information

MDPI

Article

A Comparison of Neural-Network-Based Intrusion Detection against Signature-Based Detection in IoT Networks

Max Schrötter *, Andreas Niemann and Bettina Schnor *

Department of Computer Science, University of Potsdam, 14476 Potsdam, Germany; andreas.niemann@uni-potsdam.de
* Correspondence: schroetter@cs.uni-potsdam.de (M.S.); schnor@cs.uni-potsdam.de (B.S.)

Abstract: Over the last few years, a plethora of papers presenting machine-learning-based approaches for intrusion detection have been published. However, the majority of those papers do not compare their results with a proper baseline of a signature-based intrusion detection system, thus violating good machine learning practices. In order to evaluate the pros and cons of the machine-learning-based approach, we replicated a research study that uses a deep neural network model for intrusion detection. The results of our replicated research study expose several systematic problems with the used datasets and evaluation methods. In our experiments, a signature-based intrusion detection system with a minimal setup was able to outperform the tested model even under small traffic changes. Testing the replicated neural network on a new dataset recorded in the same environment with the same attacks using the same tools showed that the accuracy of the neural network dropped to 54%. Furthermore, the often-claimed advantage of being able to detect zero-day attacks could not be seen in our experiments.

Keywords: IDS; dataset; deep learning; signature-based-IDS; IoT

check for updates

Citation: Schrötter, M.; Niemann, A.; Schnor, B. A Comparison of Neural-Network-Based Intrusion Detection against Signature-Based Detection in IoT Networks. *Information* **2024**, *15*, 164. https://doi.org/10.3390/info15030164

Academic Editors: Qianlong Wang, Weixian Liao and Changqing Luo

Received: 29 January 2024
Revised: 16 February 2024
Accepted: 24 February 2024
Published: 14 March 2024

1. Introduction

Intrusion detection belongs to the fundamental techniques in network defense. Therefore, new methods and technologies should be adapted as soon as possible to improve network intrusion detection. Over the past years, machine learning techniques, and in particular deep neural networks, have attracted great interest in the field of network intrusion detection. Several publications exist that show the benefit of this approach [1–11].

The Internet of Things (IoT) enables a wide range of applications, for example, in the field of smart homes, smart agriculture, smart cities, or smart healthcare. According to estimates from the year 2022, the sheer number of connected IoT devices was already 14.3 billion and was expected to rise to 16.7 billion in 2023 [12]. Yet, the huge number of devices connected to the Internet simultaneously enables a large number of possible attacks. In principle, we observe two different attack scenarios that we must distinguish between:

1. Attacks that target an IoT device;
2. Attacks that originate from an IoT device.

In the second case, IoT devices are typically misused for distributed denial of service (DDoS) attacks [13,14]. However, the DDoS attack is usually preceded by an attack targeting and hijacking the IoT device. Hence, intrusion detection systems (IDSs) for IoT environments focus on the first case.

IDSs are classified as signature-based or anomaly-based according to the method of detection of attacks. In a signature-based IDS, the detection of attacks is based on events or patterns, which define the attack. These are referred to as signatures. The patterns are not limited to simple character recognition but can also reflect the behavior of users in the network. With signature-based IDS, therefore, only known attacks can be detected. However, the attack patterns should be precisely defined; otherwise, the risk of false alarms increases. In an anomaly-based IDS, in contrast, attacks are detected by the fact that they differ significantly from the normal behavior of the system, i.e., they form an anomaly. The decision as to whether it is an anomaly or not can be based on the protocol used or statistics. For this purpose, statistical variables such as the mean value or the variance for the behavior of the system are collected. Anomaly-based IDSs, therefore, offer the advantage that it is no longer necessary to define each attack signature individually. However, it cannot be guaranteed in every case that an attack will differ significantly from normal behavior, which would prevent it from being detected by the IDS [15].

In recent years, new approaches have emerged alongside these classic intrusion detection systems, which use machine learning methods to detect attacks and learn patterns from the data. A recent survey from Dini et al. gave a good overview of IDS design exploiting machine learning [16]. They trained and compared seven different approaches, of which decision trees performed best. A survey from 2021 lists about 54 different intrusion detection systems based on machine learning that address the IoT domain [17]. In particular, the use of deep learning IDS (DLIDS) is especially favored in the literature [1–6,9–11]. The term deep learning is used for neural networks that consist of at least four layers [18]. This means that in addition to the input and output layer, at least two hidden layers are required.

Starting with the paper of Arp et al. in 2020 [19], the scientific community has recognized that the pitfalls in the design, implementation, and evaluation of machine-learning-based systems also occur in the context of security systems. Pitfall number 6 stated by Arp et al. is the inappropriate baseline, which is described as follows: “The evaluation is conducted without, or with limited, baseline methods. As a result, it is impossible to demonstrate improvements against the state of the art and other security mechanisms”. Regarding the research in the field of DLIDS in the IoT, this pitfall can be regularly observed in scientific publications: the authors present their promising results, which are compared against a prior ML-based system, but there are no comparisons included with state-of-the-art approaches such as the classic signature-based systems like Snort, Suricata, and Zeek [20–22]. As pitfall 6 describes, this approach does not allow any insight into how well the presented system performs in comparison with the state-of-the-art. The only publication known to us where a comparison against a proper baseline is presented originates from Gray et al. [8], in which the authors compared a random forest classifier against the Suricata intrusion detection system. Suricata was configured only with a subset of the Emerging Threats Open rule set [23] specific to the attacks, but without any configuration for the detection of SSH brute-force attacks, DDoS, or port scans. Therefore, the comparison remains unrealistic.

Cahyo et al. published a survey about hybrid intrusion detection systems in 2020 [24]. They compared 20 papers published between 2016–2020, which all combined a signature-based IDS, for example, Snort, with anomaly-detection where the anomaly-detection was based on an ML-based approach (Gaussian mixture model, K-means, decision tree, . . .). The hybrid approach obviously seems to be promising since it combines the best of both worlds: the signature-based approach to detect known attacks and the ML-based approach to detect anomalies. These hybrid approaches have yet to be adopted in practice. The well-known signature-based systems Snort and Suricata, for example, have not integrated ML-based preprocessors in their released source code.

Overall, we observed that the majority of the publications in the field do not compare their results with a proper baseline. Therefore, the real benefit of ML-based approaches, especially for multiclass detection, must be questioned. To fill this research gap, we conducted a new study to answer the following two research questions:

1. How does a state-of-the-art DLIDS perform in comparison to a signature-based IDS?
2. How well is the DLIDS able to generalize and to perfom in a replicated environment using slightly modified attacks?

To obtain deeper insight in the pros and cons, we performed a comparison study. Instead of creating a new DLIDS, we picked one of the best performing DLIDSs from the literature and conducted an experimental study under different benchmark scenarios. Further, we compared the DLIDS against Snort [20] as a well-known signature-based system.

The contributions of this work are:

1. Evaluation of a DLIDS in a replicated environment using the same attacks with and without slight modifications in the attack pattern. Furthermore, we evaluate the DLIDS against an unseen attack, demystifying the claim of zero-day attack detection.
2. Comparison of a DLIDS against the classic signature-based IDS Snort [20] addressing the pitfall P6—inappropriate baseline [19].
3. Demonstration of the shortcomings of existing training sets and proposing new validation datasets (MQTT-IoT-IDS2020-UP and variations).
4. A discussion of the weaknesses of current applied performance metrics in the context of IDS, (addressing P7—Inappropriate Performance Measures from [19]).
5. We give an overview of IoT datasets to support IDS researchers in selecting a suitable dataset for their research.

2. Related Work

2.1. Discussion of Machine Learning Pitfalls

While machine learning has shown its major potential in many different areas, several researchers have discussed pitfalls in the context of security systems [19,25–27]. The authors of [19] argued that these pitfalls may be the reason why machine-learning-based systems are potentially still unsuited for practical deployment in security tasks.

Arp et al. [19] conducted a study of 30 papers from top-tier security conferences published within the past 10 years which shows that these pitfalls are widespread in the current security literature. Among the 30 publications were four papers from the network intrusion detection area. The authors presented 10 pitfalls which frequently occur in the reviewed papers. Some of them are particularly relevant for network intrusion detection systems (NIDS):

P1 **Sampling bias,** when the datasets used for training and testing do not represent the network traffic (see Section 3).

P6 **Inappropriate baseline,** when new approaches are only compared against very similar approaches, but not against the state of the art (see Section 5.5).

P7 **Inappropriate performance measures**: In the context of NIDS, just one performance metric, like accuracy, is insufficient. Precision is important since a high false positive rate would render the system useless. Further, in the context of NIDS, binary and multiclass classification are both applied. The detailed definition of the performance metrics for the multiclass classification problem is often missing (see Section 5.7).

P8 **Base rate fallacy**: This is similar to P7, but accounts for misleading interpretation of results.

Dambra et al. discussed similar pitfalls in the context of Windows malware classification [27].

Further, there are also several publications which discuss the reliability and the practical benefit of ML-based approaches in the context of NIDS [19,28–30]. Venturi et al. discussed the influence of temporal patterns in malicious traffic on the detection performance [28]. They experimented with an ML-based NIDS which was trained on a public dataset for bot detection [31]. They observed a drastic performance drop when just 1% of benign flows are injected at the end of the training set. The authors concluded that their findings raise questions about the reliable deployment of ML-NIDS in practice [28]. A similar result was recently reported by Aldhaheri and Alhuzali [29]. The authors compared five different ML-based models (support vector machine, naïve Bayes, LSTM, logistic regression, and K-nearest neighbors) with generated synthetic adversarial attack flows. The detection rate for these slightly adapted attack flows was reduced for all five IDSs by an average of 15.93%. For example, the precision rate for detecting DDoS attacks dropped for LSTM from 98% (test set from the CICIDS2017 dataset [32]) to 73% (new generated DDoS attack traffic).

In a recent publication from Zola et al. [30], the authors stated that most of the ML-based approaches ignore a key feature of malware: its dynamic nature since malware is constantly evolving. They concluded that it may be more beneficial to train models on fewer but recent data and retrain them after a few months in order to maintain performance.

Further, ML-based approaches are the target of attacks themselves. Tidjon et al. published a recent survey of threat assessment in machine learning repositories [33]. Since ML frameworks are also software, they also can possess vulnerabilities. As an example, the authors reported vulnerabilites of five dependencies for TensorFlow with high severity according to the CVSS v3/v2 score (i.e., yaml, sqlite3, icu, libjpeg-9c, curl/libcurl7, psutil). Further, the model creation process can be the target. This is conducted to create models that behave incorrectly. In 2023, for the first time, the Open Worldwide Application Security Project (OWASP) published a Machine Learning Security Top Ten list [34]. The top three attacks are input manipulation, data poisoning, and model inversion. OWASP also started work on an AI Security and Privacy Guide [35].

2.2. DLIDS

Several research groups have proposed deep learning for network intrusion detection in recent years with very good results [1–11,36]. A survey by Ullah et al. [7] already analyzed 37 papers published between 2017 and 2021 regarding DLIDS. Some research also addresses IoT-related attacks targeting the MQTT messaging protocol [1,2,4–6,36].

