Journal of Cybersecurity and Privacy

MDPI

*Article*

# Launching Adversarial Attacks against Network Intrusion Detection Systems for IoT

**Pavlos Papadopoulos [1,*], Oliver Thornewill von Essen [1], Nikolaos Pitropakis [1,*], Christos Chrysoulas [1], Alexios Mylonas [2] and William J. Buchanan [1]**

[1] Blockpass ID Lab, School of Computing Edinburgh Napier University, Edinburgh EH10 5DT, UK; 40210534@live.napier.ac.uk (O.T.v.E.); C.Chrysoulas@napier.ac.uk (C.C.); B.Buchanan@napier.ac.uk (W.J.B.)

[2] Department of Computer Science, University of Hertfordshire, Hatfield AL10 9AB, UK; a.mylonas@herts.ac.uk

* Correspondence: pavlos.papadopoulos@napier.ac.uk (P.P.); n.pitropakis@napier.ac.uk (N.P.)

**Abstract:** As the internet continues to be populated with new devices and emerging technologies, the attack surface grows exponentially. Technology is shifting towards a profit-driven Internet of Things market where security is an afterthought. Traditional defending approaches are no longer sufficient to detect both known and unknown attacks to high accuracy. Machine learning intrusion detection systems have proven their success in identifying unknown attacks with high precision. Nevertheless, machine learning models are also vulnerable to attacks. Adversarial examples can be used to evaluate the robustness of a designed model before it is deployed. Further, using adversarial examples is critical to creating a robust model designed for an adversarial environment. Our work evaluates both traditional machine learning and deep learning models' robustness using the Bot-IoT dataset. Our methodology included two main approaches. First, label poisoning, used to cause incorrect classification by the model. Second, the fast gradient sign method, used to evade detection measures. The experiments demonstrated that an attacker could manipulate or circumvent detection with significant probability.

**Keywords:** adversarial; machine learning; network IDS; Internet of Things

check for updates

**Citation:** Papadopoulos, P.; Thornewill von Essen, O.; Pitropakis, N.; Chrysoulas, C.; Mylonas, A.; Buchanan, W.J. Launching Adversarial Attacks against Network Intrusion Detection Systems for IoT. *J. Cybersecur. Priv.* **2021**, *1*, 252–273. https://doi.org/10.3390/jcp1020014

Academic Editor: Nour Moustafa

Received: 19 February 2021
Accepted: 20 April 2021
Published: 23 April 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

The internet is increasingly being populated with new devices and emerging technologies, which increases the surface of attacks, thus allowing cybercriminals to gain control of improperly secured devices. Nowadays, we are facing a more and more digitalised world, so it is more crucial than ever to detect and prevent cyber attacks as quickly as possible. Different entities ranging from homeowners to nation-states are subject to cyber-attacks, thereby increasing the challenge posed when creating mature security measures. Moreover, the malicious parties build new skills, using more complex attack techniques to circumvent detection measures put in place [1].

The internet is shifting towards increasing the connectivity of physical devices, called the Internet of Things (IoT), where physical objects can become active participants in business processes [2]. Increasing connectivity poses problems for businesses that need to spend more resources to protect their devices. The security standards for IoT devices are not defined and often become an after-thought in a profit-driven market [3]. Gartner predicts that by 2023 there will be 48.6 billion IoT devices connected using 5G [4]. In addition to that, as IoT technologies emerge, the security risks will increase, and traditional protection methods will become obsolete [5].

Intrusion Detection Systems (IDS) are often put in place as a security measure to protect nodes within a network. Due to the nature of limited resources within an IoT device, IDS are often placed at the network perimeter to detect adversarial activity [6]. One of the IoT devices' primary aim is to keep the device's processing load to a minimum.

As such, Host Intrusion Detection Systems (HIDS) are often avoided in the IoT ecosystem because of the resource-intensive activities, including file or process monitoring [7].

Machine learning models are trained using large datasets to learn patterns within the data, allowing the model to determine whether an activity is benign or not [1]. In 1999 the KDD-CUP-1999 dataset [8] was released and became one of the most popular datasets in research related to machine learning Network Intrusion Detection Systems (NIDS). The KDD-CUP-99 was later updated to the NSL-KDD dataset [9]. Over the last two decades, both of these datasets have been losing relevance in modern IDS solutions because of the absence of modern attack methods and new protocols [10]. However, these datasets have been defined as biased and unrealistic by multiple studies [11–13]. In 2018, ref. [14] published the Bot-IoT dataset, which was the first dataset to focus on machine learning IoT security solutions [15].

However, machine learning-based IDS solutions are vulnerable when the model is targeted and exploited by adversarial cybercriminals. Adversarial machine learning methods intend to cause incorrect data classification forcing the implemented model to fail [16–19]. Within an adversarial environment, it is critical to anticipate the actions an adversary may take towards an implemented model [20]. To create the best model, previous research showed the importance of using adversarial examples within a dataset when training a machine learning model [17,21].

The Bot-IoT dataset has not knowingly been created using adversarial examples towards the machine learning model. For this reason, our work aims to evaluate the Bot-IoT dataset and model robustness against adversarial examples. Our contributions can be summarised as follows:

- We mount label noise adversarial attacks against an SVM model that detects malicious network traffic against IoT devices using the Bot-IoT dataset [14].
- We generate adversarial examples using the Fast Gradient Sign Method (FGSM) against binary and multi-class Artificial Neural Networks (ANNs) using the Bot-IoT dataset [14].
- Finally, we analyse and critically evaluate the experimental results along with the model robustness against adversarial examples.

The rest of the paper is organised into five sections, as follows. The literature review and background knowledge are presented in Section 2. Section 3 briefly explains our methodology for the experimental activities. The implementation and the experimental results are presented in Section 4, and are followed with our evaluation and discussion in Section 5. Finally, Section 6 draws the conclusion, giving some pointers for future work.

## 2. Background

### *2.1. Intrusion Detection Systems*

There are three popular approaches to protect networks. The first approach is to detect an attack by setting up alert triggers, for example, once a threshold is exceeded. Such a detection strategy informs the administrators that an attack occurs but does not prevent it. The second approach is to prevent an attack by setting protection mechanisms that deny the attack from occurring, which raises a problem if a legitimate action is deemed illegitimate, resulting in a loss of service availability. The final approach is to block an attack, which entails setting protections in place in retrospect of the attack, preventing or detecting the attack when it next occurs. In the latter two approaches, the IDS is configured as an Intrusion Prevention System (IPS) [5].

There are two locations on which IDS can be placed depending on the sources of information; firstly, sensors can be placed on a Host system (HIDS) and secondly, sensors can be placed on the Network (NIDS) [10,22]. The HIDS monitors a single host's activity such as running processes, system calls, or code in memory [3,10,22]. The NIDS monitors the network traffic by performing packet inspection to check protocol usage and their services or IP addresses of communication partners [3,10,22]. NIDS could commonly be placed at the gateway of ingress or egress traffic at the organisation network perimeter.

Therefore, it would be possible to observe all traffic from external adversaries [23]. An IDS is an essential tool used to identify intrusions; therefore, one of the most widespread problems in cybersecurity is creating sophisticated methods and generating effective rules for the NIDS to be beneficial [1]. An IDS's success is often rated according to their accuracy values when testing controlled data of the known legitimate and illegitimate activity. The accuracy is made up of four metrics [24,25], and are used when calculating the accuracy seen in Equation (1). We have True Positive (*TP*) when an attack is correctly identified and a True Negative (*TN*) when a benign incident is correctly identified. When a benign incident is identified as an attack, we have a False Positive (*FP*), and when an attack is identified as a benign incident, we have a False Negative (*FN*).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{1}$$

$$True\ Positive\ Rate = \frac{TP}{(TP + FN)} \tag{2}$$

$$True\ Negative\ Rate = \frac{TN}{(TN + FP)} \tag{3}$$

$$False\ Positive\ Rate = \frac{FP}{(FP + TN)} \tag{4}$$

$$False\ Negative\ Rate = \frac{FN}{(TP + FN)} \tag{5}$$

An IDS can be classified based on its detection method into: (i) knowledge-based IDS, where the detection is based on static knowledge, (ii) behaviour-based IDS, where the detection is based on dynamic behaviour that has been monitored, and (iii) hybrid IDS, which is a combination of (i) and (ii) [10]. More specifically, a knowledge-based IDS, also known as signature-based, is a system in which alerts are triggered based on a list of pre-determined knowledge, such as hashes, sequence of bytes, etc. A behaviour-based IDS, also known as anomaly-based, takes on the principle of having a normal activity model, triggering alerts when anomalous activity is detected. Creating a baseline of normal activity can sometimes be inaccurate if there is malicious activity captured within the baseline. Moreover, a representative normal activity baseline is unlikely possible with an ever-changing and expanding network [5,26]. A behaviour-based IDS is capable of detecting zero-day attacks which have not yet been disclosed to the public [10]. However, due to the nature of alerting all deviations from normal activity, they are known to produce high FPRs [10,26]. Thirdly there is a hybrid IDS that makes use of both knowledge-based IDS and behaviour-based IDS. A hybrid-based IDS can detect both zero-day as well as known attacks [10].

