IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.918 ⁜ Vol. 11, Issue 10, October 2022

DOI: 10.17148/IJARCCE.2022.111001

Utilizing Generative Adversarial Networks to develop a robust Defensive System against Adversarial Examples

Isaac Tumwine[1], Justin Nshunguye[2]

Department of Business Information Technology at University of Tourism,

Technology and Business Studies in Kigali Rwanda[1,2]

Email correspondence: isaactumwine11@gmail.com and nshungujustin@gmail.com

Abstract: Using the special ability of Generative Adversarial Networks (GANs) to create fresh adversarial instances for model retraining, we offer a novel defense strategy against adversarial examples in this study. In order to achieve this, we create an automated pipeline that combines a convolutional neural network that has already been trained with an external GAN called the Pix2Pix conditional GAN. This pipeline allows us to identify the transformations between adversarial examples and clean data as well as create new adversarial examples on the fly. In an iterative pipeline, these adversarial samples are used to strengthen the model, attack, and defense. Our simulation findings show that the suggested strategy works well.

Keywords: adversarial machine learning; botnet detection; generative adversarial networks; machine learning

1. INTRODUCTION

Deep learning networks, in particular, have proven to be capable tools in a wide range of challenging domains such as computer vision, healthcare, industry, finance, and so on. Machine learning applications, for example, are used in an increasing number of Internet of Things (IoT) devices for network traffic analysis and, in particular, botnet [1] detection (28.1 billion devices in 2020 and expected to reach trillions in 2025 [2]). Traditional machine learning-based botnet detection methods, such as statistical methods and behavioral techniques, rely on pattern extraction from empirical data using a set of selective features. However, for best feature engineering, these techniques necessitate expert domain knowledge. Recently, researchers in the community proposed representation learning as a technique for automatically learning representative features from raw data as a way to overcome the limitations of traditional machine learning methods. Convolutional Neural Networks (CNNs) are a well-known method for representation learning in which network traffic data can be fed to the model in either numeric or image format [3]. The latter is known as a visualization-based botnet detection approach, which is used for understanding data characteristics and visualizing embedded features [3, 4]. These deep learning models can not only learn the representative properties of network traffic data, but they can also predict the label of future network traffic data in an automated manner.

The dependability of their findings is essential given the extensive use and widespread adoption of neural networks in delicate fields. Although hostile entities are now encouraged to alter the inference of these learning models so that the output is incorrect due to the widespread use of deep learning models [5]. Deep learning algorithms have been shown to be prone to drawing conclusions from altered input data in recent years [6, 7]. Deep learning networks would be driven to misclassify the input and be deceived into providing incorrect outputs when provided with inputs containing minimal perturbation, i.e., adversarial examples [6–9]. A minor but properly planned disruption to the input dataset would have a negative effect on the model's output since deep learning methods understand the intrinsic pattern of the input dataset. The generated adversarial instances in the adversarial machine learning paradigm closely resemble the original ones and are not always the results of the input data distribution. Consequently, identifying hostile examples is a difficult process.

There are various techniques available to produce adversarial instances automatically [10]. Although some adversarial example generating algorithms work utilizing evolutionary-based optimization techniques, the majority of them are based on the gradient analysis of the model output in relation to the input [11]. For instance, Goodfellow et alFast.'s Gradient Sign Method (FGSM) [12] creates adversarial samples based purely on the gradient's sign. Kurakin et al. devised the Basic Iterative Approach (BIM) [13], which effectively repeats the FGSM approach through a number of stages. Moosavi-Dezfooli et al. created DeepFool [8] to find the shortest path between an input and the decision boundary of adversarial samples.

The Jacobian-based Saliency Map Assault (JSMA) [6] was developed by Paper not et al. It aims to generate an adversarial attack by perturbing a restricted range of input features. Contrary to popular belief, adversarial learning attacks are currently being aggressively applied to other delicate domains, despite their origins in the image domain. Grosse et al. [14] built on the existing adversarial example generating techniques to create a very effective attack that leverages adversarial examples against malware detection technologies. Utilizing adversarial scenarios, Osadchy et al. [15] created a secure CAPTCHA algorithm. There has been a lot of interest in developing defensive systems to increase the robustness of deep learning models as a result of the sensitivity of deep learning networks to adversarial examples [12, 16–18]. With the exception of the strategy of retraining, it has recently been demonstrated that nearly all of the suggested defenses against adversaries may be defeated [19]. Keep in mind that any defense mechanism must not only be effective against adversarial attacks that are already underway, but also perform well against attacks that are yet to occur and are not yet anticipated within the given threat model. Therefore, any defensive plan should be flexible.

In this paper, we offer a novel defense system that creates adversarial training instances using generative adversarial networks (GANs) [20]. We use an external GAN, namely Pix2Pix [21] conditional GAN, to attack the deep learning network automatically. This allows us to comprehend the transformations between adversarial instances and clean data and to automatically produce unseen adversarial examples. After attacking the neural network, we automatically create a large number of adversarial samples and utilize them to counterattack attackers. By doing this, we create a defense mechanism that is useful in real-world applications and uses a deep learning network that has already been trained.

The greatest response is to use an iterative offensive strategy to produce new attacks to assist build the neural network because the vulnerability against adversaries is inherent to the universe of neural networks.In particular, the following structure describes the primary contributions of this work:

• Creating a novel (attacking) approach to the production of adversarial instances that learns the initial data distribution of typical adversarial examples and modifies it to deceive a deep learning model that has already been trained.

• Producing fresh adversarial examples that models trained on the original common hostile examples may ignore.

• Using a previously provides the review technological accomplishment, attaching a pre-trained CNN to a Pix2Pix GAN and learning the generation with the intention of fooling the attached network.

• Running thorough tests to assess the effectiveness of the suggested approach and showcasing a use for visualization-based botnet detection systems.