Alaiz-Moreton et al. trained a deep neural network with their own dataset for multi-class classification [1]. The dataset is available online [37]. Further details about the dataset are given in Section 3.

Ciklabakkal et al. [2] trained a neural network for anomaly detection, i.e., to distinguish benign and attack traffic. They used 8 IP features, 16 TCP, and 6 MQTT features. Since they also trained with source and destination IP addresses, the network will hardly ever perform similarly in any other environment.

Hindy et al. [3] recorded the MQTT-IoT-IDS2020 dataset and applied six ML techniques for multiclass classification. The dataset is publicly available [38] and includes benign traffic and four attack types: aggressive scan (Scan A), UDP scan (Scan sU), Sparta SSH brute-force (Sparta), and MQTT brute-force attack (MQTT BF). They use IP and TCP features, but drop source and destination IP addresses, the upper layer protocol, and the MQTT flags. The remaining features are 7 IP features and 10 TCP flags.

Ge et al. performed both; they trained models for anomaly detection and for multiclass prediction [4] on the Bot-Iot dataset [39]. The dataset is publicly available [40]. They extracted 11,174,616 packets from the dataset and deployed custom preprocessing. The authors dropped, for example, IP destination and source addresses, and TCP and UDP checksums. Further, they merged some features. Due to the preprocessing, duplicated rows were introduced, which had to be removed. The final dataset still consists of 7,310,546 packets. However, the final set of features was not reported.

Khan et al. [5] also trained both anomaly detection and multiclass prediction models. They reused the MQTT-IoT-IDS2020 dataset from [3]. In case of multiclass classification, the network is trained to distinguish five classes: benign traffic and the four attack types present in the MQTT-IoT-IDS2020 dataset. While the MQTT flags were removed in [3], Khan et al. did not remove them and trained the packet-based models with them.

Similar to the work of Khan et al., Mosaiyebzadeh et al. [6] proposed three different DLIDS where the models were again trained with the MQTT-IoT-IDS2020 dataset for multiclass detection. The models were trained for the detection of benign traffic, aggressive scan, UDP scan, and MQTT brute-force authentication attacks. Thus, they excluded the Sparta SSH brute-force attack. The three DL models were a deep neural network (DNN), a long short-term memory (LSTM), and a mix of convolutional and recurrent neural networks (CNN-RNN-LSTM). The models performed similarly with a slight favor for the last one.

Ullah et al. [7] proposed a convolutional neural network (CNN) and gated recurrent unit (GRU) to detect and classify binary and multi-class IoT network data. The proposed model was trained and validated using four datasets: the BoT-IoT [41] and the IoT Network Intrusion dataset, both collected by Ullah and Mahmoud [42], the MQTT-IoT-IDS2020 [3], and the IoT-23 [43] dataset. It should be pointed out that the Bot-IoT dataset from Ullah and Mahmoud is different from the Bot-IoT dataset from Koroniotis et al. [39]. The authors mentioned that they used the 48 best features, the significance of the features being determined using a random forest classifier. The authors compared their very good detection results (over 99.9% accuracy, precision, recall, and F1 score) with previously published DLIDS, but not with a signature-based IDS.

Vaccari et al. published the MQTTset [44], and trained a neural network with this dataset for multiclass prediction [36]. They used 3 TCP-related and 30 MQTT-related features. The authors reported a very high accuracy of over 99%. Accuracy is the ratio of the number of correct predictions to the total number of input samples, but this result is favored by the MQTTset, which consists of 11,915,716 network packets, of which only 165,463 packets (0.01%) belong to the attacks.

Recently, federated learning was proposed as a means to cope with privacy demands [11], but again, no comparison with an adequate baseline was made (Pitfall 6: Inappropriate Baseline).

While several very promising ML-based models have been published, they, too, lack the comparison with an adequate baseline. In summary, we still observe that a deeper investigation of the performance of the presented models is necessary.

3. Discussion of IDS Benchmarks

In 2000, John McHugh pointed out that datasets are a crucial point in training and testing NIDS [45]: Either it should be demonstrated "that the training mix is an accurate representation of the real-world or accurate real-world training data should be available for each deployment environment". The problem becomes even harder due to the diversity of the environments and the dynamic behavior of IT infrastructures and services. For example, if the benign traffic is well known since only few services are available, it is much easier to obtain a realistic benign traffic dataset compared to an environment like a cloud data center with very heterogeneous users. Further, new applications evolve and may initiate new types of traffic. With new applications, new vulnerabilities may also occur. This in turn may lead to new attacks. Hence, our environments regarding benign and malicious traffic are very dynamic in nature.

About 10 years later, Nehinbe surveyed the shortcomings of existing datasets and the problems in creating datasets [46]. Twenty years later, the research community still concludes that many benchmark datasets do not adequately represent the real problem of network intrusion detection [9,47]. In the context of ML-based intrusion detection, the situation is even worse, since the datasets are not only used for evaluation but also for training.

3.1. Popular IDS Benchmarks

Since the quality of datasets is a relevant problem for IDS research, this topic is a recurring research theme (see, for example, [9,46,48]). Hindy et al. gave an overview of datasets, network threats, and attacking tools, which are available and used for collecting datasets [47]. Their analysis of 85 IDS research papers from 2008–2020 showed that most systems use the KDD-99 or its successor the NSL-KDD dataset, which cover only four attack classes. For comparison, the dataset that includes the largest number of attack classes is the more recent CICIDS2017, which contains 14 attack classes.

The CICIDS2017 dataset was released by the Canadian Institute of Cybersecurity [32]. It is not a recording of traffic in a real environment, but the authors recorded *simulated* normal and malicious traffic. The authors addressed the dynamic behavior of IT environments by including the most common attacks based on the 2016 McAfee report. They recorded attacks that are commonly detected at present such as DoS, DDoS, brute-force, XSS, SQL injection, Infiltration, Port Scan and Botnet. The CICIDS2017 dataset includes 14 different attack classes and benign traffic. The protocols covered are HTTP, HTTPS, FTP, SSH and email protocols.

While the CICIDS2017 dataset was highly appreciated by the research community and has been used in intrusion detection research [49–51], it still has some shortcomings [9,52]. Panigrahi et al. criticized the huge volume of data which made it hard to handle, as well as the high class imbalance [52]. Engelen et al. investigated the CICIDS2017 dataset in detail [9] and reported *that 25% of all flows in the dataset turn out to be meaningless*. The reasons are, for example, the misimplementation of one of the attack tools (Dos Hulk attack), and a misunderstanding of the TCP protocol which resulted in additional classified TCP flows. Engelen et al. reported that the CSE-CIC-IDS2018 dataset [32], which is the successor of CICIDS2017, also includes errors in flow construction.

It is worth mentioning that the CICIDS2017 dataset does not include recordings of the MQTT protocol, which is important for IoT environments.

3.2. Benchmarks Specific to the IoT

Message Queuing Telemetry Transport (MQTT) [53] is the most widespread protocol for edge- and cloud-based Internet of Things (IoT) solutions. Hence, an IDS for an IoT environment has to consider MQTT-specific attacks. The communication model follows the publish–subscribe style with a broker as the central heart of the architecture. The broker manages the subscriptions and forwards published messages to the subscribers. Addressing is based on so-called *topic names*.

Here, we will survey only publicly available datasets. An overview is given in Table 1. Among the first to record a dataset with MQTT-specific attacks were Alaiz-Moreton et al. [1,37]. Three different attack scenarios were captured:

1. DoS: A DoS attack against the MQTT broker by saturating it with a large number of messages per second and new connections.
2. Man in the middle (MitM): The attack uses the Kali Linux distribution and the Ettercap tool to intercept the communication between a sensor and the broker to modify sensor data.
3. Intrusion (MQTT topic enumeration): Here, the attacker contacts the broker on its well-known port (1883) and checks whether the broker returns all registered MQTT topics.

Table 1. Classification of IoT datasets. A ✓means that this kind of attack is included in the dataset.

	Datasets						
Category	**MQTT-Specific** [1]	**MQTT-Specific** [2]	**Bot-Iot** [39]	**MQTTset** [36]	**IoT-23** [43]	**MQTT-IoT-IDS2020** [3]	**CICIoT2023** [54]
Year	2019	2019	2019	2020	2020	2021	2023
Benign	✓	✓	✓	✓	✓	✓	✓
Broker DoS	✓	✓	-	✓	-	-	-
MitM	✓	-	-	-	-	-	✓
Topic-Enumeration	✓	-	-	-	-	-	-
MQTT BF	-	-	-	✓	-	✓	-
Port Scan	-	-	✓	-	✓	✓	✓
OS f. p.	-	-	✓	-	-	-	✓
Botnet DoS	-	-	✓	-	✓	-	✓
Data Theft	-	-	✓	-	-	-	-
SSH BF	-	-	-	-	-	✓	✓
C&C	-	-	-	-	✓	-	-

The dataset used in [2] is publicly available, but not documented. The only information given by the authors is that they used the MQTT malaria tool [55] to simulate MQTT DoS attacks.

The Bot-Iot dataset [39] aims to identify IoT environments that are misused as botnets. The dataset was recorded in a virtual network using the Node-Red tool [56] to simulate benign traffic. The recorded attacks are Port Scans, OS fingerprinting (OS f. p.), DoS from an IoT Botnet (Botnet DoS), and Data Theft using Metasploit to exploit weaknesses in the target machine.

The pcapfiles of the Bot-IoT dataset were adapted by Ullah et al. [41]. The new dataset is called IoT Botnet. The authors claim that it has more general network features and more flow-based network features, but no different attack types. From their description, it remains unclear what changes they made.

The MQTTset [36] consists of benign traffic and five attack classes. They recorded four different types of DoS attacks: flooding denial of service with the broker as target, MQTT Publish Flood from a single malicious IoT device, SlowITe, which initiates a large amount of connections with the MQTT broker, and an attack with malformed data sent to the MQTT broker to raise exceptions. Further, they recorded a brute-force authentication attack against the MQTT broker using the `rockyou` word list (MQTT BF). The dataset consists of 11,915,716 network packets including the 165,463 packets (0.01%) belonging to the attacks.

The IoT-23 dataset [43] was published in 2020 and consists of twenty-three captures of different IoT network traffic: three benign IoT traffic and 20 malware captures. While the benign traffic was recorded on three real IoT devices, the malicious traffic was obtained by executing different malware on a Rasberry Pi. Furthermore, the dataset provides eight labels for further information about the attacks: Three labels (Mirai, Okiro, Torii) indicating botnet behavior, horizontal port scan, DDoS from an infected IoT device, file download, or HeartBeat. In addition, if a C&C (command and control) server is part of the attack, this is shown by the label C&C.

The MQTT-IoT-IDS2020 dataset [3] includes benign traffic and four attack types. The latter consist of three generic attacks: Aggressive scan (Scan A), UDP scan (Scan sU), and Sparta SSH brute-force (Sparta). Additionally, an MQTT authentication brute-force attack (MQTT BF) was recorded.

The IoT-CIDDS dataset [10] focuses on an IoT network stack consisting of the following protocols: CoAP, UDP, IPv6, 6LoWPAN, and IEEE 802.15.4. For this environment, the authors recorded five DDoS attacks in the 6LoWPAN network, namely, hello flooding (HF), UDP flooding, selective forwarding (SF), Blackhole (BH), and the ICMPv6 flooding attack.