### 2.2. Internet of Things Security

The IoT is a market that is populated with a myriad of device types. The purpose of the IoT is to interconnect physical objects seamlessly to the information network. IoT thrives on heterogeneity which enables the different devices to connect. As IoT devices are often manufactured using different hardware, specific protocols and platforms must be created for these devices [23]. On the other hand, heterogeneity limits security advancements because there is no universal international standard. Furthermore, because of the lack of security consideration, there is some reluctance to deploy IoT devices, as both businesses and independent parties are unwilling to risk their security [3].

As IoT attacks are increasing in popularity, and the significance of data is high (e.g., within the health sector or critical infrastructure), the incentive to improve IoT security is increasing. However, through a profit-driven business model in a competitive market, IoT security is often considered an afterthought. Data theft is regarded as a high impact risk; however, sometimes, IoT data could be temperature monitoring, which is considered

trivial unless the integrity is manipulated and fails to alert deviations. Until now, the most challenging attacks in the IoT field have been botnet attacks [6,14]. A botnet consists of several zombie devices which an attacker is controlling. A large scale botnet enables a full-scale Denial of Service (DoS) attack to be carried out—such as Mirai botnet, or Bashlite [6]. Moreover, botnets can be used for information gathering attacks such as port-scanning and port-probing to gather information about running services on their targets [6].

Since the IoT network poses many challenges, and there are newly introduced network varieties, traditional NIDS can no longer act independently. The importance to protect the network lies in the pursuit of privacy. Data privacy seeks to accomplish two things. Firstly to maintain data confidentiality, and secondly, to uphold anonymity and to make it impossible to correlate data relations between the data and their owner [27].

### *2.3. Machine Learning Models and Examples*

To assist behaviour-based IDS to reduce the number of FPRs mentioned in Section 2.1, researchers have demonstrated success using machine learning IDS models [6]. Since a behaviour-based IDS does not rely on static knowledge, it can cope with the ever-growing network of devices and numerous zero-day attacks [10,28]. Machine learning models are trained using a preferably large dataset, and the intention is that the individual models recognise patterns within the data to decide whether an activity is malicious or not.

Machine learning models are primarily divided into three groups; firstly, supervised learning, secondly, unsupervised learning, and finally, semi-supervised learning [10,28]. Supervised learning models are trained using labelled data where the outcome of the data is known. Labelled data might classify network traffic as benign or as an attack. On the other hand, unsupervised learning occurs when a model is trained purely using unlabelled data. The aim is for the model to recognise patterns within the data to classify input data. Semi-supervised learning is a mixture of supervised and unsupervised datasets, meaning that some data is labelled and other data is not. Semi-supervised learning can be useful when the dataset would be excessively time-consuming to label exhaustively.

Deep learning is a subset of unsupervised machine learning and intends to imitate the human brain structure through the use of neural networks. Deep learning has proven success with regards to image recognition, speech processing, and vehicle crash prediction, among other areas [29]. More recently, deep learning has shown advancements in regards to intrusion detection when compared to traditional machine learning methods [26]. The success of deep learning comes through hierarchical pattern recognition within the features [22].

However, machine learning techniques are not the golden ticket towards the perfect IDS. It is essential to use a dataset that does not have a relatively high number of redundant values because these can increase the computation time and reduce the IDS model accuracy. Moreover, the dataset must not have unnecessary features; otherwise, the accuracy would be reduced [27]. A significant risk of machine learning IDS being considered is using inaccurate or non-integral data to train the model. For example, if an adversarial insider labels malicious activity as benign, then the IDS will not detect those attack vectors, reducing the accuracy of an IDS.

Traditional machine learning models applied to NIDS may include decision tree C4.5, Random Forests (RF), and Support Vector Machines (SVM) [29]. The C4.5 machine learning model is increasingly considered out of date and is being replaced by RF models that are made up of a forest of many smaller decision trees, each specialising in different patterns.

SVM plot each data instance within an $n$-dimensional space, where $n$ is the number of features [14]. The classification finds the hyperplane distinguishing classes [14].

Deep learning models may include Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN) [15]. RNNs are extended neural networks that use cyclic links, learning using memory where outputs further influence inputs for the neural network [14]. CNNs automatically generate features through rounds of learning processes [26].

A confusion matrix is the foundation towards further metrics known as precision, recall, and F1-score. Precision is the ratio of the total number of true positives against the total number of positive predictions made, as seen in Equation (6). The recall score is the model's ability to predict all positive values and can be seen in Equation (7). The F1-score is a weighted average of the recall and precision scores, as seen in Equation (8). The final metric is the Receiving Operator Characteristic (ROC) curve and the Area Under Curve (AUC) score. The ROC curve is plotted against the FPR and TPR, and the AUC is the amount of space under the ROC curve. When evaluating the AUC score, it is desired to have as high of a number as possible.

$$Precision = \frac{TP}{(TP + FP)} \tag{6}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{7}$$

$$F1\ Score = 2 \times \frac{(Recall \times Precision)}{(Recall + Precision)} \tag{8}$$

### 2.4. Datasets

There are several datasets used in the IDS literature, such as the KDD-CUP-99, the UNSW-NB15, the CIC-DoS-2017, and the CSE-CIC-IDS [10]. Nonetheless, a primary challenge of applied machine learning in IDS is the lack of publicly available and realistic datasets that can be used for model training and evaluation [10,14]. Organisations are not willing or allowed to disclose network traffic data as they can reveal sensitive information about the network configuration and raise privacy concerns [30].

Moreover, several datasets have been used to study adversarial attacks in the relevant literature, such as the KDD-CUP-99, the UNSW-NB15, the CIC-DoS-2017 and the CSE-CIC-IDS [31]. The KDD-CUP-99 dataset is the most widely researched dataset for the evaluation of IDS models [15], containing seven weeks of network traffic [8]. However, the KDD-CUP-99 dataset is heavily criticised by researchers stating that the data is inadequate for machine learning models due to the high number of redundant rows [9]. The UNSW-NB15 [32] collected data using various tools such as IXIA PerfectStorm, Tcpdump, Argus, Bro-IDS, in order to identify nine attack factors Fuzzers, analysis, backdoors, DoS, exploits, generic, reconnaissance, shellcode and worms [32]. The CIC-DoS-2017 [33], is focused on DoS attacks due to the recent increase in these attacks [33]. The CSE-CIC-IDS [34] consists of seven attack categories: Brute-Force, Heartbleed, Botnet, DoS, Distributed Denial of Service (DDoS), Web Attacks and Insider Attacks.

Until recently, there have been no specific IoT datasets proposed for machine learning IDS evaluation [15]. The Bot-IoT dataset is the first to focus on an IoT network architecture and includes five different attacks; DDoS, DoS, operating system and service scans, key-logging, and data exfiltration [14]. The Bot-IoT dataset makes use of the Message Queuing Telemetry Transport (MQTT) protocol—a lightweight messaging protocol used in IoT architectures [15]. There are over 72 million rows of data in the complete dataset; therefore, a 5% version has also been released for more manageable use. The dataset released with 46 features, and the authors proposed a top-10 feature list using the Pearson Correlation Matrix and entropy values [14]. There have been works demonstrating the efficacy of both traditional and deep learning techniques with this dataset [5].

### 2.5. Related Work

Machine learning and especially deep learning models introduce significant advantages concerning FPRs and accuracy [10,22]; however, they are not without their flaws. Machine learning models are often vulnerable to attacks with the intent to cause incorrect data classification [17,18]. Incorrectly classified data could enable an attacker to evade the IDS, thereby putting the organisation at the risk of undetected attacks. There are three primary categories of offences which include influential attacks, security violation attacks,

and specificity attacks [20,35]. Influential attacks impact the model used for intrusion detection and include causative attacks that entail altering the training process by manipulating the training data and exploratory attacks that attempt to gather a deeper understanding of the machine learning model [20,35].

Security violation attacks target the CIA principles concerning the machine learning model [20,35]. An attacker can infer such information through analysis after gaining access to data used for model training. An attack on model integrity aims to influence the model's metrics, such as Accuracy, F1-Score, or AUC [16]. Consequently, if the model's integrity cannot be maintained, the attacker could force a model to classify data incorrectly and bypass the model's detection. An attack on the model availability would trigger a high number of FP and FN alerts which could render a model unavailable for use. Moreover, if there are numerous alerts, an alert triage agent could experience alert fatigue, where the focus per alert will drop. In turn, the alert fatigue could lead the triage agent to dismiss a valid alert.

Adversarial attacks in traditional machine learning are more effective at the training stage rather than the testing stage. Attacking the model at the training stage will create fundamental changes to the machine learning model's classification accuracy. In contrast, an attack at the testing or deployment stage can only take advantage of inherent weaknesses in the model [36]. An adversary can affect either the features of a dataset (feature noise) or the labels of the dataset (label noise) [36]. Label noise or also referred to as label flipping, focuses on changing the labels that are used to train a model. Label flipping seems to be a more effective method of tricking a machine learning model [21,37]. The threat model of a label flipping attack is to influence the model by providing inaccurate labels during the model's training. If the labels are not correct, then the model cannot be trained integrally and correctly [38]. There are different strategies related to label flipping, including random or targeted attacks. Random label flipping entails selecting an arbitrary percentage of labels and changing them. For example, a label classifying a record as malicious could be switched to benign or vice versa. Targeted label flips focus on flipping labels that have the most significant effect on the model's performance. According to [39] SVMs can often be more robust against adversarial machine learning, and therefore a suitable choice for deployment in adversarial environments. If attackers have access to feed incorrect labels to the machine learning model during training, then they can likely best achieve their objective [20].