The remainder of the essay is structured as follows: Section 2 discusses the introduction and preliminary information. Section 3 presents the comprehensive defense we created against adversarial attacks by utilizing the strength of generative adversarial networks. In Section 4, we assess the effectiveness of the suggested approach and go over the findings. The final section, Section 5, contains the conclusion and future work.

2. BACKGROUND AND PRELIMINARIES

To create an automated defense system against hostile attacks, we conduct a thorough investigation in this work at the nexus of cybersecurity, artificial intelligence, and computer vision.

We discuss prior knowledge that is necessary as well as basic ideas in this part.

2.1 Network Traffic Data Visualization

Wang et al. has presented visualization-based network traffic data analysis [22]. The three steps in the suggested method traffic splitting, traffic clearing, and image generation convert network traffic data into grayscale images.

These phases are explained as follows:
Traffic split: This stage is responsible for dividing packet capture files into numerous distinct traffic units, either binary or packet capture for flow-based layers. (2) Traffic clear: In this stage, media access control addresses and Internet Protocol (IP) addresses are randomized in the data link layer and IP layer, respectively, to provide anonymization and sanitization. Additionally, empty or redundant data are deleted. (3) Image generation: In this step, outliers—traffic data that are noticeably larger than the rest of the dataare eliminated. The final step is to lengthen the remaining data so that it may be converted into grayscale images.

2.2 Adversarial Attacks in Deep Learning

The creation of adversarial instances is guided by a threat model, which also specifies the attacker's capabilities. The threat model is defined and described in this work in the following ways:

1. After training is complete, the opponents can only launch an attack during the test stage. As a result, training data contamination won't be investigated.
2. The model being attacked in this work is a deep convolutional neural network. Although classic machine learning models like Random Forest (RF) [24] and Support Vector Machines (SVM) [23] can also produce results with high accuracy, it has been demonstrated that adversarial examples from deep neural networks outperform these traditional models [25,26].
3. Enemies want to undermine integrity, which is shown by one of the performance indicators like accuracy or F1-score.
4. Adversarial falsehood is the attacker's aim. The assaults vary depending on whether the opponent is attempting to trick the model into misclassifying input as positive or negative. When a botnet is detected, the adversary can attempt to change its classification from malicious to benign in order to launch a successful attack or to change its classification from benign to malicious in order to have unfavorable effects on regular data.
5. In this investigation, we make the assumption that the attacker conducts repeated, iterative attacks.
6. The opponent is fully aware of the internals and model structure, which makes white-box attacks possible.

Based on the outline of our threat model, it lies in relatively easy attacks, where any attacker can design an attack to make an adverse impact on the output of deep learning models. Note that building an effective system to defend against such prevalent attacks would immune deep learning modelsagainst a huge number of potential threats, which is the main goal of this study.

2.3 Generative Adversarial Networks

A type of machine learning models known as generative adversarial networks (GEN) consists of two neural networks: a discriminator network (D) and a generator network (G) [20, 27]. GANs are able to synthesize data points that are comparable to those in the original data distribution thanks to this structure. These networks are capable of creating artificial visuals that appear flawless to humans [28], creating music [29], and even creating new molecules [30] The ability to generate examples can also be utilized to create examples of opposition.

The task of the generative network is to produce synthetic samples that are perceptually compelling and that seem to have come from a real data distribution Pdata. The generator creates a new synthetic sample, G(z), with the same dimensions as the real sample using a noise vector z from a distribution of Pz. The discriminator, on the other hand, is a binary classifier that determines whether a sample is real or fake based on input from both the actual and synthesized samples. The following optimization problem (1), where the performance of the discriminator is directly correlated with the generator's loss, is typically solved as part of a GAN's training process.

$$\min_G \max_D V(D,G) = \underset{x\sim P_{\text{data}}}{E}[\log D(\mathbf{x})] + \underset{z\sim P_z}{E}[\log(1 - D(G(\mathbf{z})))] \tag{1}$$

Here, V (D, G) is the objective function, P_{data} is real data distribution, D(x) denotes the probability that D discriminates x as real data, P_z is noise distribution, G(z) is the sample generated by the generator, and D(G(z)) indicates the probability that D determines the sample created by generatorG(z). While the discriminator tries to maximize its cost function through minimizing prediction errors, the generator tries to minimize its cost function through generating samples that are not detectable by the discriminator. More detailed information about GANs can be found in References [20, 27].

3. METHODOLOGY

This section outlines the specifics of our suggested attack and defense mechanism. The suggested solution, as shown in Figure 1, is made up of four key parts: a DL-based botnet detector, adversarial attacks based on gradients, GANs, and defensive mechanisms.