For the CICIoT2023 dataset [54], traffic from a wide range of different IoT devices (67) was recorded in a lab environment. In this environment, the following attack categories were executed predominantly by Raspberry Pis: DoS, DDoS, Recon, Web-based, brute-force, spoofing, and Mirai.

3.3. Class Imbalance

Class imbalance is an important problem frequently discussed in machine learning fields [19,26]. For network intrusion detection, this is especially relevant since the amount of malicious traffic is highly dependent on the attack type. For example, brute-force attacks and DoS attacks can generate millions of packets per second, while a targeted exploit may contain only a few packets. The same distribution problem occurs for flow data after flow analysis. When recording traffic in a real environment, the amount of benign traffic surpasses the amount of malicious traffic by far, except for DoS attacks. To circumvent this problem, the authors generate malicious traffic from one or a few attacker machines on which popular attack tools are executed. However, with this approach, the number of attack packets/flows is increased, but the variety and variability of the attack traffic is not captured. Furthermore, for the MQTT-IoT-IDS2020 dataset, the imbalance between different malicious traffic types is not remedied, as is further discussed in Section 4.3. The biggest class is Sparta, with more than 61% of the packets. The smallest class is Scan_SU, with only 0.07%. The benign traffic share accounts for only 7%.

The usual recording approach has further drawbacks. For example, attacks are executed in distinct time windows and from a single or only a few sources. This makes it necessary to exclude all features that are influenced by the time and source of the attack. Most publications describe the features that were excluded, for that reason, but features like

TTL can easily be overlooked by the authors. For example, a strong correlation between the packets' TTL value and traffic type was observed for the CIC-IDS-2017 dataset in [57].

4. Reproducibility Study

The FAIR principles were defined by a consortium of scientists and organizations in March 2016 [58]. FAIR stands for findability, accessibility, interoperability, and reusability. The intention behind the FAIR principles is to enhance the reusability of data. This can include datasets like network traces, model descriptions, and/or scripts which were used in the model evaluation. If these data are publicly available together with the published research paper, this would help to ensure transparency, reproducibility, and reusability.

As the scientific community is only slowly becoming aware of the importance of the FAIR principles, it is still not common practice to publish all the relevant data. Here, we present our efforts and difficulties during our reproducibility exercise, starting only with a publication at hand. The publicly available source code, datasets, and configuration files of our experiments can be found at [59].

4.1. Selection of Representative DNN-IDS

Only papers that fulfill the following criteria were shortlisted for our investigation:

1. Only deep learning models are considered.
2. The model is trained for IoT environments, i.e., the training set includes MQTT-specific attacks.
3. We want to compare the selected model against a proper baseline, which is a signature-based intrusion detection system. These perform multiclass classification. For example, a Snort rule is configured with so-called *rule messages* that describe the detected attack. Hence, for a fair comparison, we consider only models that are also capable of multiclass classification.
4. We restrict ourselves to recent publications that offer the best performance as far as we are aware.

Table 2 gives an overview over the publications that are most relevant for our investigation. We selected the following classification criteria: First, which datasets are used? Are both anomaly and multiclass detection investigated? Which machine learning approaches are implemented? What is the baseline used to evaluate the presented approach? And finally, is there enough material available to replicate the model? For the last question, the dataset needs to be publicly available and the source code for creating the model needs to be published or needs to be sufficiently described in the paper.

As a first result, it can be observed that the trend leans toward multiclass detection. The reported performance metrics (accuracy, precision, recall, F1 score) are always very high. This may lead to the impression that DLIDS are very well suited for multiclass detection.

Further, one can observe two different evaluation approaches: some papers implement and compare different ML-based approaches [1–3,5]. These have multiple entries in the *detection method* column and compare a deep neural network (DNN) with other ML-based approaches such as naive Bayes (NB), logistic regression (LR), one-class support vector machines (OCSVM), random forests (RFs), k-nearest neighbors (kNN), decision trees (DTs), long short-term memory (LSTM), or gated recurrent units (GRUs).

Other researchers compare against published results as a baseline, which is laudable. But sometimes these comparisons have to be interpreted with care when the cited performance results are obtained for different datasets [4–7]. Khan et al. took both of these approaches: They compared different implemented approaches, and also against [1] as baseline. Mosaiyebzadeh et al. [6], likewise, took both approaches. They also compared their proposals against a decision tree, which was the best model from [3].

Table 2. Classification of reviewed publications.

	Datasets	**Detection**		**ML-Based**	**Evaluation**	**FAIR Criteria:**
		Anomaly	**Multi-Class**	**Detection Method**	**Baseline**	**Reproducibility**
Khan et al., 2021 [5]	MQTT-IoT-IDS2020 [3]	✓	✓	DNN, NB, RF, kNN, DT, LSTM, GRU	[1]	✓
Alaiz-Moreton et al., 2019 [1]	MQTT-specific [1]	-	✓	DNN, XGBoost, GRU, LSTM	-	–
Ciklabakkal et al., 2019 [2]	MQTT-specific [2]	✓	-	OCSVM, RF, IF, K-Means, SO_GAAL, Autoencoder	-	–
Hindy et al., 2021 [3]	MQTT-IoT-IDS2020 [3]	-	✓	LR, NB, kNN, SVM, DT, RF	-	✓
Ge et al., 2019 [4]	Bot-IoT [39]	✓	✓	DNN	SVM [60]	–
Mosaiyebzadeh et al., 2021 [6]	MQTT-IoT-IDS2020 [3]	-	✓	DNN, LSTM, CNN-RNN-LSTM	DT [3]	✓
Ullah et al., 2021 [7]	MQTT-IoT-IDS2020 [3], BoT-IoT and IoT Network Intrusion [42], IoT-23 [43]	✓	✓	CNN, GRU	[3] and more	–

Even though the work of Hindy et al. [3] introduced the MQTT-IoT-IDS2020 dataset that is used in many publications, we did not consider it for our study, since they used only classical ML-based approaches and no DLIDS.

The models in Ciklabakkal et al. [2] did not involve multiclass attack classification. Further, the IP addresses of source and destination were included in the feature set. This is not recommended in the context of NIDS since attackers will change their IP addresses frequently. Furthermore, the code of the model is not available.

All considered publications worked on publicly available datasets, which shows the effort of the research community to publish comprehensible results. But only some groups published the code of their model(s) and/or gave a precise definition of their data preprocessing. For example, the authors of [1] were aware of the imbalance in their datasets and used Scikit-learn for resampling. But this process was not further described. In the paper of Ullah et al. [7], too many details of the data preprocessing were missing. For example, the list of trained features was not provided. Thus, it is not feasible to reproduce these results.

From the three works that seem to be reproducible [3,5,6], ref. [3] was not considered, as previously mentioned.

Even though Mosaiyebzadeh et al. [6] published their code, they removed the SSH Sparta attacks from the MQTT-IoT-IDS2020 dataset, without providing a reason for doing so. This makes it impossible to compare the results to other publications using the same dataset. Using only a subset of the dataset will most likely have increased the performance of their model because the SSH Sparta attacks were the most difficult to detect for us, as can be seen in Section 5.9. Therefore, Mosaiyebzadeh et al. was not further considered in our study.

The remaining work from Khan et al. [5] was chosen since it seemed to be well documented. Further, the model was trained with the popular MQTT-IoT-IDS2020 dataset, which includes several relevant attacks. Finally, the proposed DLIDS outperforms the other tested ML-based approaches for all three feature categories: packet-based, bi-flow and uni-flow.

Remark 1. *The work selected for our reproducibility study was chosen because it seems to be one of the best papers in the field. The problems discussed should not be interpreted as a finger-pointing exercise, but to showcase problems which are common in the field of ML-based intrusion detection.*

4.2. Detection Metrics

This section describes the metrics that we used in our investigation. For all models, we used 5-fold cross-validation and employed early stopping. To compare the performance to the original publication, we used the same metrics as the authors. Hindy et al. [3] gave a precise definition of the metrics they used to evaluate their multiclass detection experiments, but, in general, this is not the case. Also, in the original publication, the metrics accuracy, precision, recall, and F1 score are not defined for multiclass classification. For multiclass classification, we opted to use macro-averaging, the unweighted mean of the per-class metrics, since the severity of an attack does not correlate with the number of flows or packets included in an attack.

To be comparable to the original publication, the normal/benign class in multiclass classification was treated as all other classes. Therefore, there are no true negatives (TNs) in multiclass classification.

The metrics used for multiclass classification with L classes and n samples are therefore defined as follows:

$$\text{Accuracy}_{\text{MC}} = \frac{1}{n}\sum_{i=1}^{|L|}\text{TP}_i \tag{1}$$

$$\text{Precision}_{\text{MC}} = \frac{1}{|L|}\sum_{i=1}^{|L|}\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \tag{2}$$

$$\text{Recall}_{\text{MC}} = \frac{1}{|L|}\sum_{i=1}^{|L|}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \tag{3}$$

$$\text{F}_1\ \text{Score}_{\text{MC}} = \frac{1}{|L|}\sum_{i=1}^{|L|}\frac{2\text{TP}_i}{2\text{TP}_i + \text{FP}_i + \text{FN}_i} \tag{4}$$

For experiments 2 and 4 in Section 5, where only one attack was analyzed, we used the binary classification metrics, in which correctly identified benign traffic is considered as true negative (TN).

4.3. Replication Problems

The problems discussed here are representative of multiple papers and are only described for the chosen paper of Khan et al. [5]. Even though the chosen paper seemed to be well documented, several inconsistencies were found during the replication process.

The MQTT-IoT-IDS2020 dataset contains three levels of abstraction of features, namely, packet-based, uni-flow, and bi-flow. The packet-based feature set includes all L3 and L4 header values and MQTT header values. The uni- and bi-flow sets include IP address, port information, and flow data. For bi-flow, the flow values for both directions are provided separately. Khan et al. created models for all three feature sets.

P1 Dataset In the first step, we aggregated all records in the MQTT-IoT-IDS2020 dataset from the packet-based CSV files. The resulting class distribution can be seen in Table 3. Our numbers, however, differ significantly from the results of Khan et al. After email consultation with the first author about these differences, we obtained the information that they "might have used techniques like random oversampling, SMOTE (Synthetic Minority Over-sampling Technique), or random undersampling [. . .]" without providing any specific details. Therefore, to achieve a similar class distribution as Khan et al., the classes MQTT brute-force and Sparta were cut off: Since both classes are password brute-force attacks trying passwords from a dictionary, only about the first 200,000 attack packets were sampled. Because of this class imbalance, the folds for cross-validation were stratified. To keep temporal context, the data were not shuffled, and all packets of a flow were kept in the same fold.

Table 3. Class distribution comparison.

Class	MQTT-IoT-IDS2020 [38]	Khan et al. [5]	Replicated Result
Normal	7.21%	72.62%	82.04%
Scan_A	0.13%	2.42%	1.44%
Scan_SU	0.07%	1.34%	0.79%
MQTT Brute-force	31.17%	7.83%	7.83%
Sparta	61.42%	7.90%	7.90%

P2 Missing Value: The missing-value strategy used by Khan et al. uses the median of that feature in order to be less susceptible to outliers. But for the packet-based data, non-TCP or non-MQTT packets do not have any data in the TCP or MQTT features, respectively. Using this strategy results in non-TCP packets having TCP options set. Therefore, we decided to train a second model where we used unset values instead of the median value. For example, for TCP flags, this is zero.