Deep learning is a subset of machine learning that shown success in classifying well-labelled data. Therefore, if labels are not accurately assigned, then the resulting model accuracy will also be inaccurate [38]. Conventional adversarial machine learning methods such as label flipping do not apply to deep learning models because the model's data is split into batches; it would be required for labels to be flipped within every batch [17]. The authors of [40] proposed the Fast Gradient Sign Method (FGSM) to generate new perturbed data. The FGSM is a linear manipulation of data; therefore, it is quick to generate new data [40]. The FGSM aims to increase the deep learning model's cost than the gradient of the loss with the input image to generate a new image. The work in [41] proposed a family of attacks under the name of Jacobian-based Saliency Map Attack (JSMA). JSMA is more computationally heavy than FGSM; however, it can create adversarial examples which appear closer to the original sample [41]. The work of [42], extended the JSMA and introduced targeted and non-targeted methods to generate adversarial examples. Nevertheless, JSMA uses more computational power than FGSM [43].

In 2018, ref. [44] presented complete success in using the FGSM to get on average 26% incorrect classification per class with their neural network model. As reviewed in Section 2.5, [44] used the NSL-KDD dataset and perturbed the dataset using an epsilon value of 0.02. The experiments carried out in this paper have received a higher percentage of incorrect classification, while the epsilon value has been at least 0.1. The result suggests that the Bot-IoT dataset and neural networks are more robust towards adversarial FGSMs than the NSL-KDD dataset.

Nevertheless, ref. [45] brought up the concern that the FGSM perturbs the entire dataset of which it is given. The impact of complete data perturbing is two-fold. Firstly, by perturbing the Bot-IoT dataset, the features primarily impacted are those residing within network packet headers. A layer seven firewall could likely detect these kinds of attacks depending on the algorithm used. Secondly, ref. [45] propose to use the JSMA method instead of FGSM because JSMA perturbs a portion of the dataset rather than the entire dataset. On the contrary, ref. [43] suggests that FGSM is better used in experiments because of the drawback that JSMA has with high computational power. FGSM against the NSL-KDD was also used by [46], who found that perturbing the reduced the accuracy of their CNN from 99.05% to 51.35%. Besides, ref. [46] experimented against the NSL-KDD dataset using the JSMA method and found that the accuracy decreased from 99.05% to 50.85%. Comparing these results to the findings of our work's experiments, we observed a similar decrease in the accuracy, as it can be seen in Section 4.

Countermeasures can be categorised into reactive and proactive [16]. Reactive countermeasures are only able to detect adversarial behaviour after the neural network has been built and deployed. Reactive countermeasures could be input reconstruction, and data sanitation, which could be performed using multiple classifier systems or statistical analysis [21,38]. Yuan et al. propose re-training to be a proactive countermeasure against adversaries, as adversarial training could prepare a model in an improved manner [16]. Furthermore, if the machine learning model is planned to be deployed in an adversarial environment, then it is crucial to use adversarial methods to train a robust model [17,21]. Machine learning in an adversarial environment requires analysis of attacker's methods to force inaccuracies from machine learning model [20].

Our work differentiates from all previous approaches because it is the first to attempt both label noise attacks and adversarial examples generation using the Bot-IoT dataset [14]. Our results support the fact that the Bot-IoT dataset combined with neural networks generates a more robust setup compared to the NSL-KDD dataset.

## 3. Methodology

### *3.1. Dataset Overview*

There have already been many adversarial machine learning works performed on the datasets of the liteterature [31]. However, the Bot-IoT dataset finds itself still in its early days, and to the best of our knowledge, no other work in the literature has used it to study adversarial machine learning. The Bot-IoT dataset was created in a simulated environment formed up of victim and attacking machines [14]. The traffic was captured in .pcap files, analysed and exported to comma-separated value (.CSV) files resulting more than 73 million records with 46 features, including three classification features. The work of [14], categorised the different attacks into five classes; (1) Normal, (2) Reconnaissance, (3) DDoS, (4) DoS, (5) Information Theft. For the easier management of the dataset, ref. [14] extracted a random 5% selection of each class which reduces the amount to approximately 3.6 million records. In Table 1 the value counts of each category can be seen. The total number of records is 73,370,443, including benign traffic.

**Table 1.** Bot-IoT category value counts.

| Category | Full Amount | 5% Amount | Training Amount | Testing Amount |
|---|---|---|---|---|
| DDoS | 38,532,480 | 1,926,624 | 1,541,315 | 385,309 |
| DoS | 33,005,194 | 1,650,260 | 1,320,148 | 330,112 |
| Normal | 9543 | 477 | 370 | 107 |
| Reconnaissance | 1,821,639 | 91,082 | 72,919 | 18,163 |
| Theft | 1587 | 79 | 370 | 14 |
| Total | 73,370,443 | 3,668,522 | 2,934,817 | 733,705 |

In Section 2.3, the importance of feature selection was emphasised. Feature selection activities were performed using the Correlation Coefficient and Joint Entropy Score methods against the dataset. Using the feature selection metrics, a top 10 list of features was extracted and are highlighted in Table 2 [14]. There are three classification features in this dataset. The first classification feature is *attack* which is intended for binary classification. The label is either True or False corresponding to malicious or benign traffic, respectively. All the various attack categories are labelled as True. The second classification feature is a five-class multi-classification feature, namely *category*, and is made up of the string values seen in Table 1. The final classification feature is *subcategory* and is composed of ten classifications. The subcategory classification feature is more fine-grained than the category classification feature. For example, the DoS category is split into DoS by protocol (TCP, UDP, HTTP). Information theft was categorised into keylogging and data theft. Reconnaissance is divided into service scanning and OS fingerprinting, thus completing the ten subcategory class values. Nevertheless, Table 1 depicts a severe data imbalance leaning towards DDoS and DoS classes than the others, something that intuitively makes sense as the IoT devices are used more frequently for such attacks as zombie devices. Besides, a common practice in the literature to combat this asymmetry is introducing class weights during the training process [47].

**Table 2.** Total features in the Training dataset [14].

| Features | Description |
|---|---|
| pkSeqID | Row Identifier |
| Proto | Textual representation of transaction protocols present in network flow |
| saddr | Source IP address |
| sport | Source port number |
| daddr | Destination IP address |
| dport | Destination port number |
| attack | Class label: 0 for Normal traffic, 1 for Attack Traffic |
| category | Traffic category |
| subcategory | Traffic subcategory |

| Top-10 Features | Description |
|---|---|
| seq | Argus sequence number |
| stddev | Standard deviation of aggregated records |
| N_IN_Conn_P_SrcIP | Number of inbound connections per source IP. |
| min | Minimum duration of aggregated records |
| state_number | Numerical representation of transaction state |
| mean | Average duration of aggregated records |
| N_IN_Conn_P_DstIP | Number of inbound connections per destination IP. |
| drate | Destination-to-source packets per second |
| srate | Source-to-destination packets per second |
| max | Maximum duration of aggregated records |

### *3.2. Orchestrating Attacks against Machine Learning*

The first stage was to replicate the linear SVM model that was proposed by [14]. The purpose of the aforementioned replicating is to enable valid comparison of metrics such as accuracy, recall, precision, F1-scores. An artificial neural network (ANN) was implemented to train and evaluate data more quickly than the RNN and LSTM models.

Nevertheless, the activation functions and neural network structure remained the same for a fairer comparison. The second and third stages focused on adversarial examples for both the SVM and ANN. With SVM adversarial examples, the undertaken method was labelled noise generation during the training phase. As previously discussed, the label noise generation is a more effective attack against traditional machine learning models than feature noise experiments. Firstly, a certain percentage of data labels was manipulated, ranging from 0% to 50% of flipped labels. The SVM model was trained using adversarial examples. The increasing rate of flipped labels should incrementally reduce the accuracy of the SVM models. However, as discussed in Section 2.5, random label manipulation could also result in little effect on model accuracy. Accordingly, targeted label manipulation was also performed. On the default SVM model, each label's margin to the SVM hyperplane is calculated; consequently, the top selected percentage (ranging from 0% to 50%) with the smallest margin to the SVM hyperplane was chosen, and their label is flipped. Using the targeted label flipping method should result in a more significant impact on model accuracy and negatively impact the metrics.

After experimenting with SVM manipulation, FGSM methods were applied to the data used for the ANN. In contrast to the SVM threat model, which assumes that the attacker can manipulate the training data, the FGSM assumes that the attacker controls the data and tries to evade the model after it has been deployed. The CleverHans package was used to generate adversarial examples based on the testing dataset, which simulates attacking the model after the deployment [48]. As discussed in Section 2.5, the noise factor determines the level at which the generated adversarial examples will differ from the non manipulated data. The work of [44], found that a noise factor of 0.02 was sufficient to get 100% incorrect classification on the NSL-KDD dataset. However, as the NSL-KDD dataset is smaller in size than the Bot-IoT dataset, a noise factor from 0 to 1, with an increment in steps of 0.1, was applied and evaluated. The ANN was trained on a binary classification model, similar to that of the RNN and LSTM models proposed by [14]. However, neural networks have proven to be highly successful using multi-class classification. As the Bot-IoT dataset also contains five multi-class label features, experimentation using FGSM was carried out in multi-classification mode.