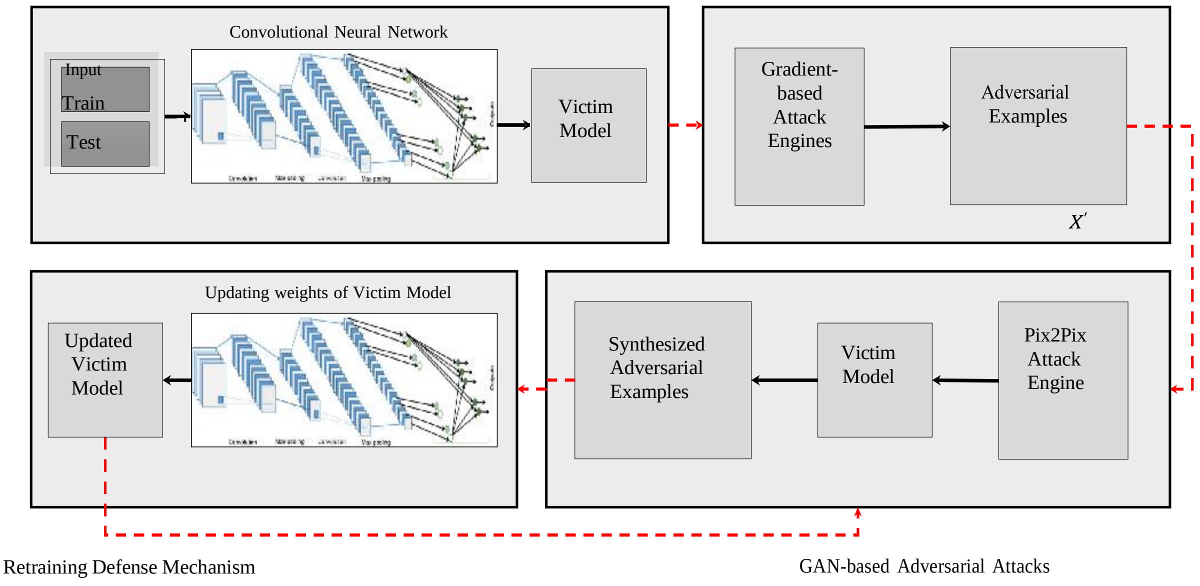

Convolutional Neural Network
Input
Train
Test
Victim Model
Gradient-based Attack Engines
Adversarial Examples
X′
Updating weights of Victim Model
Updated Victim Model
Synthesized Adversarial Examples
Victim Model
Pix2Pix Attack Engine
Retraining Defense Mechanism
GAN-based Adversarial Attacks

Figure 1 depicts the overall layout of our suggested attack and protection system. It is made up of four key parts: a DL-based botnet detector, adversarial assaults using gradients and GANs, adversarial attacks using gradients and GANs, and a defense mechanism based on retraining victim models.

3.1. Victim Model

One of the well-known representation learning approaches, convolutional neural networks are widely used in a variety of sensitive fields, from computer vision to visualization-based botnet detection. In this investigation, our baseline botnet detection model was a convolutional neural network. Assuming that this baseline model is the target of adversarial attacks, we refer to it as the victim model. Remember that the major objective of this work is to create a defensive system to increase the resilience of this model against adversarial attacks by utilizing the special power of generative adversarial attacks, as we will discuss in more depth in the following sections. Network traffic information is transformed into gray-scale photos and used to create our victim model. Whether the data is related to legitimate traffic or malicious behavior, these images provide information that is typical of the nature of the traffic. Two sequential convolutional layers serve as the feature extractor in our botnet detector, while a fully connected layer serves as the classifier.

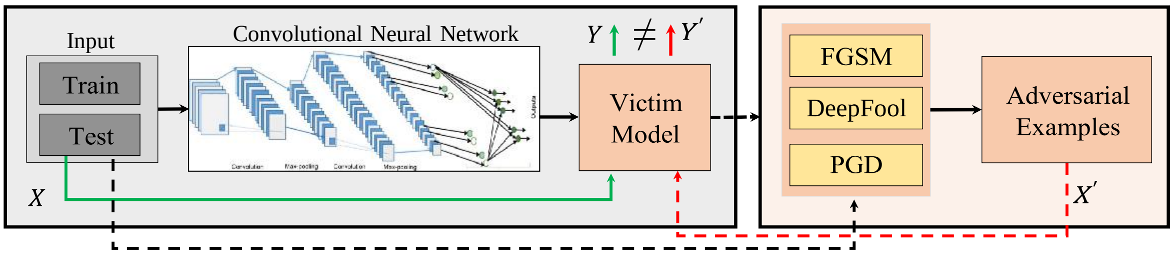
3.2. Attack Engines

We discussed our attack engines in this part, which were used to create adversarial instances to target our victim model. First, we created adversarial instances using general gradient-based attack engines. These adversarial attack techniques, which will be covered in Section 3.2.1, include the rapid gradient sign method [12], DeepFool [8], and projected gradient descent. Second, we used a unique generative adversarial network, which will be covered in Section 3.2.2, that can produce an infinite number of adversarial samples.

3.2.1. Gradient-Based Attack Engine

Researchers have recently developed a number of adversarial attack techniques for deep learning models. The majority of these assault strategies rely on gradient descent to function. The DL-based botnet detection system was subjected to adversarial attacks in this work using three of the most popular attack engines, including FGSM, DeepFool, and PGD. Figure 2 depicts the general layout of our gradient-based adversarial assaults on DL-based botnet detection systems.

Following a brief description of each of these assault engines, we direct readers who are interested to the original articles for more details. DL-based Botnet DetectorGeneric Example Generator



Attack Engine based on Gradients

- Figure 2 shows the general organization of the generic adversarial assault on botnet detection systems based on deep learning. In this study, we used the Fast Gradient Sign Method (FGSM), DeepFool, and PGD as three popular gradient-based adversarial assaults on our DL-based botnet detector (victim model).

- FGSM: Developed by Goodfellow et al. in 2015 [12]. FGSM is a quick approach that updates each feature in the direction of the gradient's sign. Back-propagation can be used to produce this perturbation.

- This assault is frequently used in real-world circumstances due to its speed. FGSM may be created by utilizing (2).

$$X_{\text{Adv}} = X + \varepsilon \cdot \text{sign}\,(\nabla_X J(X, Y)) \quad (2)$$

Here, X is the clean sample, X_{Adv} is the adversarial example, J is the classification loss, Y is the label of the clean sample, and ε is a tunable parameter that controls the magnitude of the perturbation. Note that, in FGSM only direction of the gradient is important not its magnitude.

- **DeepFool:** This attack is introduced by Moosavi-Dezfooli et al. [8] as an untargeted iterative attack based on the L_2 distance metric. In this attack the closest distance from the clean input tothe decision boundary is found. Decision boundaries are the boundaries that divide different classes in the hyper-plane created by the classifier. Perturbations are created in a manner thatpushes the adversarial example outside of the boundary, causing it to be misclassified as another class, as demonstrated in Algorithm 1.

•1.