P3 Feature Selection: According to [5], certain features were removed from the uni- and bi-flow dataset, but no complete list was provided. Also, the encoding and normalization was not specified for all features in the packet-based dataset. This makes it difficult to reproduce the study. Khan et al. reported a feature count of 52, while by adhering to the publication we counted 48 features after preprocessing. With that, we were unable to replicate the feature count after preprocessing in the given paper.

P4 Feature Scaling: Furthermore, the feature encoding must not only fit the dataset but also the real-world use case. Using feature scaling only on values from the dataset instead of protocol specific minimum and maximum values may worsen the detection rate in a real deployment when encountering packets or flows not previously encountered in the training dataset.

The details and the code of the replication are publicly available [59].

4.4. Replication Results

In practice, complete flow data are hard to obtain during runtime. State-of-the-art IDSs like Snort [20] and Suricata [21] work on a per-packet basis and provide partial flow data that include information up to the current packet. However, even these partial flow data are not complete. For example, for the MQTT-IoT-IDS2020 dataset, Snort reported more than 8000 TCP-Stream events, signaling errors in the stream reassembly.

In recent work [8], flow data were aggregated on P4 programmable switches. The authors used a hash table to store the flow data. However, when the table is filled, entries are evicted. These partial flows are then exported; however, any other packets of that flow can no longer be associated with it and will be lost. Since complete flow data is an unrealistic assumption in practical deployment, we primarily focused on the models based on the packet-based data.

Table 4 shows the results of the replicated model. We replicated the model adhering strictly to the information given in the publication of Khan et al. The replicated model shows a slightly worse accuracy and precision when compared to the results reported by Khan et al. Since the original publication only mentioned that the feature `ip_len` is scaled using feature scaling in Python, multiple features like `ttl`, `mqtt_messagelength`, `mqtt_messagetype`, and `ports` were left unscaled. Therefore, in the next step, we trained a model with features that were all scaled using min–max scaling, with the protocol specific limits. The results can be seen in the row "with adopted scaling" in Table 4. The results show a significant uplift compared to the results of Khan et al. and the replicated model.

Table 4. Mean of 5-fold cross-validation results for multiclass classification.

Model	Accuracy	Precision	Recall	F_1 Score
Khan et al. [5]	90.798	89.4118	81.640	85.346
Replicated model	82.04	16.41	20.00	18.03
With adopted scaling	99.69	98.67	99.72	99.14
Without IP and timestamps	93.04	97.54	92.10	83.91

The model, however, still includes the features IP address and timestamp. In the MQTT-IoT-IDS2020 dataset, all actors have fixed addresses, as can be seen in Figure 1. Furthermore, all attacks were recorded at disjoint time intervals. Since, in practice, attackers will not have a fixed address nor adhere to fixed time windows, we also removed these features. The results can be seen in the last row of Table 4. The low standard deviation, as shown in Table 5, shows the small variance between the folds.

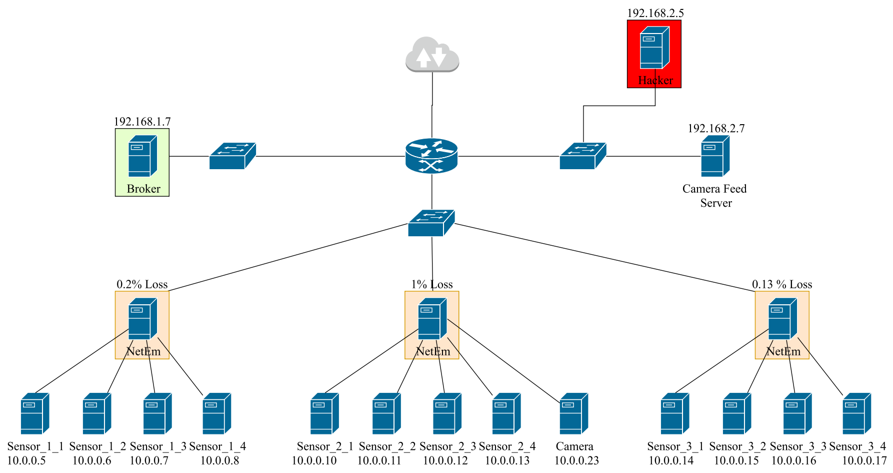

Figure 1. Architecture of the MQTT-IoT-IDS2020 dataset [3].

Table 5. Standard deviation of 5-fold cross-validation results for multiclass classification.

Model	Accuracy	Precision	Recall	F_1 Score
Khan et al. [5]	2.498	1.115	0.403	0.620
Replicated model	0.00	0.00	0.00	0.00
With adopted scaling	0.58	2.32	0.39	1.31
Without IP and timestamps	0.61	0.77	1.22	1.44

The results still do not match the reported numbers of Khan et al. and are slightly better. This shows that we were not able to replicate the results of Khan et al. exactly, but achieved a trustworthy substitute as a basis for further analysis and comparison in practice.

5. Evaluation

To validate these results in practice, we experimented with known traffic (a new recording of the MQTT-IoT-IDS2020 traffic in a similar environment), similar traffic (slightly modified network scans), and unknown traffic (sensor update, zero-day attacks). Finally, we compared the trained model against the signature-based IDS Snort.

5.1. First Experiment: Replication of Dataset

First, we analyzed the models on newly recorded data. For this, we rebuilt the architecture of the MQTT-IoT-IDS2020 dataset using the CORE network emulator [61]. The recorded traffic resembled the original dataset as close as possible and, therefore, similar results were expected. The same tools as mentioned by [3] were used. For brute-force attacks, the `rockyou` word list was used. The sensors were deployed as docker containers using the MYNO [62] sensor emulator. The camera streamed lecture recordings using `ffmpeg`. The dockerfiles and the resulting pcaps and the configuration can be found here: [59].

For further validation, we trained the two best performing models, "adopted scaling" and "without IP and timestamps", on the entire MQTT-IoT-IDS2020 dataset. These are our *final models*. We validated them against the new recorded dataset, called MQTT-IoT-IDS2020-UP. The results of the models can be seen in rows two and four in Table 6. The results are compared here to the results that the best performing model of the 5-fold cross-validation set achieved on its 20% validation set of the MQTT-IoT-IDS2020 dataset (in rows one and three).

Table 6. Comparison of the results for multiclass classification for MQTT-IoT-IDS2020 and the new recorded MQTT-IoT-IDS2020-UP dataset.

Model	Dataset	Accuracy	Precision	Recall	F_1 Score
Adopted scaling	MQTT-IoT-IDS2020	99.99	99.99	99.99	99.99
	MQTT-IoT-IDS2020-UP	98.44	96.38	70.65	74.34
Without IP and timestamps	MQTT-IoT-IDS2020	93.72	98.54	83.72	85.78
	MQTT-IoT-IDS2020-UP	54.52	74.28	67.90	60.17

For the "adopted scaling" model, a significant decrease from 99.99% to 70.65% in recall was observed. This is due to the fact that the model misclassified the aggressive scan and part of the UDP scan traffic as MQTT brute-force, as can be seen in Figure A1 in Appendix A. The other model trained without IP addresses and timestamps misclassified 80% of the MQTT brute-force traffic as normal traffic. The model also mispredicted parts of the scans, as can be seen in Figure A2 in the Appendix A.

The "adopted scaling" model performance deteriorated significantly less than the model "without IP and timestamps". This is due to the fact that the attacker still used the same IP address as in the original training dataset, as could be shown in the following experiment: Table 7 shows the results of changed IP addresses in the MQTT-IoT-IDS2020-UP dataset. The first address is the attacker address of the original dataset, the second address is an unused address in the same subnet, and the last address is a completely random, unseen address. The prediction results for the last IP address can be seen in Figure A3 in Appendix A, showing that the network classified almost all packets as benign traffic. This shows that the adopted scaling model has predominantly learned IP addresses, making it completely unfit for practical use.

Table 7. Evaluation of model "adopted scaling" on the MQTT-IoT-IDS2020-UP dataset with different attacker IP addresses.

IP Address	Accuracy	Precision	Recall	F_1 Score
192.168.2.5	98.44	96.38	70.65	74.34
192.168.2.42	98.44	96.38	70.65	74.34
37.142.89.42	15.71	62.27	43.55	36.15

The model trained without IP addresses and timestamps in Table 6, however, also showed a significant drop in overall performance and only achieved an accuracy of 54.52%. Since the same attacks and the same network topology were used for the MQTT-IoT-IDS2020-UP dataset, the significant performance drop of both models was unexpected.

Table A2 in Appendix B summarizes the binary classification results for the models. The results for the model "adopted scaling" are still very good, since it benefits from the learned IP addresses. The results for the model without IP addresses and timestamps are, again, bad. This is astonishing, since the model detects the aggressive scan, UDP scan, and Sparta mostly correctly as attacks and only misclassifies most packets of the MQTT brute-force attack as benign. The reason lies in the class imbalance of the MQTT-IoT-IDS2020-UP dataset. Since the MQTT brute-force attack class dominates the MQTT-IoT-IDS2020-UP dataset (56.20%), the recall drops to 49%.

5.2. Second Experiment: New Scan Variants

In this experiment, the neural network models were tested against slightly modified network scans. Therefore, two additional port scans were recorded.

The MQTT-IoT-IDS2020 dataset used the tool `nmap` for a UDP and an "aggressive" TCP SYN Scan. The TCP SYN Scan scans the 1000 most common ports. The "aggressive" flag enables OS detection, version detection, script scanning, and traceroute. The new scans were also recorded using `nmap`'s TCP SYN Scan. The first modification "Full Port Scan" was used to scan not only the 1000 most common ports, but all 65,535 ports. The second modification "Sensor" was used to scan a sensor instead of the MQTT broker.

The models are now evaluated for those new scans and compared against their detection performance on only the scan attacks from the MQTT-IoT-IDS2020 (original scan) and MQTT-IoT-IDS2020-UP (UP scan) dataset. The results are shown in Table 8. Rows one and five of Table 8 show, again, the scan results of the best model from the 5-fold cross-validation on the test portion of the MQTT-IoT-IDS2020 dataset. The other rows show the results of the final models on the scans from the new MQTT-IoT-IDS2020-UP dataset and the new scan recordings "Full Port Scan" and "Sensor".

Surprisingly, the models performed much worse on the scan of MQTT-IoT-IDS2020-UP dataset (UP scan) than on the other scans, even though the attacks were recorded with the same options as in the original dataset. It should be noted that the model "adopted scaling" misclassified the "Full Port Scan" as other types of attack but not as benign traffic, showcasing, again, that it has predominantly learned IP addresses. Astonishingly, the model performed admirably on the "Sensor" scan.

The model trained without IP addresses and timestamps misclassified, again, about 50% of the MQTT-IoT-IDS2020-UP aggressive scan as a Sparta attack, as can be seen in the second row of Figure A2. This also applies in case of the "Sensor" scan. The reason that the model has trouble detecting scans correctly might be due to the class imbalance in the training data and, thus, the low sample count of the scans.