### *3.3. Evaluation Criteria*

The fourth and final stage is to evaluate the results, comparing adversarial examples to the results without any data manipulation. The accuracy, precision, recall, and F1-scores were compared to evaluate both the SVM and ANN models. In addition to these numeric metrics, confusion matrices were generated, which graphically represent the incorrect classifications. Further, as explained in Section 2.3, the recall score is a key-metric as a decrease shows the extent that the model classifies false negatives. An increased count of FNs could allow a cyber attacker to launch attacks which the machine learning or deep learning models would not detect.

## 4. Implementation and Results

### *4.1. Data Preparation*

The first stage of data preprocessing is to reduce the number of features to increase accuracy and decrease the chance of overfitting models. Ref. [14] used the Correlation Coefficient and the Joint Entropy Scores to extract the top-10 features of this dataset, as described in Table 2. For more manageable model development, ref. [14] have excerpted 5% of the total records resulting in 3.6 million records. [14] have also already split the 5% extracted records into training and testing splits, sized 80% and 20% respectively. The released datasets include additional features which could be useful for further statistical analyses; making a total of 19 features. The full list of features, their description and the top-10 features can be seen in Table 2. The features proto, saddr, sport, daddr, dport can be used to identify data points uniquely and are therefore considered flow identifiers and

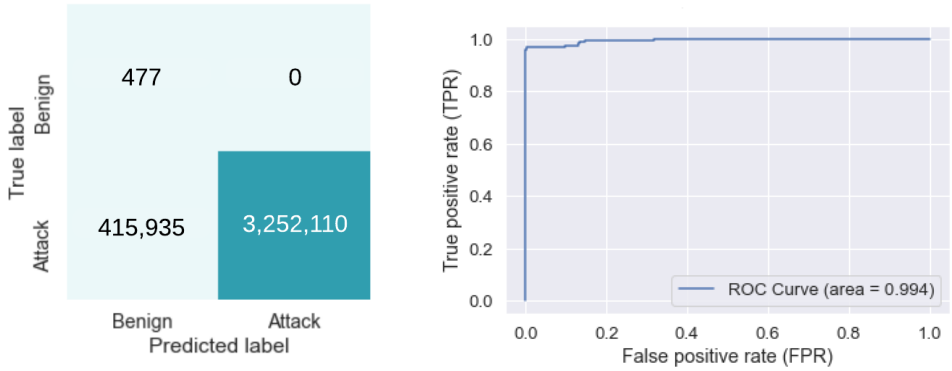
are removed for model development [14]. The pkSeqID feature is a row identifier and is consequently also removed for model development.

The training features did not need to be encoded as the data were already in numeric form. However, the category feature used for five-class multi-classification must be encoded using Scikit-Learn's preprocessing OneHotEncoder function, which transforms categorical features into a numerical array. As the margin of labels to the hyperplane will be measured and used for label manipulation, the data must be normalised. The purpose of data normalisation is to apply an equal scale to the training and testing splits without distorting the ranges within the relative values. Scikit-learn's preprocessing MinMaxScaler can be used to scale the data between values of $-1$ and 1.

After pre-processing the data, it was essential to train a machine learning model using trusted datasets. The trusted dataset means that there has been no data manipulation performed yet, which ensures data integrity. Using data that maintains integrity allows the creation of models which perform at their best. Besides, the trusted models can be compared to the models that undergo adversarial training or testing manipulation.

#### 4.1.1. SVM Trusted Model

The trained SVM model is a Support Vector Classifier (SVC) using the linear kernel. The hyper-parameters were primarily the default parameters in the Scikit-Learn package, except for the penalty score set to 1 and the maximum iterations set to 100,000 as replicated by [14]. Rather than evaluating the model using the training and testing split and as explained in Section 2.3, four-fold cross-validation was used. Similar to the method of [14], the combined training and testing datasets were used to generate the confusion matrix. The generated confusion matrix can be seen in Figure 1a, and the ROC curve in Figure 1b. Viewing the confusion matrix, the model inaccurately predicts 415,935 true attack traffic as predicted benign traffic. The ROC curve and, consequently, the AUC score show that the model was highly capable of detecting attack traffic over benign traffic. Further, the accuracy, recall, precision and F1 scores can be seen in Table 3.




(**a**) (**b**)

**Figure 1.** (**a**) Support Vector Machines confusion matrix without label flipping. (**b**) Support Vector Machines Receiving Operator Characteristic curve without label flipping.

#### 4.1.2. ANN Trusted Model

The trained ANN had one input layer, three intermittent layers and one output layer. The input layer was composed of ten nodes, equally to the number of features. The intermittent layers are 20, 60, 80, 90, respectively. The output layer was either two for binary classification or five for multi-class classification. The activation function of the intermittent layers are TanH, and the output activation layer was Sigmoid. Sigmoid was chosen for the output layer to replicate the [14] method because Sigmoid models are more robust [40]. The ANN designs for both binary and five-class classification layers can be seen in Figure 2. The left side shows the ANN design for the binary classification, while the right side shows the ANN design for multi-class classification.

| | | |
|---|---|---|
| dense_1: Dense | input | (None, 1, 10) |
| | output | (None, 20) |
| | Param # | 220 |

| | | |
|---|---|---|
| dense_2: Dense | input | (None, 20) |
| | output | (None, 60) |
| | Param # | 1260 |

| | | |
|---|---|---|
| dense_3: Dense | input | (None, 60) |
| | output | (None, 80) |
| | Param # | 4880 |

| | | |
|---|---|---|
| dense_4: Dense | input | (None, 80) |
| | output | (None, 90) |
| | Param # | 7920 |

| | | |
|---|---|---|
| dense_5: Dense | input | (None, 90) |
| | output | (None, 2) |
| | Param # | 182 |

| | | |
|---|---|---|
| dense_6: Dense | input | (None, 1, 10) |
| | output | (None, 20) |
| | Param # | 220 |

| | | |
|---|---|---|
| dense_7: Dense | input | (None, 20) |
| | output | (None, 60) |
| | Param # | 1260 |

| | | |
|---|---|---|
| dense_8: Dense | input | (None, 60) |
| | output | (None, 80) |
| | Param # | 4880 |

| | | |
|---|---|---|
| dense_9: Dense | input | (None, 80) |
| | output | (None, 90) |
| | Param # | 7920 |

| | | |
|---|---|---|
| dense_10: Dense | input | (None, 90) |
| | output | (None, 5) |
| | Param # | 455 |

**Figure 2.** Artificial Neural Networks Design.

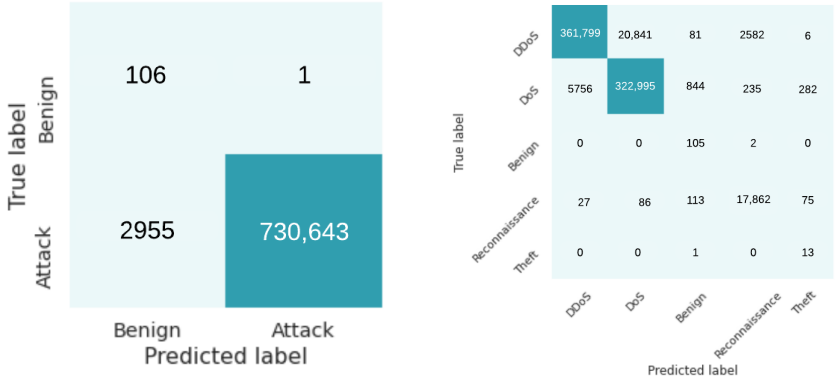
The confusion matrices for the ANN were calculated using the testing dataset. The testing dataset is used rather than the full dataset with the SVM model because there is a different threat model. As stated in Section 3.2, the ANN's threat model was that the attacker can manipulate data after the model has been deployed. The testing dataset was used to simulate adversarial examples by an attacker. The binary classification confusion matrix in Figure 3a, shows that the model incorrectly classified 2955 true attack records as benign. In Figure 3b, the confusion matrix for five-class multi-classification shows that the model inaccuracies lie in predicting more labels as DDoS or DoS compared to their true labels. Further, Table 3 shows that the ANN can report very high accuracy, recall, precision, and F1 scores while maintaining a low loss score.

**Table 3.** SVM and ANN scores without label flipping and adversarial examples.

| (a) SVM scores without label flipping | |
|---|---|
| **Scoring** | **Percentage (%)** |
| Accuracy | 85.897 |
| Recall | 85.895 |
| Precision | 100 |
| F1 | 91.255 |
| (b) ANN scores without adversarial examples | |
| **Scoring** | **Percentage (%)** |
| Accuracy | 99.692 |
| Loss | 1.170 |
| Recall | 99.813 |
| Precision | 99.591 |
| F1 | 99.702 |

### *4.2. Creating Adversarial Examples*

This subsection details the undertaken activities to generate adversarial labels and is split into two parts. The SVM part details how to perform random and targeted label flipping on the training dataset. The ANN part specifies how to use the CleverHans package to generate adversarial examples [48]. The CleverHans FGSM also implements a targeted and non-targeted approach. The targeted FGSM approach aims to alter the dataset to appear similar to the classification feature. On the contrary, the non-targeted FGSM approach aims to modify the labels of the dataset.




(**a**) (**b**)

**Figure 3.** (**a**) Binary Artificial Neural Networks confusion matrix without adversarial examples. (**b**) Multi-class Artificial Neural Networks confusion matrix without adversarial examples.

