research> (accessed on 7 November 2022).
21. Sandbox. NSX Sandbox | VMware. Full-System Emulation Sandbox for Accurate Threat Analysis. 19 October 2022. Available online: <https://www.vmware.com/products/nsx-sandbox.html> (accessed on 7 November 2022).
22. Sandbox. Cuckoo Sandbox—Automated Malware Analysis. Analyze Many Different Malicious Files. 19 June 2019. Available online: <https://cuckoosandbox.org/> (accessed on 7 November 2022).
23. Chen, S.; Webb, G.I.; Liu, L.; Ma, X. A novel selective naïve Bayes algorithm. *Knowl.-Based Syst.* **2019**, *192*, 105361. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.