Table 8. Results of the models "adopted scaling" and "without IP and timestamps" for scans from the MQTT-IoT-IDS2020 (original scan), MQTT-IoT-IDS2020-UP (UP scan), and the two described variants.

Model	Dataset	Accuracy	Precision	Recall	F_1 Score
Adopted scaling	Original scan	100.00	100.00	100.00	100.00
	UP scan	68.66	88.46	1.98	3.87
	Full Port Scan	3.44	0.00	0.00	0.00
	Sensor	99.79	99.66	99.56	99.61
Without IP and timestamps	Original scan	100.00	100.00	100.00	100.00
	UP scan	83.66	99.13	49.18	65.75
	Full Port Scan	86.41	99.99	85.93	92.43
	Sensor	88.46	98.40	57.39	72.50

Table A3 in Appendix B summarizes the binary classification results for both models for the scan variants. While the models misclassified some attacks, they never classified one of the scan attacks as benign traffic. Hence, the binary classification results are much better.

5.3. Third Experiment: Sensor Updates

In contrast to the previous experiment where the attack traffic was modified, this experiment modifies the benign traffic. The MQTT-IoT-IDS2020 dataset contains MQTT traffic from sensors to the broker. Each sensor sends a small quantity of sensor data in regular periods to the broker. For the *MUP* dataset, a sensor was updated using the MUP-Protocol [62], which sends the firmware using MQTT. This is different to the previously seen traffic in direction and quantity. The update contains a large payload compared to the previously sent sensor values.

The models are now evaluated on the firmware update traffic and compared against their detection performance on exclusively benign traffic from MQTT-IoT-IDS2020 and MQTT-IoT-IDS2020-UP dataset. The results in Table 9 show that the models are able to detect the update traffic with high accuracy. Both models perform even better on the "MUP" dataset than on the benign traffic of the MQTT-IoT-IDS2020-UP dataset.

Table 9. Results of the models "adopted scaling" and "without IP and timestamps" for benign traffic in both datasets and the MQTT Firmware Update (MUP).

Model	Dataset	Accuracy	Precision	Recall	F_1 Score
Adopted scaling	Original benign	100.00	100.00	100.00	100.00
	UP benign	90.05	100.00	90.05	94.76
	MUP	99.89	100.00	99.89	99.94
Without IP and timestamps	Original benign	100.00	100.00	100.00	100.00
	UP benign	97.21	100.00	97.21	98.59
	MUP	99.71	100.00	99.71	99.86

5.4. Fourth Experiment: Zero-Day Attacks

Intrusion detection with deep learning models on zero-day attacks is a highly researched topic [63–66]. Multiple papers [2,64–68] claim that the advantage of machine-learning-based intrusion detection systems is the ability to detect zero-day attacks. In the next experiment, we test this claim in the context of multiclass classification. The models are tested against an attack on which they have not been trained. Since the feed-forward model has no dedicated output for this attack, all outputs apart from the benign label will be interpreted as true positives. The attack used in this experiment is the DoS-New-IPv6 attack out of the THC-toolkit [69]. This attack prevents a sensor from configuring an IPv6 address by claiming it is already in use by another party. This attack exploits the missing authentication in the neighbor discovery protocol of IPv6.

Table 10 shows that both models detect all attack packets as benign traffic. This is, however, not surprising, since the model was not trained on this kind of attack or any similar to it. During our experiments, we retrained our models multiple times. While we observed little to no variance (std-dev: 0.0046 for recall) in the performance on the MQTT-IoT-IDS2020 and MQTT-IoT-IDS2020-UP datasets, we noticed a very high variance on the DoS-New-IPv6 attack: Out of five runs, three models correctly classified DoS-New-IPv6 as an attack, while two models classified the attack as benign. This shows that unseen traffic is not automatically and consistently classified as an attack.

Table 10. Results of the models "adopted scaling" and "without IP and timestamps" for a previously unseen DoS attack.

Model	Attack	Accuracy	Precision	Recall	F_1 Score
Adopted scaling	DoS New IPv6	81.45	0.00	0.00	0.00
Without IP and timestamps	DoS New IPv6	89.02	0.00	0.00	0.00

5.5. Comparison against Snort as Baseline

To evaluate the benefits of the trained model, it needs to be compared to a proper baseline. A realistic baseline and state-of-the-art is a signature-based IDS that is configured to detect the same attacks as the ML-based IDS. In the sections above, we showed that the model "adopted scaling" primarily learned IP addresses and is therefore excluded from further evaluation.

5.5.1. Snort Configuration

One of the most commonly used signature-based intrusion detection systems is Snort [20]. For our evaluation, the latest version of Snort (snort3-3.1.66) was used. Before running Snort, variables like `HOME_NET` and `EXTERNAL_NET`, which are used by Snort rules as placeholders, need to be configured. For the detection of attacks like a port scan, an IDS needs to keep some state. In the case of Snort, this is performed by so-called *inspectors*. Therefore, thresholds for inspectors like the `port_scan` inspector need to be configured and enabled. If the administrator is interested in all inspector events, this can be enabled globally or on a per-event basis. Snort can also be easily extended via plugins with other inspectors and rule options. The neighbor discovery protocol inspector `indp` [70], for example, detects attacks on the IPv6 neighbor discovery protocol [71].

There are multiple rule collections available for Snort. In addition to the openly available *community rule set*, there are also commercial rule sets available, like the talos rule set, that is available for free after 30 days. To be alerted to too many SSH authentication attempts, a custom rule needs to be written, which requires a threshold to be configured per environment. An example implementation of this rule can be seen in Listing 1, which comes from the Snort 3 rule writing guide, Section "detection_filters" [20].

Listing 1. Snort rule to detect too many SSH authentication attempts.

```
alert tcp any any -> $HOME_NET 22
(
  flow: established,to_server;
  content: "SSH", nocase, offset 0, depth 4;
  detection_filter: track by_src, count 30, seconds 60;
  msg: "ssh bruteforce"; sid:1000002
)
```

This rule creates an alert when more than 30 SSH connections are established from the same source IP to the local network within 60 s. It does not count the actual failed SSH authentication attempts, since this information is TLS-encrypted. Since most SSH configurations only allow three authentication attempts per connection, this is a good substitute.

Since MQTT servers like mosquitto reply to a failed authentication attempt with a Connect ACK packet (type 20) with a `reason code` of 5, a rule as seen in Listing 2 that can easily be created to detect too many MQTT authentication attempts. The limits chosen here were chosen arbitrarily and not tuned to the dataset.

Listing 2. Snort rule to detect too many MQTT authentication attempts.

```
alert tcp any 1883 -> any any
(
  flow: established,to_client;
  content: "|20|", offset 0, depth 1;
  content: "|05|", offset 3, depth 1;
  detection_filter:track by_src, count 30, seconds 60;
  msg: "MQTT Bruteforce"; sid:1000001;
)
```

The Snort configuration for the evaluation includes the previously mentioned rules and the default medium port scan profile. The resulting Snort installation and complete configuration can be found here [59].

5.5.2. Results for MQTT-IoT-IDS2020-UP

The results of our Snort configuration on the MQTT-IoT-IDS2020-UP dataset can be seen in Table 11. Snort was not used on the original dataset because the original MQTT-IoT-IDS2020 dataset does not provide pcap files prefiltered by attack classes. All pcap files of the original dataset contain benign traffic. Since the MQTT-IoT-IDS2020-UP dataset was created with the same tools, the attack signatures did not change. Therefore, the Snort results for the MQTT-IoT-IDS2020 should be similar to those of MQTT-IoT-IDS2020-UP.

Table 11. Comparison of results for Snort and the model trained without timestamps and IP addresses on the MQTT-IoT-IDS2020-UP dataset.

Model	Data	Accuracy	Precision	Recall	F_1 Score
Snort	Normal	99.81	99.81	100.00	99.90
	Aggressive Scan	96.81	100.00	96.81	98.38
	UDP Scan	83.77	100.00	83.77	91.17
	MQTT Brute-force	99.99	100.00	99.99	99.99
	Sparta	99.91	100.00	99.91	99.96
Without IP and timestamps	Normal	19.21	19.32	97.21	32.23
	Aggressive Scan	37.71	61.77	49.18	54.76
	UDP Scan	68.60	91.33	73.38	81.38
	MQTT Brute-force	19.75	100.00	19.75	32.98
	Sparta	98.96	98.96	100.00	99.48

For all four attacks, Snort achieved a precision of 100%, showing that there were no false positives. This is, however, not surprising since only the rules relevant to the discussed attacks, and not the community rule set, were enabled. The recall of the attacks, however, varied and was below 100%. This is due to the fact that a brute-force attack is only detected after the count threshold specified in the detection filter is reached. The same principle applies to the port scan. However, for the UDP scan, the recall was significantly lower, showing that Snort did not classify all UDP scan packets as attack traffic even though the `alert_all` option was enabled. Because of the port scan, the victim replied with `ICMP unreachable` packets, which in turn were classified as possible ICMP scan. This is correct and a result of the scan, but can be misleading to novice users.

For all classes included in the MQTT-IoT-IDS2020-UP dataset, Snort outperformed the neural network model.

5.5.3. Results for Traffic Variations

The results for Snort for the additional experiments with traffic variations from Sections 5.2–5.4 can be seen in Table 12. Except for the new Sparta variations, Snort outperformed the model. These variations are somehow artificial since the configuration of the attacked SSH server needed to be changed. In "Sparta fast", unlimited login attempts are allowed, while in "Sparta slow", only 10 login attempts are allowed from the same IP address. Therefore, Snort's threshold for too many connections was not reached quickly enough, resulting in false negatives.

Table 12. Comparison of results for Snort and the model trained without timestamps and IP addresses on the datasets with traffic variations.

Model	Data	Accuracy	Precision	Recall	F_1 Score
Snort	Sensor	99.62	100.00	98.53	99.26
	Full Port Scan	99.98	100.00	99.98	99.99
	Sparta Slow	96.67	100.00	95.33	97.61
	Sparta Fast	96.26	100.00	94.88	97.37
	DoS New IPv6	100.00	100.00	100.00	100.00
	MUP	100.00	100.00	100.00	100.00
Without IP and timestamps	Sensor	88.46	98.40	57.39	72.50
	Full Port Scan	86.41	99.99	85.93	92.43
	Sparta Slow	99.77	98.50	100.00	99.24
	Sparta Fast	99.98	99.98	100.00	99.99
	Dos New IPv6	89.02	0.00	0.00	0.00
	MUP	99.71	100.00	99.71	99.86

5.6. Discussion

Even though the model achieved an accuracy, precision, and recall of over 90% on the validation split, our experiments showed that the model performance deteriorated significantly when tested on similar but unseen data (see Section 5.2). This highlights two current problems of machine-learning-based intrusion detection. First, the used datasets are too small and homogeneous. There is little to no variance in the benign traffic and the attacks. Therefore, once there is some variation in new data, the model is not able to generalize. Furthermore, even if more variance is added to the dataset, the Dimpled Manifold Model [72] indicates that the model will most likely create a "dimple" around the new variation in the decision boundary, instead of generalizing. Second, the differences between benign traffic and attacks are so significant that even arbitrarily chosen (fixed) thresholds can be used to differentiate between attacks and benign traffic. For example, in the MQTT-IoT-IDS2020 dataset, the average MQTT authentication rate of benign traffic is 11.94 per second with a standard deviation of 0.47, while for the MQTT brute-force attack, the average authentication rate is 660.07 per second with a standard deviation of 298.22. Machine learning models are not needed to detect these easily distinguishable events. The area of application should be scenarios where static thresholds and signature-based IDSs struggle. This underlines, again, the importance of realistic datasets and the comparison against a baseline that was configured and tuned to the environment. For example, it should be no surprise that a machine learning model can outperform a signature-based IDS on attacks that are distinguishable on flow data if the signature-based IDS only includes the emerging threats dataset, but no configured flow limits.