#### 4.2.1. Support Vector Machine Label Flipping Activities

As stated in Section 3.2, there are two approaches to flip the labels: random label flip and targeted label flip. Additionally, the number of labels to be flipped ranges from 5% to 50%, in 5% increments (5%, 10%, . . . , 45%, 50%) (Appendix A displays the method to flip a random percentage of labels). The first stage is to sample the training dataset, in this case, 5% (as defined by n), and store the indexes of this sample. The second stage is to alter the *attack* column and use a binary Exclusive OR operation to flip the label from 1 to 0 and vice versa. After flipping the labels, data must be pre-processed with the same steps seen in Section 4.1. Placing this code within a loop allows one to iteratively increase the percentage of labels manipulated and further and train the model with the modified dataset.

Prior to executing the SVM targeted label flip, each label's distance from the hyperplane requires to be measured. The distance to the hyperplane is known as the margin. As explained in Section 3.2, the records with the lowest margin should have the highest impact on the metrics. The impact is high because these labels affect the orientation of the hyperplane when a model is trained (Appendix B, shows a function that can be used to select the nearest $X$% record indexes). Upon retrieving the relevant indexes, the defined function applies a binary Exclusive OR operation to flip the label from 1 to 0 and vice versa.

#### 4.2.2. Creating Artificial Neural Network Adversarial Examples

The activities to create adversarial models using the FGSM are the same for the binary and the five-class multi-classification ANNs. The difference between the binary and five-class multi-classification lies in the feature column; hence, after the pre-processing stage, the data formatted correctly and is ready for the CleverHans 3.0.1 package [48].

The first stage of using the CleverHans package includes converting the Keras model into a logistic regression model (Appendix C). Following this, the testing dataset must be converted into a Tensorflow Tensor (Appendix D). After converting the testing dataset into a Tensor, an epsilon value must be defined as it determines the scale of which the data is perturbed. Using the `fast_gradient_method` import, it is possible to generate a new testing dataset (Appendix E, line 6 defines whether the FGSM is a targeted attack or not). The generated new testing dataset is used to evaluate the ANN model and a confusion matrix as well as the accuracy, loss, recall, precision, and F1 metrics are recorded.

### *4.3. Adversarial Example Results*

#### 4.3.1. Support Vector Machine Label Flipping Result

As described in Section 4.2.1, both a random and targeted selection of labels ranging from 0% to 50% were flipped. It was hypothesized that the targeted label manipulation should present a greater effect on how the SVM calculates its hyperplane. Appendix F,

Figure A1a shows that as the number of labels flipped increased to 50%, all of the scores decreased, showing success in label flipping towards manipulating the SVM. Nonetheless, looking at the metric gradients, it can be observed that the accuracy increased in some cases. The increase in accuracy is surprising as the increased number of label manipulation should cause the accuracy to decrease continuously. An example of accuracy increasing is seen from 20% to 25% of labels flipped.

Performing the random label manipulation activities, it was expected that the metrics could be influenced significantly or minimally depending on the random sample of labels selected.We assumed that the random sample selection included labels that had a small margin to the default hyperplane. In that case, the impact should be more significant compared to if the labels had a large hyperplane margin. The metrics from the experiments carried out are presented in Appendix F, Figure A1b. The metrics confirm the hypothesis and show eradicate metrics; there is no correlation to the increasing label manipulation percentage and the metrics retrieved. Table 4 contains the numerical metrics with 0% and 50% label manipulation.

**Table 4.** Effect of Zero vs. 50% label flips against the metrics using hyperplane margin method.

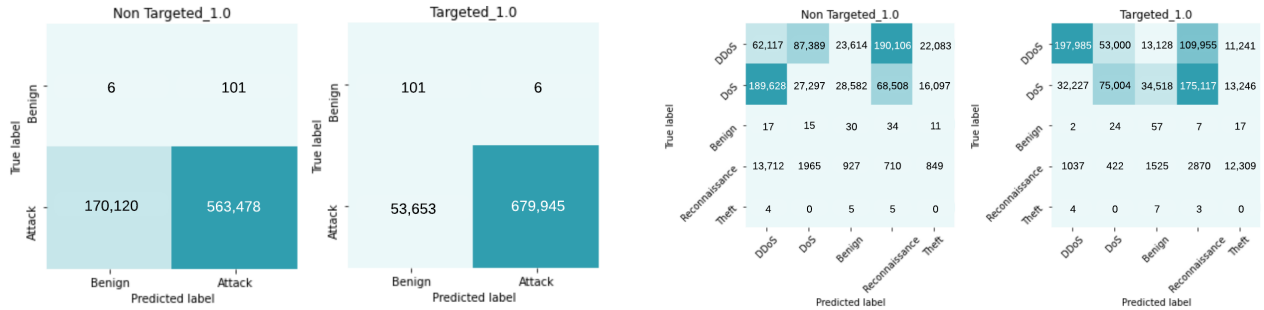
| **Scoring** | **Accuracy** | | **Precision** | | **Recall** | | **F1** | |
|---|---|---|---|---|---|---|---|---|
| Percentage of flipped labels (%) | 0 | 50 | 0 | 50 | 0 | 50 | 0 | 50 |
| Random Flip | 0.999 | 0.441 | 0.999 | 0.610 | 1.0 | 0.613 | 0.999 | 0.612 |
| Targeted Flip | 0.999 | 0.610 | 0.999 | 0.621 | 1.0 | 0.913 | 0.999 | 0.737 |

#### 4.3.2. Artificial Neural Network Adversarial Examples Result

In Appendix F, Figure A2b, the targeted FGSM is applied to the binary classification problem. The accuracy, precision, F1, and recall scores were reduced as the epsilon value grew. Additionally, the loss score increased as the epsilon value increases. However, the accuracy and precision scores are insignificantly impacted until the epsilon value is 0.8. The recall and precision scores are significantly impacted, with a lower epsilon score of 0.5. Table 5 contains the numerical metrics of zero epsilon as well as 1 epsilon values. Viewing Figure A2a which is the non-targeted FGSM to the binary classification problem, it is seen that the attack is more significant compared to the targeted attack. Figure A2a shows that the accuracy, precision, recall, and F1 scores are all impacted when the epsilon value is 0.2 and higher.

The Confusion matrices in Figure 4a are an alternative representation of the effect that a 1.0 epsilon value had. On the left in Figure 4a is the non-targeted method, and on the right is the targeted method. In the non-targeted method, it is seen that there is a greater amount of incorrectly predicted benign records. The greater amount of false-negative cases affects lowering the recall score significantly, as confirmed in Appendix F, Figure A2b. On the other hand, viewing Figure A3b, the targeted FGSM attack on the multi-class ANN model shows that the accuracy, precision, F1, and recall scores gradually decrease while the epsilon value grows to 1. Interestingly, in the non-targeted FGSM applied to the multi-class ANN model, the accuracy, precision, F1 and recall scores sharply fall until an epsilon value of 0.1, though the impact thereon is insignificant.

The confusion matrices shown in Figure 4b were created using the perturbed testing dataset, with an epsilon value of 1.0 in the multi-class ANN model. On the left side, the non-targeted FGSM is presented, while the targeted FGSM is presented on the right. Comparing these confusion matrices with the baseline multi-class confusion matrix in Figure 3b, it is observed that the model is lowering its capability of detecting DDoS and DoS attacks while incorrectly predicting a higher number of benign, reconnaissance, and theft attacks.




(**a**) (**b**)

**Figure 4.** (**a**) Confusion matrices of binary class perturbed with 1.0 epsilon. (**b**) Confusion matrices of multi-class perturbed with 1.0 epsilon.

**Table 5.** Effect of Zero epsilon vs. 1 epsilon against the metrics using Fast Gradient Sign Method in targeted and non-targeted modes.

| **Scoring** | **Accuracy** | | **Loss** | | **Precision** | | **Recall** | | **F1** | |
|---|---|---|---|---|---|---|---|---|---|---|
| Epsilon | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Binary Targeted | 0.996 | 0.927 | 0.016 | 0.151 | 0.996 | 0.895 | 0.996 | 0.563 | 0.996 | 0.690 |
| Binary Non-Targeted | 0.996 | 0.768 | 0.016 | 1.080 | 0.996 | 0.769 | 0.996 | 0.771 | 0.996 | 0.769 |
| Multi-Targeted | 0.956 | 0.421 | 0.045 | 1.764 | 0.952 | 0.312 | 0.957 | 0.493 | 0.955 | 0.382 |
| Multi-Not-Targeted | 0.956 | 0.141 | 0.045 | 2.403 | 0.952 | 0.153 | 0.957 | 0.249 | 0.955 | 0.189 |

## 5. Evaluation and Discussion

### *5.1. Machine Learning Model Metrics*

When training different models, one must consider the metrics that can be used when comparing them. Primarily, the focus is to achieve a high level of accuracy while also retaining low FPRs [30]. A confusion matrix can also be used with labelled datasets to visualise the values used to calculate the accuracy. It is possible to exhaustively evaluate the various machine learning models through a combination of the varying metrics. The evaluation stage of an IDS model is highly critical as it can help direct further improvements for the model during the testing phase. Moreover, a holistic evaluation can help to select the most applicable IDS model for deployment.