Algorithm 1: The process of generating adversarial examples based on Deep Fool Method [8].

```
1 input: Image $x$, classifier $f$
2 output: Perturbation $\hat{r}$
3 Initialize $x_0 \leftarrow x$, $i \leftarrow 0$
4  while sign $(f(x_i))$ = sign $(f(x_0))$ do
5    $r_i \leftarrow -\frac{f(x_i)}{\|\nabla f(x_i)\|_2^2} \nabla f(x_i)$
6    $x_{i+1} \leftarrow x_i + r_i$
7    $i \leftarrow i + 1$
8 end while
9 return $\hat{r} = \sum_i r_i$
```

DOI: 10.17148/IJARCCE.2022.111001

- **PGD:** This attack is proposed by Madry et al. [31] as an iterative adversarial attack that creates adversarial examples based on applying FGSM on a data point x_0, in an iterative manner, that is obtained by adding a random perturbation of magnitude α to the original input x. Then the perturbed output is projected to a valid constrained space. The projection is conducted by finding the closet point to the current point within a feasible region. This attack can be formulated based on the following equation.

$$x^{i+1} = \text{Proj}_{x+S} \quad x^i + \alpha\,\text{sign} \quad \nabla_{xi} J \quad \theta, x^i, y \qquad , \tag{3}$$

where x^{i+1} is the perturbed input at iteration $i + 1$ and S denotes the set of feasible perturbations for x.

3.2.2. GAN-Based Attack Engine

Generative adversarial networks are capable of creating artificial visuals that are flawless to the human eye [28], creating music [29], and even creating new molecules [30]. In this study, we make use of Pix2Pix, a conditional generative adversarial network that was created exclusively for the image domain [21]. Conditional GANs provide additional data, such as the source image. So, a conditional GAN's loss function can be calculated as (4).

$$\text{L}_{cGAN}(G.D) = E_{x\sim p_{\text{data}}(x)}[\log(D(x.y))] + E_{z\sim p_z(z)}[\log(1 - D(G(z.y).y))]. \tag{4}$$

This fact makes Pix2Pix a perfect fit for the image classification and botnet detection domain to be used to understand the transformations between normal data and malicious data and generating unlimited amounts of new adversarial examples. In Pix2Pix, the generator is build based on the U-net structure, while the discriminator has built based on Patch GAN architecture. We refer the interested readers to the source paper for more information about Pix2Pix structure [21].

Pix2Pix is a type of conditional GAN employs a loss function to train the mapping from input image to output image. This adversarial loss function, shown in (5), forces the generator to create a sample that resembles the conditioning variable x. Finally, this adversarial loss function is added to (4).

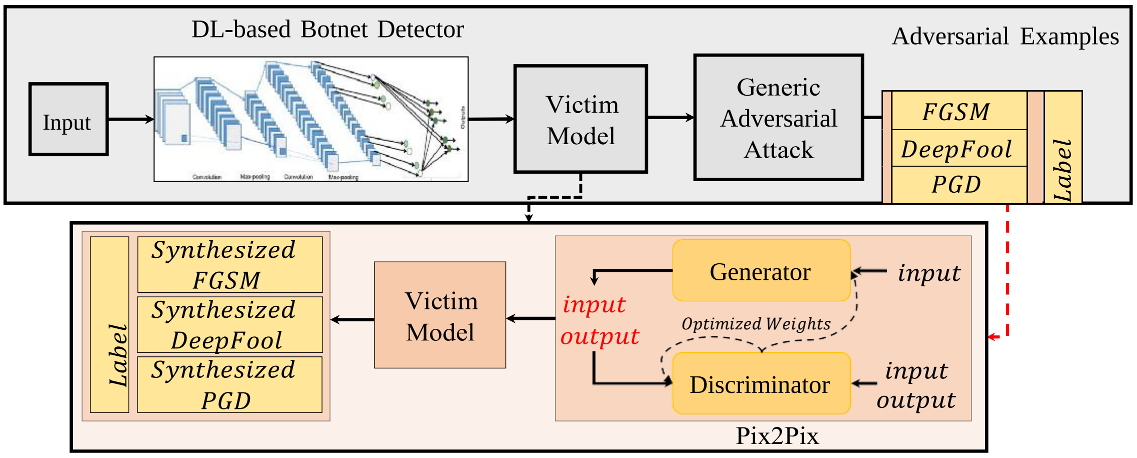
$$\text{L}_{L1}(G) = E_{x,y,z}\,[\|x - G(z.y)\|_1]\,. \tag{5}$$

The loss function in this work is as (6).

$$\text{L}(G.D) = \text{L}_{cGAN}(G.D) + \lambda \text{L}_{L1}(G), \tag{6}$$

Where λ is a hyper-parameter that controls the weight of the term.

The general architecture of our Pix2Pix-based adversarial example generator is shown in Figure 3. After the transformations are fond, any given clean data from the dataset can be transformed into a malicious data. In other words, the Pix2Pix learns how to add perturbation to the data in an automated manner. This automated attacking is leveraged, and a huge corpus of new attack data is generated. To enable the Pix2Pix to generate better attacks, each generated image is fed to the victim model and is tested to see if it fools the model or not. This can be formulated via adding a loss function for the attack on the model, which can help the training of the Pix2Pix. By doing so, the Pix2Pix autonomously attacks the model using the understanding of the perturbation distribution and is able to generate huge corpus of unseen adversarial example



GAN-based Attack Engine

Figure 3 shows the general organization of adversarial GAN-based assaults against DL-based botnet detection systems. Here, Pix2Pix was used to create new hostile examples that are comparable to the ones created in Section 3.2.

3.3. Defense Mechanism

We can use the retraining strategy to protect against hostile cases that have been generated in a significant number through automated means. Retraining entails instructing the victim model once more using the adversarial data. By doing this, we can inform the neural networks that some of their inputs are incorrect. By enhancing its decision limits, the neural network then learns how to prevent these errors, as depicted in Figure 4. When other protections, like a change in structure, have failed, it has been demonstrated that this technique can withstand adversarial attempts.