Since the creation of realistic datasets is another challenging problem, it is important not only to trust the results of 5-fold cross-validation but also to validate the final model on similar data and slight variations of it.

Furthermore, there are multiple pitfalls when splitting the dataset into folds. Data leakage (temporal inconsistencies, data snooping) is, according to [26], one of them. For network data, leaks can occur if packets of the same attack or of a benign flow are split into different folds. Hence, the model would be trained on packets from an attack or from a benign flow it is then tested on. This would bias the performance results.

5.7. Discussion of Performance Metrics

As in the original paper [5], we provided the performance measures accuracy, precision, recall, and F1 score based on each packet or flow analyzed. These per-packet or per-flow metrics make sense in other domains of machine learning. For example, in image recognition it is interesting how many of the provided images were classified correctly. In intrusion detection, however, this is not the correct metric. Cybersecurity analysts are not interested in diagnostics on a packet/flow layer but on an attack layer. A scan or brute-force attack consists of thousands of packets and flows. The performance of an intrusion detection solution, therefore, should be evaluated on whether it detects each attack instance and not whether it classifies each packet correctly. This difference seems to be insignificant at first; however, it has a significant impact on the precision metric. Imagine a dataset containing 10^6 packets, which are part of three distinct brute-force attacks. If a model detects 999,000 packets correctly with only 1000 false positives, it has an impressive precision of 99.9%. However, since the 10^6 packets are only part of three distinct brute-force attacks, this changes the number of true positives down to three. The resulting precision is $\frac{3}{3+1000} = 0.29\%$. The 1000 false positives are unchanged from this conversion since each false positive still produces an alert. The introduction of a simple threshold such that an alert is triggered only when a certain number of true positives are detected does not constitute a fix, because the number of flows in an attack varies significantly depending on the attack type.

All analyzed models detected at least one packet of each attack in the datasets, not counting the zero-day experiment. However, the model "without IP and timestamps" alone created 930,723 false positive alerts, making it unusable in real-world scenarios.

5.8. Usability

The usability of intrusion detection systems is an important factor for practical use. Usability is multifaceted and can be divided into multiple subcategories. Here, we will discuss two factors: the easy integration into existing environments and the quality and quantity of alerts. Most signature-based IDSs provide multiple input and output formats. Most publications of machine-learning-based IDS analyze already preprocessed data. For integration into live traffic, the processing, including flow analysis of network traffic, needs to be conducted in real time. As mentioned in Section 4.4, the flow analysis in real time may lead to a significant decrease in flow data quality if compared to the preprocessed data. Since most SIEM systems accept a variety of input formats, the chosen model output format is generally not a problem.

The quality of alerts is another important factor. For signature-based intrusion detection systems, the quality of alerts is directly related to the quality of the rule set used. One of the approaches for IDS deployment is to start with huge rule sets and then remove rules that lead to false positives.

For machine-learning-based IDSs, the quality of alerts is directly related to the quality of the dataset. Creating appropriate datasets is a hard problem. The effort to label real network traffic is usually too high. Therefore, attacks are most commonly reproduced in lab environments. The transferability of those models to real-world data is not guaranteed and may lead to a significant decrease in performance, as we showed in this paper. Furthermore, in a signature-based IDS, the reason for an alert is directly visible in the rule set, while for deep learning models, explainability is still an open problem.

5.9. Analysis with Flow Data

As mentioned in Section 4.4, we focused our experiments on packet data. To ensure that the transferability problem of the models to real-world data is also present in models that were trained on flow data, we also replicated the uni-flow and bi-flow models. The results for the 5-fold cross-validation on the original MQTT-IoT-IDS2020 dataset and the results of the final model on the MQTT-IoT-IDS2020-UP dataset can be seen in Table 13.

Table 13. Results for the replicated models trained on uni-flow and bi-flow data from the MQTT-IoT-IDS2020 dataset compared with the results reported by Khan et al. The replicated models were also evaluated on the MQTT-IoT-IDS2020-UP dataset.

Model/Dataset	Characteristic	Accuracy	Precision	Recall	F_1 Score
Khan et al. [5]	uni-flow	97.08	94.76	86.44	90.61
	Bi-flow	98.13	95.11	86.72	90.72
MQTT-IoT-IDS2020	uni-flow	99.82	99.94	99.35	99.63
	Bi-flow	99.76	99.92	99.25	99.56
MQTT-IoT-IDS2020-UP	uni-flow	7.77	58.99	58.00	39.72
	Bi-flow	45.32	56.05	66.82	50.91

While our replicated models surpassed the results of the original paper, the models failed to detect the same attacks on the MQTT-IoT-IDS2020-UP dataset. The flow models correctly identified scan attacks, since these are easy to detect in flow data, but had problems with the brute-force attacks, as can be seen in Figure A4 in Appendix A.

5.10. Limitations and Future Work

The missing baseline comparison against a signature-based IDS is a systematic problem in the field of neural-network-based intrusion detection. We replicated a recent paper and showed that it is outperformed by the signature-based IDS Snort. Further, the problem of generalization to similar data and the limitations of the dataset are shown in this example. However, we suspect that these problems are not limited to the selected model and dataset.

Our work shows that comparison with a proper baseline is necessary when proposing new intrusion detection models.

Our comparison focused on detection performance, but there are also other properties that are important for an IDS. First, efficient implementation within an IDS workflow is a crucial aspect. To date, we have rarely seen implementations of a complete system. More frequently, we only see implementations of parts like flow analysis, training, and inference. The experiments of Khan et al. showed that flow-based detection is better than packet-based, but for this, agglomeration of these statistics is needed during runtime. In [8], it was shown that for 100 GBit networks, a P4-enabled Intel© Tofino^{TM} switch with hardware acceleration was needed to perform this metadata extraction. Hence, the runtime performance is important to judge the usability of a system.

Further, an important aspect is the explainability of an IDS. While for ML-based systems this is a heavily researched topic, the explainability of signature-based IDSs relies heavily on the rule set they employ. These rule sets generally comprise hundreds of rules, which often leads to a high false positive rate, resulting in the *alert fatigue* problem. Furthermore, the messages in the rules rarely provide enough information on what the author intended to detect. Hence, there is room for improvement on both sides.

6. Conclusions

We presented experiments with a deep neural network model for intrusion detection. While the model achieved high accuracy, precision, and recall using 5-fold cross-validation, its performance on similar datasets was significantly worse. The most likely reason is the low variance in the synthetically generated training datasets. Furthermore, it was not possible to confirm the long-asserted claim that machine-learning-based intrusion detection systems are superior to signature-based IDSs. However, it is worth mentioning that even though the models' recall was not high, they mostly only mispredicted an attack as a different attack and not as benign traffic. In case of binary classification, those models would have performed significantly better. We conclude that a suitable area of application for ML-based intrusion detection is most probably anomaly detection.

The importance of a good baseline is also highlighted by our experiments. New machine learning models should always be compared to classical signature-based IDS to show that the problem is not trivially solvable with a static threshold, or already solved in their rule set. For this comparison, it is important to configure the baseline with equivalent effort to realistically demonstrate where a machine learning model can outperform state-of-the-art solutions.

We also argue that performance metrics should be based on detected attacks and not on correctly classified flows/packets, since practitioners are not interested whether all packets in an attack were correctly attributed to this attack, but that the attack as a whole was detected. This is particularly important in real-world detection since every false positive will result in an alert.

Last, but not least, we want to thank all researchers who publish all of their research artifacts including datasets and source code, and we want to stress how important the FAIR principles are for easy replication of research results and trust in publications.

Author Contributions: Conceptualization, M.S., A.N. and B.S.; methodology, M.S., A.N. and B.S.; software, M.S. and A.N.; validation, M.S., A.N. and B.S.; writing—original draft preparation, M.S., A.N. and B.S.; writing—review and editing, M.S., A.N. and B.S.; visualization, A.N.; supervision, B.S. All authors have read and agreed to the published version of the manuscript.

Funding: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—Projektnummer 491466077.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The publicly available source code, datasets, and configuration files of the experiments can be found at [59].

Acknowledgments: The authors want to thank Klemens Kittan for his technical support.

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

Appendix A. Confusion Matrices

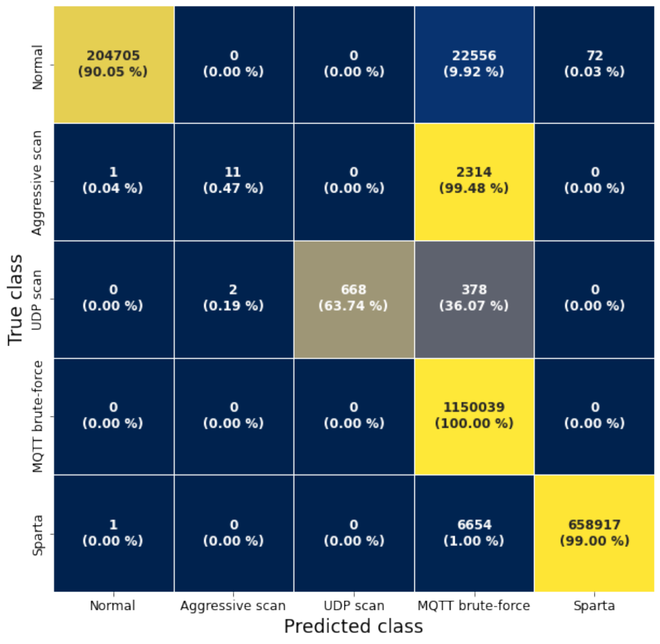

Figure A1. Confusion matrix of the results for the model "with adopted scaling" on the MQTT-IoT-IDS2020-UP dataset.

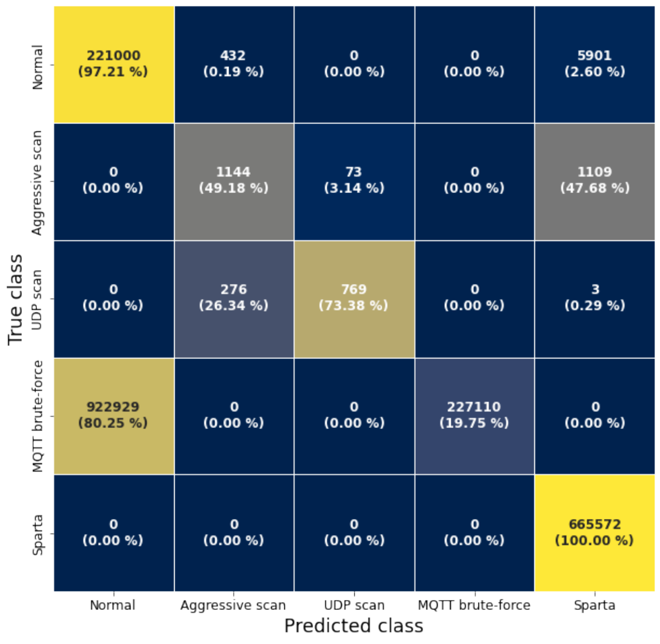

Figure A2. Confusion matrix of the results for the model "without IP and timestamps" on the MQTT-IoT-IDS2020-UP dataset.