### *5.2. SVM Dataset Label Manipulation*

In Section 3.3, it was discussed that an increase in FNR would allow for a greater likelihood of the model not detecting malicious events. The FNR is inversely proportional to the recall score; albeit the FN count increases, the recall score will decrease. The recall metrics that are generated in Appendix F, Figure A1 have been extracted, and are seen in Figure 5. It is observed in Figure A1a that in targeted label flipping, the accuracy score falls at a quicker rate than the recall score in the case of targeted label flipping. This is because the number of FN is not increasing as quickly as the false positive rate. In the case of random label flipping, as observed in Figure A1b, the recall score is firmly related to the accuracy score. The strong correlation could mean that the accuracy is falling because the FNR increases more than the targeted experiment. Therefore, the outcome is that an adversary would have more success if they performed random label flipping. However, comparing these results to the works of [21,37], it was found that one must flip labels with a large margin to distort the hyperplane significantly. The hypothesis suggested in Section 3.2 of flipping labels with a small margin to the hyperplane is false. To distort the hyperplane, labels' generation with large margins must be selected.

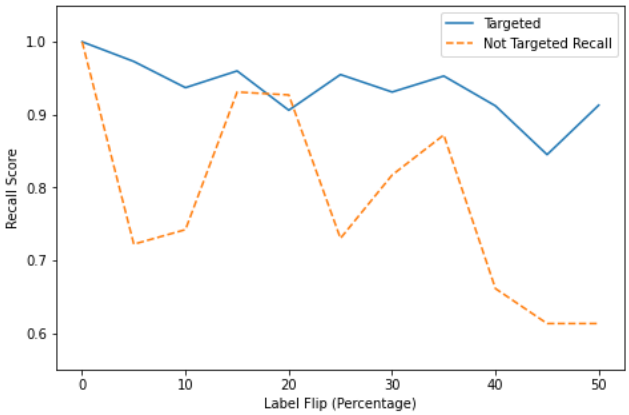



**Figure 5.** Comparing recall scores in targeted/non-targeted Support Vector Machines label flip.

Still, further concerns must be considered regarding the targeted label flipping attack against the SVM. The metrics show a decreasing trend as the percentage of label flips go up. However, it must be considered why the metric gradient is repetitively alternating from positive to negative. The complete effect of 0% and 50% label flip in Table 4 shows that the metrics all decrease, nevertheless in Appendix F, Figure A1 shows the alternating positive and negative metric gradients. This anomaly could be explained depending on how the hyperplane is adjusted after flipping $X$% of labels. For example, labels with a margin of 1% over those flipped labels may be manipulating the result. Therefore, the alternating positive and negative metric gradients is an anomaly.

Finally, the method of attack against the SVM must be considered. The threat model used against the SVM assumed that the adversary had access to the training dataset. The authors in [20] stated that if an adversary can manipulate the training dataset, the results should correspond with a strong negative impact on the metrics. Nevertheless, even though the accuracy and recall scores decreased, the metrics' amount could be considered insignificant. It took a large amount of data manipulation to substantially impact the metrics, using adversarial countermeasures such as statistical analysis, and this attack could be detected [21].

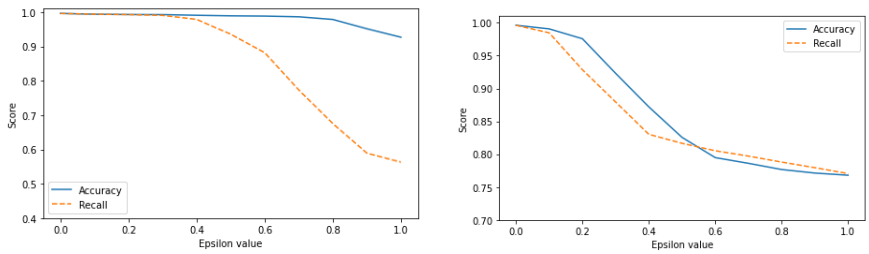
### *5.3. ANN Adversarial Example Generation*

In Section 3.1, it was described that the dataset could be used to train neural networks with a multi-classification problem. The experiments were carried out, generating adversarial examples using the FGSM against binary and multi-classification ANN models. The threat model assumed that the adversary could manipulate the data after the model had been deployed. An adversarial attack was simulated by manipulating the testing dataset. The CleverHans package can generate targeted, and non-targeted adversarial examples using the FGSM [48]. Both targeted and non-targeted FGSMs tested with the binary and multi-class ANN models. In Appendix F, the Figures A2 and A3 show that generating adversarial examples negatively impacted the metrics. The negatively impacted metrics show that the intrusion detection system failed to detect adversarial examples. Therefore an adversary could use the FGSM to bypass the machine learning IDS model.

#### 5.3.1. Binary Classification ANN Model

Using the targeted FGSM to generate adversarial examples showed decreased accuracy when the epsilon value reached 0.8. As explained in Section 3.3, a decreased recall metric shows an increase in false negatives, enabling an attacker to bypass the IDS. Figure 6a shows that as the epsilon value increases to 1, the accuracy falls from 99.8% to 92.7%, and the recall score falls from 99.7% to 56.3%. In addition, the confusion matrix seen in Figure 4a, shows a significant increase in false-negative classifications than the original confusion matrix in Figure 3a. The learning outcome is that while the accuracy remains high, the recall score falls significantly. For example, if an organisation accepts a model

with over 90% accuracy, then they are at a greater risk of the success of an undetected attack using the FGSM. Using the non-targeted FGSM to generate adversarial examples showed a more significant negative impact than the targeted FGSM. In Figure 6b, the accuracy score fell from 99.6% to 76.8%, and the recall score fell from 99.6% to 77.1%. It was noted in Section 4.2 that the CleverHans package could perform a targeted and non-targeted attack. The non-targeted attack focuses on trying to make the labels incorrect [48]. The attack on the labels forces the model to incorrectly classify any label, which could be why the accuracy is falling more significantly compared to the targeted attack.



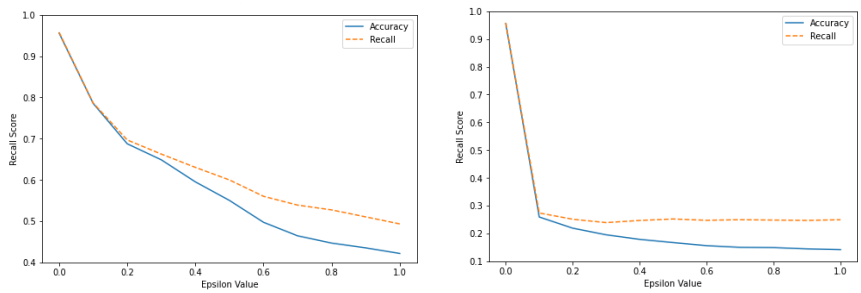

(**a**) (**b**)

**Figure 6.** (**a**) Comparing accuracy versus recall scores in binary targeted Fast Gradient Sign Method. (**b**) Comparing accuracy versus recall scores in binary non-targeted Fast Gradient Sign Method.

However, taking the confusion matrices in Figure 4a into consideration, it is observed that the non-targeted FGSM created adversarial examples results in a higher number of false negatives. The recall score falls significantly in the targeted FGSM and seen in Figure 7a because the model's ability to classify true positives correctly remains high.

### 5.3.2. Multi-Class ANN Model

Using the targeted FGSM against the multi-class ANN showed a gradual negative impact during the time the epsilon value grew to 1, as seen in Figure 7a. The accuracy score fell from 95.6% to 42.1%, and the recall score fell from 95.7% to 49.3%. The targeted FGSM aims to move the dataset in the direction of the feature class. In the non-targeted FGSM activities, the metrics had a greater impact. The metrics in the non-targeted FGSM fall sharply with an epsilon of 0.1, but insignificantly, thereafter, as seen in Figure 7b. The recall score falls significantly in the non-targeted FGSM; however, it must be considered that an adversary might not bypass the machine learning ANN IDS. Hence, as the false-negative rate increases, there is still a significant chance that an attack is detected but incorrectly labelled as another attack, thereby creating an alert. Nevertheless, the increased false-negative rate will create more overhead for those within an organisation that triage the alerts.




(**a**) (**b**)

**Figure 7.** (**a**) Comparing accuracy versus recall scores in multi-class targeted Fast Gradient Sign Method. (**b**) Comparing accuracy vs recall scores in multi-class non-targeted Fast Gradient Sign Method.

The confusion matrices of the ANN after applying the FGSM to the testing dataset in Figure 4b, to the confusion matrix without data manipulation in Figure 3b, it is seen that

the model is predicting a large portion of the DDoS and DoS traffic as benign. A small part of reconnaissance attacks were classified incorrectly as benign, and interestingly theft traffic was classified with a similar accuracy degree. The takeaway is that an adversary could launch DDoS and DoS more successfully, whilst data theft attacks would still be detected.

## 6. Conclusions

The study of IDS has become a rapidly growing field once more due to the involvement of machine learning models. Machine learning-based IDS are vulnerable to adversarial attacks, in which the intent is to cause incorrect data classification so adversaries can avoid detection. Our work aims to provide a deeper understanding of adversarial machine learning methods and how they can bypass machine learning-based IDS. Our work evaluated the robustness of the Bot-IoT dataset against label noise attacks using an SVM model. In addition to the SVM models, adversarial example generation was experimented using the FGSM against binary and multi-class ANNs.