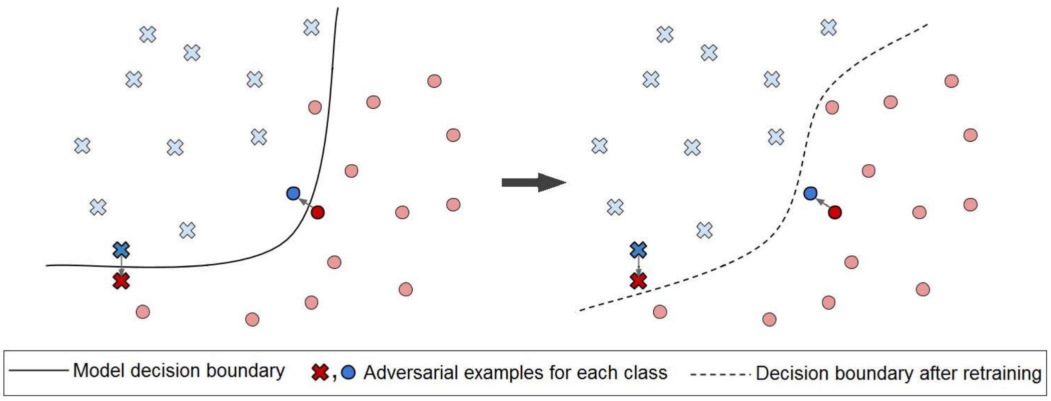

Model decision boundary
Adversarial examples for each class
Decision boundary after retraining

Figure 4. In retraining the defensive approach the victim model learns how to avoid mistakes byimproving its decision boundaries.

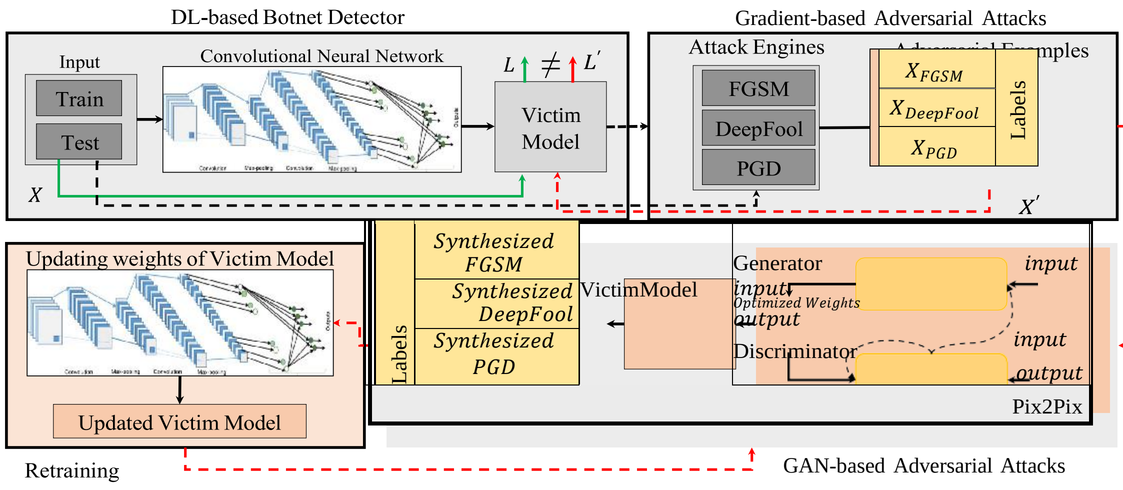


Figure 5. General architecture of our proposed attack and defense system. In the retraining process the weights of the victim model are being updated in each iteration, thus improves its decision boundaries. In return, this process also yields to more powerful adversarial examples after each iteration.

The neural network will develop over time as a result of this iterative process. The effect of this defense on neural networks in the security area may be substantial. This method enables fresh adversarial samples to be created for a particular pre-trained neural network on each iteration, as in the case of malware detection. This model, which in real life detects malware, is then retrained using the fresh attack data, making it more resistant to them. When a particular level of accuracy is reached, the task is repeated and the model is updated.

4. EVALUATION AND DISCUSSION

We evaluate the performance of the proposed method for both generating new adversarial examples and defending against them through comprehensive experiments. In the following, first we give a brief description about utilized dataset, experimental setup, and evaluation metrics in Sections 4.1–4.3, respectively. Finally, the obtained results are discussed in Section 4.4.

4.1. Dataset

We used the CTU-13 dataset [32], which is a dataset of botnet, regular, and background traffic with corresponding labels, to assess the performance of the suggested technique. We solely used data from regular and botnet traffic in this analysis. The distribution of the normal and botnet samples used in this study is shown in Table 1. To carry out our studies, we randomly chose a smaller portion of the train and test data while keeping the original distribution, comprising 50,000 samples as train samples and 10,000 samples as test data. Based on the method outlined in Section 2.1, all train and test data are transformed into gray-scale pictures.

Table 1. Distribution of train and test samples across normal and botnet classes.

	Train	Test
Normal	34,144	**7376**
Botnet	15,856	**2624**
Total	**50,000**	**10,000**

4.2. Experimental Setup

In the paragraphs that follow, we go into great depth on the experimental setup that we used in this work in order to ensure the reproducibility of our suggested procedure.

Implementation. To create our attack strategies, we used the Python module Cleverhans [33], which is designed to evaluate whether machine learning systems are vulnerable to adversarial situations. It serves as a repository for adversarial cases that can be used to design defenses, plan attacks, and benchmark [33].

Parameter. The suggested strategy includes multiple options for both assault and defense scenarios.
To get the desired results—a larger misclassification rate in the assault phase and a lower fooling rate in the defense phase—multiple tests were run with various values for each parameter. For instance, the two parameters e and number of iterations that need to be modified in PGD were set to 0.15 and 10, respectively. With this setting we were able to achieve a fooling rate of 99.98%.