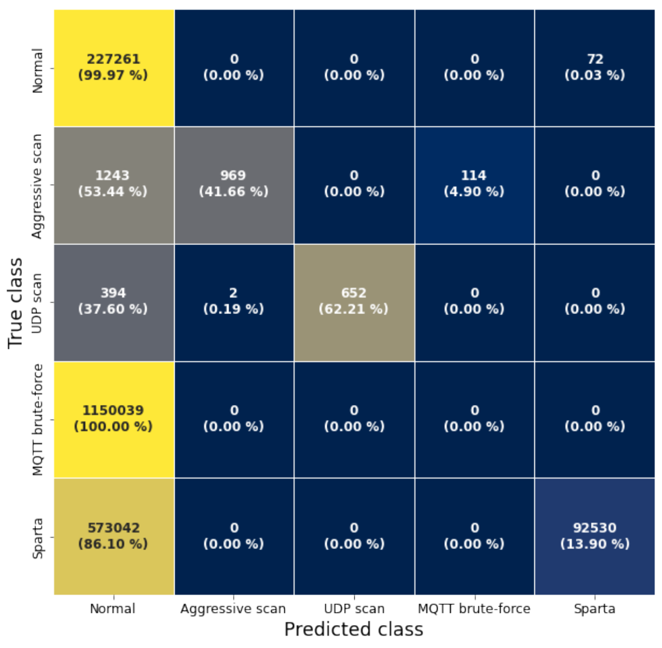

Figure A3. Results of the model “with adopted scaling” for attacks originating from a completely unseen attacker address.

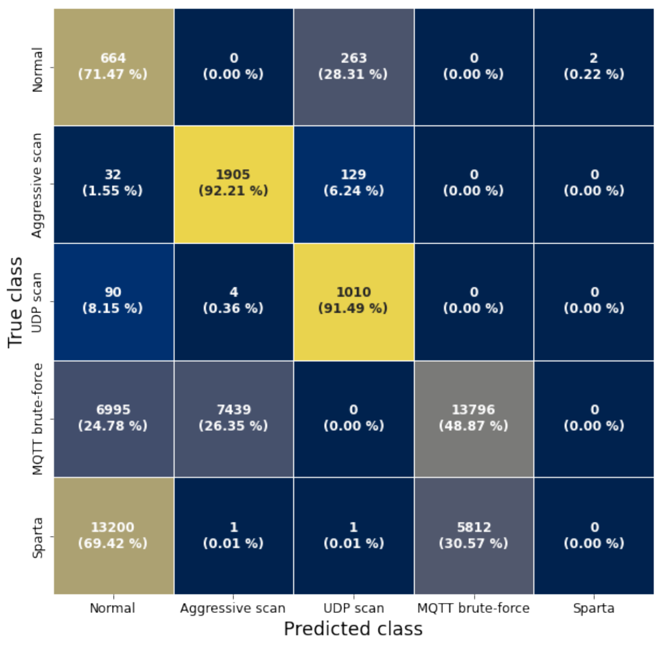

Figure A4. Confusion matrix of the uni-flow model validated on the MQTT-IoT-IDS2020-UP dataset.

Appendix B. Binary Classification

Table A1 shows that the MQTT-IoT-IDS2020-UP dataset that is used as test dataset has a similar class distribution as the original MQTT-IoT-IDS2020 dataset. For both datasets, the brute-force attacks are the most common classes. However, the Sparta class contains significantly less samples in the MQTT-IoT-IDS2020-UP dataset. This is due to the SSH server configuration limiting the number of connections significantly, as described in Section 5.2.

Table A1. Class distribution MQTT-IoT-IDS2020-UP.

Class	MQTT-IoT-IDS2020	Replicated Training Set	MQTT-IoT-IDS2020-UP
Normal	7.21%	82.04%	11.11%
Scan_A	0.13%	1.44%	0.11%
Scan_SU	0.07%	0.79%	0.05%
MQTT Brute-force	31.17%	7.83%	56.20%
Sparta	61.42%	7.90%	32.53%

When using binary classification, this class imbalance has a significant effect on the used metrics. Since for multiclass classification macroaveraging was used, the class imbalance of the test dataset had no effect, while for binary classification, the class imbalance had a significant bias towards the MQTT brute-force class. This is also reflected in the results in Table A2. The model "without IP and timestamps" has a significantly lower recall and accuracy than the earlier reported results in Table 6 using multiclass classification.

Table A2. Comparison of the results for binary classification for MQTT-IoT-IDS2020 and the new recorded MQTT-IoT-IDS2020-UP dataset.

Model	Dataset	Accuracy	Precision	Recall	F_1 Score
Adopted scaling	MQTT-IoT-IDS2020	99.99	100.00	99.99	99.99
	MQTT-IoT-IDS2020-UP	98.89	98.77	99.99	99.38
Without IP and timestamps	MQTT-IoT-IDS2020	93.73	100.00	65.07	78.84
	MQTT-IoT-IDS2020-UP	54.59	99.30	49.26	65.85

Table A3. Results for binary classification of the models "adopted scaling" and "without IP and timestamps" for scans out of the MQTT-IoT-IDS2020 (original scan), MQTT-IoT-IDS2020-UP (UP scan) and the two described variants.

Model	Dataset	Accuracy	Precision	Recall	F_1 Score
Adopted scaling	Original scan	100.00	100.00	100.00	100.00
	UP scan	99.63	99.74	99.10	99.42
	Full Port Scan	99.99	99.99	100.00	99.99
	Sensor	99.90	99.66	99.95	99.80
Without IP and timestamps	Original scan	100.00	100.00	100.00	100.00
	UP scan	99.86	99.57	100.00	99.79
	Full Port Scan	99.99	99.99	99.99	99.99
	Sensor	99.75	99.08	100.00	99.54

References

1. Alaiz-Moreton, H.; Aveleira-Mata, J.; Ondicol-Garcia, J.; Muñoz-Castañeda, A.L.; García, I.; Benavides, C. Multiclass Classification Procedure for Detecting Attacks on MQTT-IoT Protocol. *Complexity* **2019**, *2019*, 6516253. [CrossRef]
2. Ciklabakkal, E.; Donmez, A.; Erdemir, M.; Suren, E.; Yilmaz, M.K.; Angin, P. ARTEMIS: An Intrusion Detection System for MQTT Attacks in Internet of Things. In Proceedings of the 2019 38th Symposium on Reliable Distributed Systems (SRDS), Lyon, France, 1–4 October 2019; pp. 369–371.
3. Hindy, H.; Bayne, E.; Bures, M.; Atkinson, R.; Tachtatzis, C.; Bellekens, X. Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset). In *INC 2020: Selected Papers from the 12th International Networking Conference*; Lecture Notes in Networks and Systems; Ghita, B., Shiaeles, S., Eds.; Springer: Cham, Switzerland, 2021; Volume 180, pp. 73–84.
4. Ge, M.; Fu, X.; Syed, N.; Baig, Z.; Teo, G.; Robles-Kelly, A. Deep Learning-Based Intrusion Detection for IoT Networks. In Proceedings of the 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), Kyoto, Japan, 1–3 December 2019; pp. 256–265.
5. Khan, M.A.; Khan, M.A.; Jan, S.U.; Ahmad, J.; Jamal, S.S.; Shah, A.A.; Pitropakis, N.; Buchanan, W.J. A Deep Learning-Based Intrusion Detection System for MQTT Enabled IoT. *Sensors* **2021**, *21*, 7016. [CrossRef] [PubMed]

6. Mosaiyebzadeh, F.; Araujo Rodriguez, L.G.; Macêdo Batista, D.; Hirata, R. A Network Intrusion Detection System using Deep Learning against MQTT Attacks in IoT. In Proceedings of the 2021 IEEE Latin-American Conference on Communications (LATINCOM), Santo Domingo, Dominican Republic, 17–19 November 2021; pp. 1–6.
7. Ullah, I.; Ullah, A.; Sajjad, M. Towards a Hybrid Deep Learning Model for Anomalous Activities Detection in Internet of Things Networks. *IoT* **2021**, *2*, 428–448. [CrossRef]
8. Gray, N.; Dietz, K.; Seufert, M.; Hossfeld, T. High Performance Network Metadata Extraction Using P4 for ML-based Intrusion Detection Systems. In Proceedings of the 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), Paris, France, 7–10 June 2021; pp. 1–7. [CrossRef]
9. Engelen, G.; Rimmer, V.; Joosen, W. Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study. In Proceedings of the 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 27 May 2021; pp. 7–12. [CrossRef]
10. Kamaldeep; Malik, M.; Dutta, M. Feature Engineering and Machine Learning Framework for DDoS Attack Detection in the Standardized Internet of Things. *IEEE Internet Things J.* **2023**, *10*, 8658–8669. [CrossRef]
11. Cholakoska, A.; Gjoreski, H.; Rakovic, V.; Denkovski, D.; Kalendar, M.; Pfitzner, B.; Arnrich, B. Federated Learning for Network Intrusion Detection in Ambient Assisted Living Environments. *IEEE Internet Comput.* **2023**, *27*, 15–22. [CrossRef]
12. Sinha, S. State of IoT 2023: Number of Connected IoT Devices Growing 16% to 16.7 Billion Globally. Available online: https://iot-analytics.com/number-connected-iot-devices/ (accessed on 27 September 2023).
13. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and Other Botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
14. Kumari, P.; Jain, A.K. A comprehensive study of DDoS attacks over IoT network and their countermeasures. *Comput. Secur.* **2023**, *127*, 103096. [CrossRef]
15. BSI. Grundlagen: 1. IDS-Grundlagen und Aktueller Stand. Available online: https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Studien/IDS02/gr1_htm.html?nn=132646 (accessed on 19 August 2023).
16. Dini, P.; Elhanashi, A.; Begni, A.; Saponara, S.; Zheng, Q.; Gasmi, K. Overview on Intrusion Detection Systems Design Exploiting Machine Learning for Networking Cybersecurity. *Appl. Sci.* **2023**, *13*, 7507. [CrossRef]
17. Mliki, H.; Kaceam, A.H.; Chaari, L. A Comprehensive Survey on Intrusion Detection based Machine Learning for IoT Networks. *EAI Endorsed Trans. Secur. Saf.* **2021**, *8*, e3. [CrossRef]
18. IBM. What Are Neural Networks? Available online: https://www.ibm.com/topics/neural-networks (accessed on 20 November 2023).
19. Arp, D.; Quiring, E.; Pendlebury, F.; Warnecke, A.; Pierazzi, F.; Wressnegger, C.; Cavallaro, L.; Rieck, K. Dos and Don'ts of Machine Learning in Computer Security. In Proceedings of the USENIX Security Symposium, Boston, MA, USA, 12–14 August 2020.
20. Snort. Snort 3 Rule Writing Guide. Available online: https://docs.snort.org/ (accessed on 19 November 2023).
21. The Open Information Security Foundation. Suricata. Available online: https://suricata.io/ (accessed on 19 November 2023).
22. Project, T.Z. The Zeek Network Security Monitor. Available online: https://zeek.org/ (accessed on 19 November 2023).
23. Threats, E. Emerging Threats Ruleset. Available online: https://community.emergingthreats.net/ (accessed on 20 November 2023).
24. Cahyo, A.N.; Kartika Sari, A.; Riasetiawan, M. Comparison of Hybrid Intrusion Detection System. In Proceedings of the 2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, Indonesia, 6–8 October 2020; pp. 92–97. [CrossRef]
25. Leevy, J.; Khoshgoftaar, T. A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. *J. Big Data* **2020**, *7*, 104. [CrossRef]
26. Ceschin, F.; Botacin, M.; Bifet, A.; Pfahringer, B.; Oliveira, L.S.; Gomes, H.M.; Grégio, A. Machine Learning (In) Security: A Stream of Problems. *Digit. Threat.* **2023**. [CrossRef]
27. Dambra, S.; Han, Y.; Aonzo, S.; Kotzias, P.; Vitale, A.; Caballero, J.; Balzarotti, D.; Bilge, L. Decoding the Secrets of Machine Learning in Windows Malware Classification: A Deep Dive into Datasets, Feature Extraction, and Model Performance. *arXiv* **2023**, arXiv:2307.14657.
28. Venturi, A.; Zanasi, C.; Marchetti, M.; Colajanni, M. Robustness Evaluation of Network Intrusion Detection Systems based on Sequential Machine Learning. In Proceedings of the 2022 IEEE 21st International Symposium on Network Computing and Applications (NCA), Boston, MA, USA, 14–16 December 2022; Volume 21, pp. 235–242. [CrossRef]
29. Aldhaheri, S.; Alhuzali, A. SGAN-IDS: Self-Attention-Based Generative Adversarial Network against Intrusion Detection Systems. *Sensors* **2023**, *23*, 7796. [CrossRef]
30. Zola, F.; Bruse, J.L.; Galar, M. Temporal Analysis of Distribution Shifts in Malware Classification for Digital Forensics. In Proceedings of the 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Delft, The Netherlands, 3–7 July 2023; pp. 439–450. [CrossRef]
31. Apruzzese, G.; Andreolini, M.; Marchetti, M.; Venturi, A.; Colajanni, M. Deep Reinforcement Adversarial Learning Against Botnet Evasion Attacks. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 1975–1987. [CrossRef]
32. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Portugal, 22–24 January 2018; Mori, P., Furnell, S., Camp, O., Eds.; SciTePress: Setubal, Portugal, 2018; pp. 108–116. [CrossRef]
33. Tidjon, L.N.; Khomh, F. Threat Assessment in Machine Learning based Systems. *arXiv* **2022**, arXiv:2207.00091.