The label flipping SVM experiment's threat model was white-box such that an adversary could control the input data used to train the model. The hypothesis predicted that those labels with the smallest margin would significantly affect how the SVM hyperplane was calculated. The result showed that a significant amount of labels had to be manipulated to affect the accuracy and, most importantly, the recall score. On the other hand, using the random flip method, the accuracy and recall scores could be impacted with a significantly lower amount of manipulated labels. Upon comparing the results to the related experiments, it was concluded that the manipulated labels with a high margin would affect the SVM hyperplane generation more significantly. The FGSM was used to generate adversarial examples to perturb the data for the ANN. The experiments performed adversarial FGSMs against both binary and five-class multi-classification ANN models. The threat model defined that the testing dataset would be perturbed using the FGSM using the CleverHans package in targeted and untargeted approaches. In the binary classification experiments, it was found that as the epsilon value grew to 1, the accuracy, recall, precision and F1 scores dropped. In turn, the results showed that perturbed data was able to trick the ANN significantly. The five-class multi-classification experiments found that as the epsilon value grew to 1, the accuracy, recall, precision, and F1 scores dropped. The increasing epsilon value perturbs the data more significantly, which resulted in more significantly impacted metrics.

Our results show that the Bot-IoT classes are imbalanced, something that is addressed by the literature by configuring the models' class weighting parameter. However, adversarial example generation in a dataset with balanced classes remains difficult to create in a real-world environment and is positioned first in our priority list to test as soon as it becomes available. Additionally, the multi-classification experiments aggregated all types of attacks existing in the Bot-IoT dataset, rather than investigating the specific attacks individually. We also plan to investigate how different adversarial methods could be used for the Bot-IoT dataset's particular attacks. As our work did not cover manipulating labels with a high margin to the SVM hyperplane, we also want to explore further how these labels affect the SVM model. Finally, we want to implement and study the suggested countermeasures' effects on the adversarial machine learning methods investigated.

**Author Contributions:** All authors contributed in the conceptualization and methodology of the manuscript; O.T.v.E. performed the data preparation; P.P., N.P. and C.C. contributed in writing; A.M. and W.J.B. reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

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

## Appendix A. SVM Random Label Flip 5% Sample

**Listing A1.** SVM random label flip 5% sample.

```
n = 0.05 # 5%
change = training.sample(int(n*len(training))).index
# Use binary XOR to flip from 1 to 0 and vice versa
training.loc[change, 'attack'] ^= 1
```

## Appendix B. SVM Targeted Label Flip Function

**Listing A2.** SVM targeted label flip function.

```
def get_change(model, X_train, n):
    distances = model.decision_function(X_train)
    abs_values = np.abs(distances) # gen absolute distance
    df = pd.DataFrame(data={'distance': distances,
                            'abs_value': abs_values},
                      columns=["distance", "abs_value"])

    change = df.sort_values(
        by=['abs_value']).head(int(n*len(df))).index

    return change
```

## Appendix C. Create Logits Model

**Listing A3.** Create logits model.

```
import tensorflow as tf # Version 2.3.0
logits_model = tf.keras.Model(model.input,
                              model.layers[-1].output)
```

## Appendix D. Convert Testing Dataset into Tensor

**Listing A4.** Convert testing dataset into tensor.

```
original_data = X_test_scaled
original_data = tf.convert_to_tensor(original_data.reshape((len(X_test_scaled), 10)))
```

## Appendix E. Use Fast Gradient Sign Method to Generate Adversarial Examples

**Listing A5.** Use fast gradient sign method to generate adversarial examples.

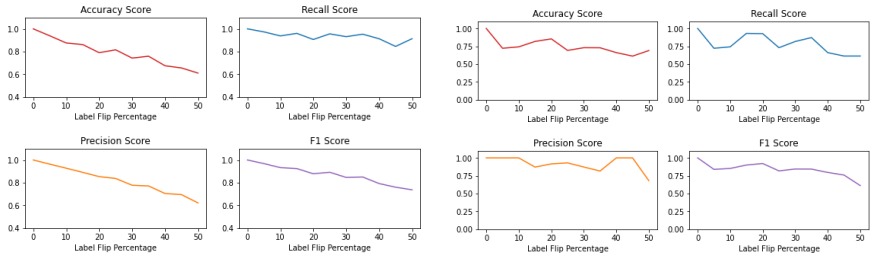
```
import cleverhans.future.tf2.attacks.fast_gradient_method

epsilon = 1
adv_data = fast_gradient_method(logits_model, original_data,
                                epsilon, np.inf,
                                targeted=False)

adv_data_pred = model.predict(adv_data)
```

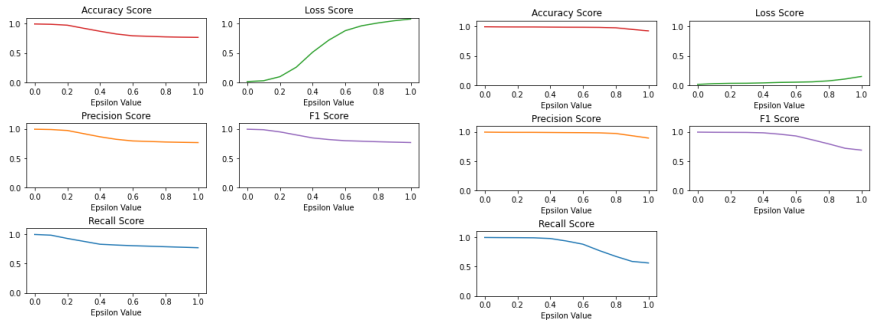
## Appendix F. Visualisation of Metrics

The metric results were visualised using Matplotlib Pyplot (Matplotlib: Visualization with Python: https://matplotlib.org/ (accessed on 22 April 2021)).



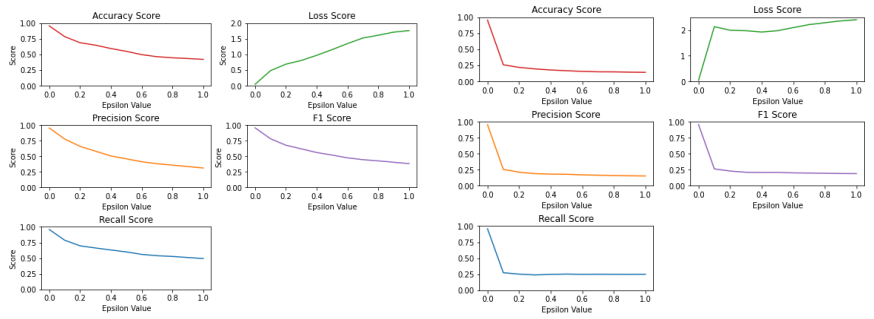

(**a**) (**b**)

**Figure A1.** (**a**) Targeted Support Vector Machines flip metrics. (**b**) Non-targeted Support Vector Machines flip metrics.




(**a**) (**b**)

**Figure A2.** (**a**) Binary classification metrics, non-targeted Fast Gradient Sign Method. (**b**) Binary classification metrics, targeted Fast Gradient Sign Method.




(**a**) (**b**)

**Figure A3.** (**a**) Multi-classification metrics, targeted Fast Gradient Sign Method. (**b**) Multi-classification metrics, non-targeted Fast Gradient Sign Method.

## Appendix G. Software and Hardware

Many tools can be used to process the Bot-IoT dataset. Initially, the Python programming language was selected due to the abundance of packages that have been developed, namely the Scikit-Learn package [49]. The focus of the Scikit-Learn package is to create an approachable interface while providing state-of-the-art machine learning algorithm implementations [50]. Conversely, Scikit-Learn has some limitations; for example, it may not be scalable in performance when considering large datasets. To combat the limitations of Scikit-Learn, TensorFlow has been used to spread the workload across many different cores [51]. TensorFlow is an open-source project developed by Google that is popular within the data science community for its flexibility in maximizing the use of the available hardware. TensorFlow's primary benefit is the inclusion of Keras, a library used to develop neural networks in Python, thus allowing data analysis using estimators [51].

As it can be seen in Figure A4, the system's configuration is provisioned through the Google Colaboratory (Colab) (Google Colaboratory: https://colab.research.google.com/

(accessed on 22 April 2021)) service. Google Colab is a cloud Software as a Service (SaaS) platform. The GPU provided in Google Colab is an Nvidia Tesla K80 which has 4992 CUDA cores. The high amount of CUDA cores will be beneficial for the machine learning model training using TensorFlow.