Evaluation System All of the experiments are conducted on a Lambda Quad deep learning workstation with Ubuntu 18.04 OS, Intel Xeon E5-1650 v4 CPU, 64 GB DDR4 RAM, 2TB SSD, 4TB HDD, and 4 NVIDIA Titan-V Graphics Processing Units (GPUs).

4.3. Evaluation Metrics

A set of appropriate performance measures must be created in order to determine how well the attacks and defense mechanisms are working. These metrics, which are accuracy or F1 score in classification tasks, are described in paragraphs (7) and (8), respectively. Although accuracy score is a required metric in the adversarial example synthesis domain, we still require additional assessment metrics to assess the efficacy of the synthesized instances and their fooling rate. By computing the distance metric as the average of their L2 norm distance between the normalized form of clean data and the synthesized data, the quality of generated adversarial examples is evaluated. Furthermore, the number of samples that are incorrectly labeled is thought to constitute the fooling rate.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

$$\text{F1 Score} = \frac{2TP}{2TP + FP + FN} \tag{8}$$

Where True Positive (TP) and True Negative (TN) are correctly identified botnet and normal samples, respectively. While False Positive (FP) is normal samples that are classified as botnet, False Negative(FN) is botnet samples that are classified as normal. These metrics are used to create the confusion matrix, as shown in Table 2.

Table 2. The confusion matrix for botnet detection.

		True Label	
		Botnet	**Normal**
Predicted Label	**Botnet**	True Positive (TP)	False Positive (FP)
	Normal	False Negative (FN)	True Negative (TN)

4.4. Results

For the purpose of evaluating the proposed approach's potential for producing fresh hostile cases and fending them off, we offer our tests using the above-described systematic methodologies in this part. The CTU-13 dataset [32], which is a dataset containing labels for botnet, regular, and background traffic, is the basis for all of our research. We used only the botnet and the regular traffic statistics for our analysis. Using the method outlined in Section 2.1, the gathered set is converted into gray-scale images.

Victim Model Performance. As stated in Section 3.1, our victim model is constructed using the generated photos and a convolutional neural network architecture. Table 3 displays the evaluation outcomes of these experiments. This table shows that our model can attain an F1 score of 99.98% and an accuracy rate of 99.99%. Please take note that we will utilize this model as our victim model (baseline) model to produce adversarial examples and counter them.

Table 3. Analysis of confusion matrix of victim model in classification of botnet and normal traffic data.

Confusion Matrix				Accuracy Rate (%)	F1 Score (%)
		Predicted Label			
		Botnet	Normal	99.99	99.98
True Label	Botnet	TP = 2624	FN = 0		
	Normal	FP = 1	TN = 7375		

Gradient-based Adversarial Attacks Three well-known adversarial approaches were utilized in our tests to test the resistance of our victim model to gradient-based adversarial attacks. Our results show that all FGSM, DeepFool, and PGD techniques have a high success rate for deceiving the classifier. Table 4 displays the outcomes that were attained for these tests. For instance, PGD outperforms the other two attacks with a fooling rate of 99.98% and an average distortion rate of 18.93%. We assume that the DeepFool's reduced fooling rate is a result of its underlying assumptions, which hold that classifiers should be thought of as linear with hyperplanes dividing each class from the others.

Table 4. Fooling and average distortion rate of each adversarial attack method.

Attack	Fooling Rate (%)	Distortion Rate (%)
FGSM	99.69	39.73
DeepFool	67.73	43.93
PGD	99.98	18.93

GAN-based Iterative Attack and Defense. The basic objective of the GAN-based iterative attack and defense is to generate increasingly powerful adversarial instances while simultaneously increasing the robustness of the model by creating a large number of adversarial examples utilizing gradient-based attack methods. Utilizing all three gradient-based adversarial approaches, we assessed the effectiveness of this attack and defense method. In a nutshell, we create synthetic adversarial cases that are similar to the gradient-based strategy to retrain the victim model, and then assess the victim model's robustness against the original adversarial examples that the victim model does not witness. By avoiding actually providing gradient-based adversarial instances to the victim model, we are able to increase the victim model's resistance to them.

FGSM Figure 6 shows the effectiveness of the suggested strategy in producing more powerful adversarial instances, which are then used to enhance the victim model's defense against hostile examples based on FGSM. As can be observed, it was able to synthesis additional samples that are deceiving the victim model with each iteration. On the other hand, retraining the model using these synthetic instances enhances the victim model's decision boundaries as expected, resulting in a fooling rate drop from 673 to 237 samples only after five iterations.

DeepFool. Figure 7 displays the outcomes for an iterative attack and defense using DeepFool. Although the DeepFool technique only had a fooling rate of 67.73 percent, the GAN-based approach can provide an equal number of effective synthesized adversarial samples as FGSM. This is encouraging since it shows that our GAN-based technique may provide fresh and powerful adversarial instances with even less samples. The victim model that has been retrained has also enhanced in robustness.

PGD The obtained results for iterative attack and defense based on PGD is shown in Figure 8. The obtained results from our experiments demonstrates a similar trend to that of both FGSM and DeepFool methods.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.918 ⁜ Vol. 11, Issue 10, October 2022

DOI: 10.17148/IJARCCE.2022.111001

We show our collected results in a normalized way in Figure 9 to help you comprehend the findings. This graphic illustrates how the FGSM and PGD techniques exhibit remarkably similar patterns during both the offensive and defensive phases. While the DeepFool technique demonstrates less success in the defense phase, it performs substantially better in the attack phase. Additionally, the trend demonstrates that even while the quantity of fresh fooling samples rises after each iteration, the system's defenses become saturated after a limited number of repetitions. This is because the distribution of the artificially created hostile samples alters with each iteration. The victim model begins to learn other distributions in addition to the initial distribution as a result.