34. OWASP. OWASP Machine Learning Security Top Ten. Available online: https://owasp.org/www-project-machine-learning-security-top-10/ (accessed on 21 November 2023).
35. OWASP. OWASP AI Security and Privacy Guide. Available online: https://owasp.org/www-project-ai-security-and-privacy-guide/ (accessed on 21 November 2023).
36. Vaccari, I.; Chiola, G.; Aiello, M.; Mongelli, M.; Cambiaso, E. MQTTset, a New Dataset for Machine Learning Techniques on MQTT. *Sensors* **2020**, *20*, 6578. [CrossRef] [PubMed]
37. Aveleira, J. MQTT_UAD: MQTT Under Attack Dataset. A Public Dataset for the Detection of Attacks in IoT Networks Using MQTT. Available online: https://joseaveleira.es/dataset/ (accessed on 20 November 2023).
38. Hindy, H.; Tachtatzis, C.; Atkinson, R.; Bayne, E.; Bellekens, X. MQTT-IoT-IDS2020: MQTT Internet of Things Intrusion Detection Dataset. *IEEE Dataport* **2020**. [CrossRef]
39. 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]
40. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. The Bot-IoT Dataset. Available online: https://research.unsw.edu.au/projects/bot-iot-dataset (accessed on 5 October 2023).
41. Ullah, I.; Mahmoud, Q.H. A Technique for Generating a Botnet Dataset for Anomalous Activity Detection in IoT Networks. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 134–140. [CrossRef]
42. Ullah, I.; Mahmoud, Q.H. IoT Network Intrusion Datasets. Available online: https://sites.google.com/view/iotdataset1 (accessed on 18 December 2023).
43. Garcia, S.; Parmisano, A.; Erquiaga, M.J. IoT-23: A Labeled Dataset with Malicious and Benign IoT Network Traffic (Version 1.0.0). 2020. Available online: https://www.stratosphereips.org/datasets-iot23 (accessed on 18 December 2023).
44. Vaccari, I.; Chiola, G.; Aiello, M.; Mongelli, M.; Cambiaso, E. MQTTset. Available online: https://www.kaggle.com/datasets/cnrieiit/mqttset (accessed on 5 January 2024).
45. McHugh, J. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.* **2000**, *3*, 262–294. [CrossRef]
46. Nehinbe, J.O. A critical evaluation of datasets for investigating IDSs and IPSs researches. In Proceedings of the 2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS), London, UK, 1–2 September 2011; pp. 92–97. [CrossRef]
47. 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]
48. Thakkar, A.; Lohiya, R. A Review of the Advancement in Intrusion Detection Datasets. *Procedia Comput. Sci.* **2020**, *167*, 636–645. [CrossRef]
49. Rosay, A.; Carlier, F.; Leroux, P. MLP4NIDS: An Efficient MLP-Based Network Intrusion Detection for CICIDS2017 Dataset. In *MLN 2019: Machine Learning for Networking*; Boumerdassi, S., Renault, É., Mühlethaler, P., Eds.; Springer: Cham, Switzerland, 2020; pp. 240–254.
50. Kurniabudi; Stiawan, D.; Darmawijoyo; Bin Idris, M.Y.; Bamhdi, A.M.; Budiarto, R. CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection. *IEEE Access* **2020**, *8*, 132911–132921. [CrossRef]
51. Holland, J.; Schmitt, P.; Feamster, N.; Mittal, P. New Directions in Automated Traffic Analysis. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security—CCS'21, New York, NY, USA, 9–13 November 2021; pp. 3366–3383. [CrossRef]
52. Panigrahi, R.; Borah, S. A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. *Int. J. Eng. Technol.* **2018**, *7*, 479–482.
53. OASIS Open. *MQTT Version 5.0*; OASIS Standard; OASIS Open: Woburn, MA, USA, 2019.
54. Neto, E.C.P.; Dadkhah, S.; Ferreira, R.; Zohourian, A.; Lu, R.; Ghorbani, A.A. CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors* **2023**, *23*, 5941. [CrossRef]
55. Palsson, K. Malaria Toolkit. 2019. Available online: https://github.com/etactica/mqtt-malaria (accessed on 28 January 2024).
56. Open JS Foundation & Contributors. Node-RED. Available online: https://nodered.org (accessed on 5 January 2024).
57. Jacobs, A.S.; Beltiukov, R.; Willinger, W.; Ferreira, R.A.; Gupta, A.; Granville, L.Z. AI/ML for Network Security: The Emperor Has No Clothes. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security—CCS'22, New York, NY, USA, 7–11 November 2022; pp. 1537–1551. [CrossRef]
58. Wilkinson, M.D.; Dumontier, M.; Aalbersberg, I.J.; Appleton, G.; Axton, M.; Baak, A.; Blomberg, N.; Boiten, J.W.; da Silva Santos, L.B.; Bourne, P.E.; et al. The FAIR Guiding Principles for scientific data management and stewardship. *J. Sci. Data* **2016**, *3*, 160018. [CrossRef]
59. Schrötter, M. Replication DLIDS Khan. Available online: https://gitup.uni-potsdam.de/maxschro/replication-dlids-khan (accessed on 5 January 2024).
60. Raj Kumar, P.A.; Selvakumar, S. Distributed denial of service attack detection using an ensemble of neural classifier. *Comput. Commun.* **2011**, *34*, 1328–1341. [CrossRef]
61. Ahrenholz, J.; Danilov, C.; Henderson, T.R.; Kim, J.H. CORE: A real-time network emulator. In Proceedings of the MILCOM 2008—2008 IEEE Military Communications Conference, San Diego, CA, USA, 16–19 November 2008; pp. 1–7. [CrossRef]

62. Sahlmann, K.; Clemens, V.; Nowak, M.; Schnor, B. MUP: Simplifying Secure Over-The-Air Update with MQTT for Constrained IoT Devices. *J. Sens.* **2020**, *21*, 10. [CrossRef]
63. Ahmad, R.; Alsmadi, I.; Alhamdani, W.; Tawalbeh, L. Zero-day attack detection: A systematic literature review. *Artif. Intell. Rev.* **2023**, *56*, 10733–10811. [CrossRef]
64. Gharib, M.; Mohammadi, B.; Dastgerdi, S.H.; Sabokrou, M. AutoIDS: Auto-encoder Based Method for Intrusion Detection System. *arXiv* **2019**, arXiv:1911.03306.
65. Hindy, H.; Atkinson, R.; Tachtatzis, C.; Colin, J.N.; Bayne, E.; Bellekens, X. Utilising Deep Learning Techniques for Effective Zero-Day Attack Detection. *Electronics* **2020**, *9*, 1684. [CrossRef]
66. Soltani, M.; Ousat, B.; Jafari Siavoshani, M.; Jahangir, A.H. An adaptable deep learning-based intrusion detection system to zero-day attacks. *J. Inf. Secur. Appl.* **2023**, *76*, 103516. [CrossRef]
67. Ouiazzane, S.; Addou, M.; Barramou, F. A Suricata and Machine Learning Based Hybrid Network Intrusion Detection System. In *Advances in Information, Communication and Cybersecurity*; Maleh, Y., Alazab, M., Gherabi, N., Tawalbeh, L., Abd El-Latif, A.A., Eds.; Springer: Cham, Switzerland, 2022; pp. 474–485.
68. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
69. Heuse, M. THC IPv6 Attack Tool Kit. 2005. Available online: https://github.com/vanhauser-thc/thc-ipv6 (accessed on 5 January 2024).
70. Schroetter, M.; Scheffler, T.; Schnor, B. Inspector for the Neighbour Discovery Protocol. Available online: https://redmine.cs.uni-potsdam.de/projects/snort3-ipv6-plugin/files (accessed on 26 January 2024).
71. Schroetter, M.; Scheffler, T.; Schnor, B. Evaluation of Intrusion Detection Systems in IPv6 Networks. In Proceedings of the International Conference on Security and Cryptography (SECRYPT 2019), Prague, Czech Republic, 26–28 July 2019.
72. Shamir, A.; Melamed, O.; BenShmuel, O. The Dimpled Manifold Model of Adversarial Examples in Machine Learning. *arXiv* **2021**, arXiv:2106.10151.