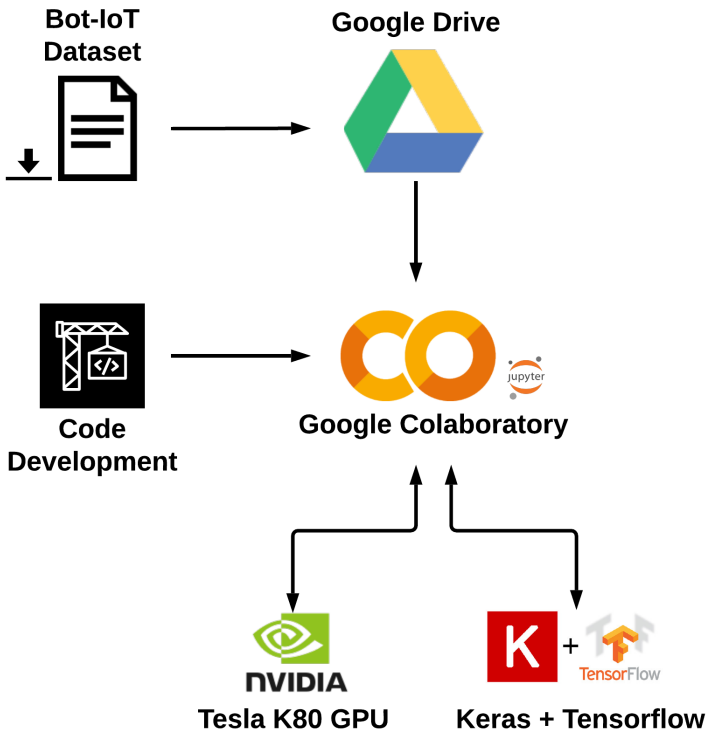



**Figure A4.** Testbed architecture.

## References


1. Sapre, S.; Ahmadi, P.; Islam, K. A Robust Comparison of the KDDCup99 and NSL-KDD IoT Network Intrusion Detection Datasets Through Various Machine Learning Algorithms. *arXiv* **2019**, arXiv:1912.13204.
2. Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U.; Gidlund, M. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4724–4734. [CrossRef]
3. Benkhelifa, E.; Welsh, T.; Hamouda, W. A critical review of practices and challenges in intrusion detection systems for IoT: Toward universal and resilient systems. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3496–3509. [CrossRef]
4. Goasduff, L. Gartner Predicts Outdoor Surveillance Cameras Will Be Largest Market for 5G Internet of Things Solutions Over Next Three Years. 2019. Available online: https://www.gartner.com/en/newsroom/press-releases/2019-10-17-gartner-predicts-outdoor-surveillance-cameras-will-be (accessed on 22 April 2021)
5. Ibitoye, O.; Shafiq, O.; Matrawy, A. Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; IEEE: New York, NY, USA, 2019; pp. 1–6.
6. Soe, Y.N.; Feng, Y.; Santosa, P.I.; Hartanto, R.; Sakurai, K. Towards a Lightweight Detection System for Cyber Attacks in the IoT Environment Using Corresponding Features. *Electronics* **2020**, *9*, 144. [CrossRef]
7. Elrawy, M.F.; Awad, A.I.; Hamed, H.F. Intrusion detection systems for IoT-based smart environments: a survey. *J. Cloud Comput.* **2018**, *7*, 1–20. [CrossRef]
8. Cup, K. Data. 1999. Available online: http://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data (accessed on 22 April 2021).
9. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; IEEE: New York, NY, USA, 2009; pp. 1–6.
10. Nisioti, A.; Mylonas, A.; Yoo, P.D.; Katos, V. From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3369–3388. [CrossRef]
11. 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]

12. Mahoney, M.V.; Chan, P.K. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *International Workshop on Recent Advances in Intrusion Detection*; Springer: Berlin, Germany, 2003; pp. 220–237.
13. Athanasiades, N.; Abler, R.; Levine, J.; Owen, H.; Riley, G. Intrusion detection testing and benchmarking methodologies. In Proceedings of the First IEEE International Workshop on Information Assurance, 2003. IWIAS 2003 Proceedings, Darmstadt, Germany, 24 March 2003; IEEE: New York, NY, USA, 2003; pp. 63–72.
14. 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]
15. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. [CrossRef]
16. Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2805–2824. [CrossRef]
17. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial machine learning at scale. *arXiv* **2016**, arXiv:1611.01236.
18. Pitropakis, N.; Panaousis, E.; Giannetsos, T.; Anastasiadis, E.; Loukas, G. A taxonomy and survey of attacks against machine learning. *Comput. Sci. Rev.* **2019**, *34*, 100199. [CrossRef]
19. Kantartopoulos, P.; Pitropakis, N.; Mylonas, A.; Kylilis, N. Exploring Adversarial Attacks and Defences for Fake Twitter Account Detection. *Technologies* **2020**, *8*, 64. [CrossRef]
20. Huang, L.; Joseph, A.D.; Nelson, B.; Rubinstein, B.I.; Tygar, J.D. Adversarial machine learning. In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, Chicago, IL, USA, 21 October 2011; pp. 43–58.
21. Xiao, H.; Biggio, B.; Nelson, B.; Xiao, H.; Eckert, C.; Roli, F. Support vector machines under adversarial label contamination. *Neurocomputing* **2015**, *160*, 53–62. [CrossRef]
22. Van, N.T.; Thinh, T.N. An anomaly-based network intrusion detection system using deep learning. In Proceedings of the 2017 International Conference on System Science and Engineering (ICSSE), Ho Chi Minh City, Vietnam, 21–23 July 2017; IEEE: New York, NY, USA, 2017; pp. 210–214.
23. Oh, S.R.; Kim, Y.G. Security requirements analysis for the IoT. In Proceedings of the 2017 International Conference on Platform Technology and Service (PlatCon), Busan, Korea, 13–15 February 2017; IEEE: New York, NY, USA, 2017; pp. 1–6.
24. Davis, J.; Goadrich, M. The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 233–240.
25. Flach, P.A. The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In Proceedings of the 20th International Conference on Machine Learning (ICML-03), Washington, DC, USA, 21–24 August 2003; pp. 194–201.
26. Wu, P.; Guo, H. LuNet: A Deep Neural Network for Network Intrusion Detection. In Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6–9 December 2019; IEEE: New York, NY, USA, 2019; pp. 617–624.
27. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks. *Electronics* **2019**, *8*, 1210. [CrossRef]
28. Atawodi, I.S. A Machine Learning Approach to Network Intrusion Detection System Using K Nearest Neighbor and Random Forest. Master's Thesis, University of Southern Mississippi, Hattiesburg, MS, USA, 2019; Volume 651.
29. Dong, B.; Wang, X. Comparison deep learning method to traditional methods using for network intrusion detection. In Proceedings of the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, China, 4–6 June 2016; IEEE: New York, NY, USA, 2016; pp. 581–585.
30. Fernandez, G. Deep Learning Approaches for Network Intrusion Detection. Ph.D. Thesis, The University of Texas at San Antonio, San Antonio, TX, USA, 2019.
31. Pacheco, Y.; Sun, W. Adversarial Machine Learning: A Comparative Study on Contemporary Intrusion Detection Datasets. In Proceedings of the 7th International Conference on Information Systems Security and Privacy—Volume 1: ICISSP, INSTICC, Online Streaming, Vienna, Austria, 11–13 February 2021; SciTePress: Setubal, Portugal, 2021; pp. 160–171. [CrossRef]
32. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; IEEE: New York, NY, USA, 2015; pp. 1–6.
33. Jazi, H.H.; Gonzalez, H.; Stakhanova, N.; Ghorbani, A.A. Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling. *Comput. Netw.* **2017**, *121*, 25–36. [CrossRef]
34. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the ICISSP, Madeira, Portugal, 22–24 January 2018; pp. 108–116.
35. Xiao, H.; Xiao, H.; Eckert, C. Adversarial Label Flips Attack on Support Vector Machines. In Proceedings of the ECAI, Montpellier, France, 27–31 August 2012; pp. 870–875.
36. Biggio, B.; Nelson, B.; Laskov, P. Support vector machines under adversarial label noise. In Proceedings of the Asian Conference on Machine Learning, Taoyuan, Taiwan, 13–15 November 2011; pp. 97–112.
37. Koh, P.W.; Steinhardt, J.; Liang, P. Stronger data poisoning attacks break data sanitization defenses. *arXiv* **2018**, arXiv:1811.00741.
38. Taheri, R.; Javidan, R.; Shojafar, M.; Pooranian, Z.; Miri, A.; Conti, M. On defending against label flipping attacks on malware detection systems. *Neural Comput. Appl.* **2020**, *32*, 1–20. [CrossRef]

39. Zhou, Y.; Kantarcioglu, M.; Thuraisingham, B.; Xi, B. Adversarial support vector machine learning. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012; pp. 1059–1067.
40. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2014**, arXiv:1412.6572.
41. Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The limitations of deep learning in adversarial settings. In Proceedings of the 2016 IEEE European symposium on security and privacy (EuroS&P), Saarbruecken, Germany, 21–24 March 2016; IEEE: New York, NY, USA, 2016; pp. 372–387.
42. Wiyatno, R.; Xu, A. Maximal jacobian-based saliency map attack. *arXiv* **2018**, arXiv:1808.07945.
43. Yang, K.; Liu, J.; Zhang, C.; Fang, Y. Adversarial examples against the deep learning based network intrusion detection systems. In Proceedings of the MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; IEEE: New York, NY, USA, 2018; pp. 559–564.
44. Wang, Z. Deep learning-based intrusion detection with adversaries. *IEEE Access* **2018**, *6*, 38367–38384. [CrossRef]
45. Rigaki, M. Adversarial Deep Learning Against Intrusion Detection Classifiers. 2017. Available online: http://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-64577 (accessed on 22 April 2021).
46. Jeong, J.; Kwon, S.; Hong, M.P.; Kwak, J.; Shon, T. Adversarial attack-based security vulnerability verification using deep learning library for multimedia video surveillance. *Multimed. Tools Appl.* **2019**, *79*, 16077–16091. [CrossRef]
47. 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; IEEE: New York, NY, USA, 2019; pp. 256–25609.
48. Papernot, N.; Faghri, F.; Carlini, N.; Goodfellow, I.; Feinman, R.; Kurakin, A.; Xie, C.; Sharma, Y.; Brown, T.; Roy, A.; et al. Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *arXiv* **2018**, arXiv:1610.00768.
49. Srinath, K. Python–The Fastest Growing Programming Language. *Int. Res. J. Eng. Technol.* **2017**, *4*, 354–357.
50. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
51. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation OSDI 16, Savannah, GA, USA, 2–4 November 2016; pp. 265–283.