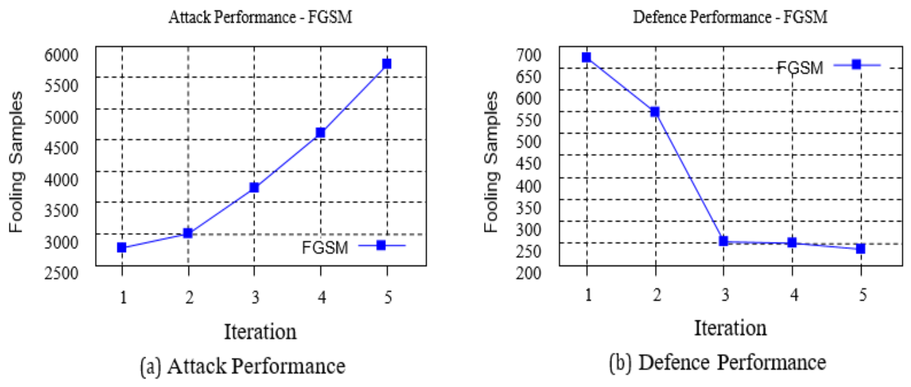


Figure 6. The results of FGSM.

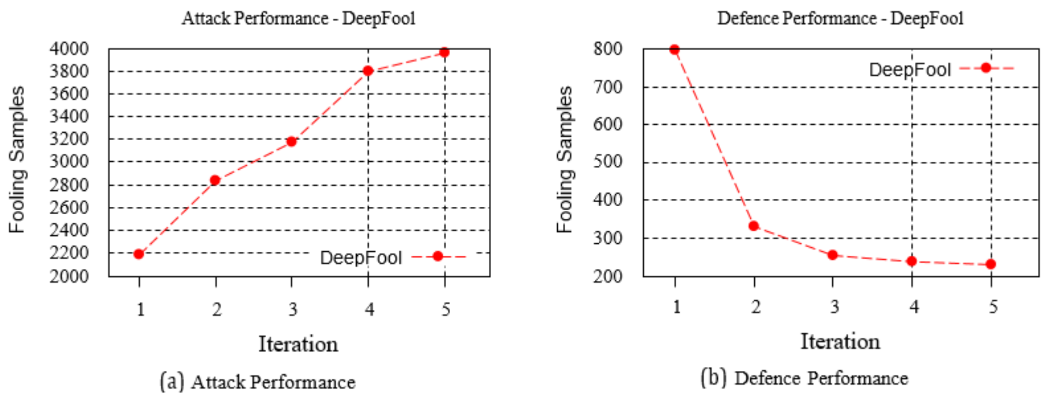


Figure 7. The results of DeepFool.

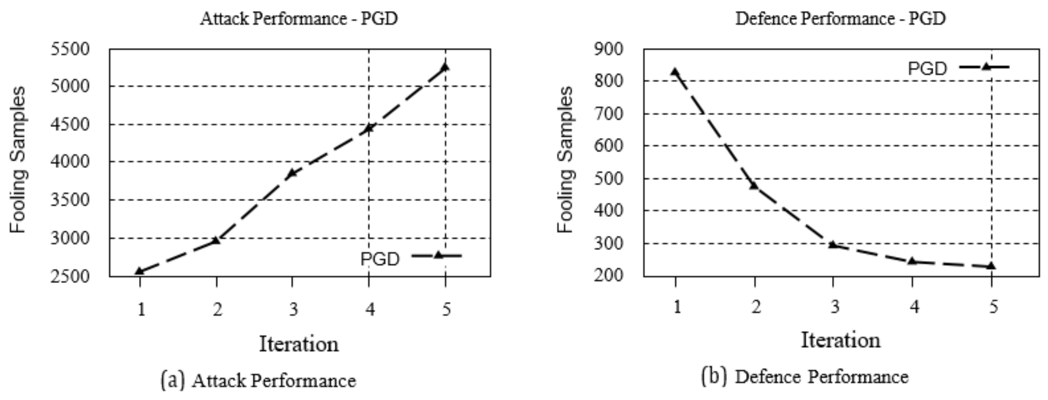


Figure 8. Our simulation results based on PGD attack method. While (**a**) present's attack performance, (b) Shows the success of our proposed defensive system for five iterations.

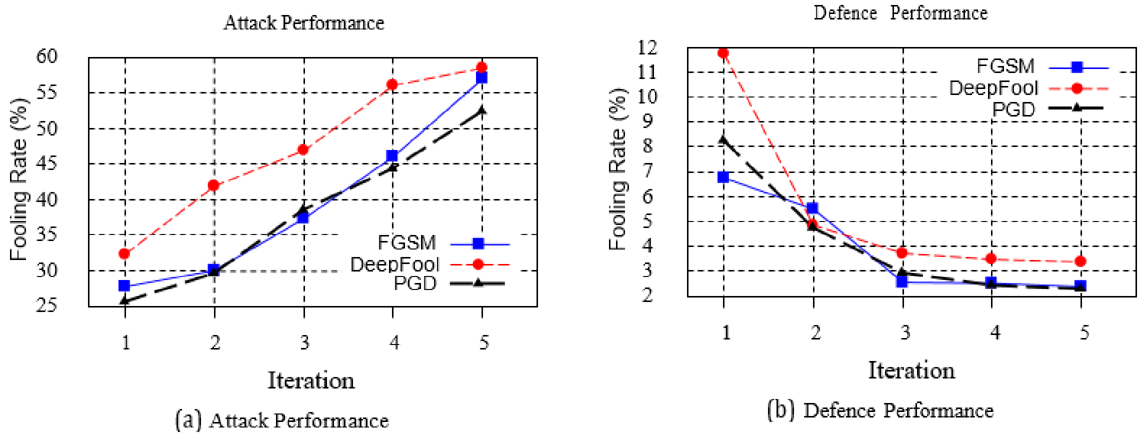


Figure 9. Overall simulation results for all three attack types. We observed similar trends for all three attack methods in both attack and defence scenarios.

5. CONCLUSIONS

Neural networks are frequently used in a variety of disciplines, particularly sensitive sectors, because of their strong learning capabilities. Through the use of adversarial examples, it has been demonstrated that hostile entities can control how the model is inferred. Although there are many defensive strategies, it has been demonstrated that the majority of them can be evaded. As a result, we present a novel defense system in this study that makes use of the special ability of GANs to provide adversarial cases for retraining. We use a conditional external GAN called Pix2Pix to comprehend the transformations between adversarial examples and clean data and to automatically generate unseen adversarial instances as part of an automated attack on the neural network. After attacking the neural network, we automate protection against adversaries by building a large number of adversarial samples in an iterative fashion. By doing this, we create a protection mechanism that works well with neural networks that have already been trained in the actual world. We compare the effectiveness of our created method to systems that detect botnets using visualization. Our findings show that our suggested approach works well. For instance, after just five rounds, the PGD method's output of tricking hostile examples drops from 824 to 226 samples.

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

REFERENCES

[1] Silva, S.S.; Silva, R.M.; Pinto, R.C.; Salles, R.M. Botnets: A survey. *Comput. Netw.* **2013**, *57*, 378–403.

[2] Manyika, J.; Chui, M.; Bisson, P.; Woetzel, J.; Dobbs, R.; Bughin, J.; Aharon, D. *Unlocking the Potential of the Internet of Things*; McKinsey Global Institute: Washington, DC, USA, 2015.

[3] Taheri, S.; Salem, M.; Yuan, J.S. Leveraging Image Representation of Network Traffic Data and Transfer Learning in Botnet Detection. *Big Data Cogn. Comput.* **2018**, *2*, 37.

[4] Vinayakumar, R.; Alazab, M.; Srinivasan, S.; Pham, Q.V.; Padannayil, S.K.; Simran, K. A Visualized Botnet Detection System based Deep Learning for the Internet of Things Networks of Smart Cities. *IEEE Trans. Ind. Appl.* **2020**.

[5] Chen, B.; Ren, Z.; Yu, C.; Hussain, I.; Liu, J. Adversarial examples for CNN-based malware detectors. *IEEE Access* **2019**, *7*, 54360–54371.

[6] 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), San Sebastian, Spain, 7–8 July 2016; pp. 372–387.

[7] Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z.B.; Swami, A. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, Abu Dhabi, UAE, 2–6 April 2017; pp. 506–519.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.918 ⁜ Vol. 11, Issue 10, October 2022

DOI: 10.17148/IJARCCE.2022.111001

[8] Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. Deepfool: A simple and accurate method to fool deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2574–2582.

[9] Wang, B.; Yao, Y.; Viswanath, B.; Zheng, H.; Zhao, B.Y. With great training comes great vulnerability: Practical attacks against transfer learning. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1281–1297.

[10] 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.

[11] Su, J.; Vargas, D.V.; Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 828–841.

[12] Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. In Proceedings of the International Conference on Learning Representation ICLR, San Diego, CA, USA, 7–9 May 2015.

[13] Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial examples in the physical world. *arXiv* **2016**, arXiv:1607.02533.

[14] Grosse, K.; Papernot, N.; Manoharan, P.; Backes, M.; McDaniel, P. Adversarial examples for malware detection. In Proceedings of the European Symposium on Research in Computer Security, Oslo, Norway, 11–15 September 2017; pp. 62–79.

[15] Osadchy, M.; Hernandez-Castro, J.; Gibson, S.; Dunkelman, O.; Pérez-Cabo, D. No bot expects the DeepCAPTCHA! Introducing immutable adversarial examples, with applications to CAPTCHA generation. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2640–2653.

[16] Buckman, J.; Roy, A.; Raffel, C.; Goodfellow, I. Thermometer encoding: One hot way to resist adversarial examples. In Proceedings of the International Conference on Learning Representation ICLR, Vancouver, BC, Canada, 30 April–3 May 2018 .

[17] Guo, C.; Rana, M.; Cisse, M.; Van Der Maaten, L. Countering adversarial images using input transformations. In Proceedings of the International Conference on Learning Representation ICLR, Vancouver, BC, Canada, 30 April–3 May 2018 .

[18] Song, Y.; Kim, T.; Nowozin, S.; Ermon, S.; Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In Proceedings of the International Conference on Learning Representation ICLR, Vancouver, BC, Canada, 30 April–3 May 2018 .

[19] Athalye, A.; Carlini, N.; Wagner, D. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In Proceedings of the 35th International Conference on Machine Learning, Vienna, Austria, 25–31 July 2018; Volume 80, pp. 274–283.

[20] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y.Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.

[21] Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–28 July 2017; pp. 1125–1134.

[22] Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware traffic classification using convolutional neural network for representation learning. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; pp. 712–717.

[23] Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297.

[24] Ho, T.K. Random decision forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; Volume 1, pp. 278–282.

[25] Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks?In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3320–3328.

[26] Dong, Y.; Pang, T.; Su, H.; Zhu, J. Evading defenses to transferable adversarial examples by translation-

invariant attacks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 July 2019; pp. 4312–4321.

[27] Pan, Z.; Yu, W.; Yi, X.; Khan, A.; Yuan, F.; Zheng, Y. Recent progress on generative adversarial networks (GANs): A survey. *IEEE Access* **2019**, *7*, 36322–36333.

[28] Tolosana, R.; Vera-Rodriguez, R.; Fierrez, J.; Morales, A.; Ortega-Garcia, J. DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection. *arXiv* **2020**, arXiv:2001.00179.

[29] Engel, J.; Agrawal, K.K.; Chen, S.; Gulrajani, I.; Donahue, C.; Roberts, A. Gansynth: Adversarial neural audio synthesis. *arXiv* **2019**, arXiv:1902.08710.

[30] De Cao, N.; Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv* **2018**, arXiv:1805.11973.

[31] Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to adversarial attacks. In Proceedings of the International Conference on Learning Representation ICLR, Vancouver, BC, Canada, 30 April–3 May 2018.

[32] Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An empirical comparison of botnet detection methods. *Comput. Secur.* **2014**, *45*, 100–123.

[33] 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.