Journal of Cybersecurity and Privacy

MDPI

Article

Polymorphic Adversarial Cyberattacks Using WGAN

Ravi Chauhan [1], Ulya Sabeel [1], Alireza Izaddoost [2] and Shahram Shah Heydari [1,*]

[1] Faculty of Business and IT, University of Ontario Institute of Technology, Oshawa, ON L1G 0C5, Canada; ravi.chauhan1@ontariotechu.net (R.C.); ulya.sabeel@ontariotechu.net (U.S.)
[2] Department of Computer Science, California State University-Dominguez Hills, Carson, CA 90747, USA; aizaddoost@csudh.edu
* Correspondence: shahram.heydari@ontariotechu.ca

Abstract: Intrusion Detection Systems (IDS) are essential components in preventing malicious traffic from penetrating networks and systems. Recently, these systems have been enhancing their detection ability using machine learning algorithms. This development also forces attackers to look for new methods for evading these advanced Intrusion Detection Systemss. Polymorphic attacks are among potential candidates that can bypass the pattern matching detection systems. To alleviate the danger of polymorphic attacks, the IDS must be trained with datasets that include these attacks. Generative Adversarial Network (GAN) is a method proven in generating adversarial data in the domain of multimedia processing, text, and voice, and can produce a high volume of test data that is indistinguishable from the original training data. In this paper, we propose a model to generate adversarial attacks using Wasserstein GAN (WGAN). The attack data synthesized using the proposed model can be used to train an IDS. To evaluate the trained IDS, we study several techniques for updating the attack feature profile for the generation of polymorphic data. Our results show that by continuously changing the attack profiles, defensive systems that use incremental learning will still be vulnerable to new attacks; meanwhile, their detection rates improve incrementally until the polymorphic attack exhausts its profile variables.

Keywords: adversarial attacks; Generative Adversarial Network (GAN); Intrusion Detection Systems; DDoS/DoS attacks; machine learning; Wasserstein Generative Adversarial Network (WGAN)

Citation: Chauhan, R.; Sabeel, U.; Izaddoost, A.; Shah Heydari, S. Polymorphic Adversarial Cyberattacks Using WGAN. *J. Cybersecur. Priv.* **2021**, *1*, 767–792. https://doi.org/10.3390/jcp1040037

Academic Editor: Nour Moustafa

Received: 20 September 2021
Accepted: 9 December 2021
Published: 12 December 2021

1. Introduction

The increasing usage of the Internet in all aspects of life causes concerns regarding network security and needs constant improvements in securing Internet technologies from various attacks. There are many tools deployed to secure data communication or prevent cyber-security attacks, such as Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), Anti-Malware, Network Access Control, and Firewalls. Our focus in this work is on Intrusion Detection Systems (IDS), given the rise in malicious intrusions or attacks on network vulnerabilities [1].

IDS analyzes the traffic data to distinguish between malicious and normal traffic and to generate alerts so that necessary precautions can be carried out to prevent damage. With the advancement in network attacks, the security detections and prevention systems are also improving. Artificial Intelligence (AI) is now commonly used in defensive measures in IDS [2,3], and opponents also have started to use AI techniques for generating malicious attacks and adversarial data [4,5].

One of the frameworks to generate adversarial data is to use a Generative Adversarial Network (GAN). It is an architecture that consists of two deep neural networks: the Generator and the Discriminator. The generator synthesizes adversarial data, and the discriminator distinguishes between the original and the synthesized adversarial data. The generator and the discriminator compete in this way, and, in the end, the generator produces synthetic or adversarial data [6]. GAN has been utilized in research to generate various types of datasets such as images [7], sound [8], text [9], and network attack data [10].

Polymorphic is a term that consists of a vital keyword, morph, which means changing the form. In the context of network security, polymorphic attacks refer to the type of attack that mutates its signatures to evade detection. In other words, a polymorphic attack changes in such a way that it maintains its functionality. The polymorphic engine is employed to change the signature of the attack. The polymorphic property of attacks makes old detection signatures obsolete [11]. These types of attacks are specifically deployed when the detection system uses signature-based pattern matching techniques. The first known polymorphic attack was used to generate malicious URLs for a phishing attack [11]. This attack can evade the signature-based anti-phishing defense systems, and defense systems are unable to blacklist malicious URLs.

The current state of research on generating adversarial attack data using machine learning and AI methods is limited and mostly focused on specific formats such as image, text, and sound. For example, phishing attacks are automated using adversarial AI [12], and attackers can use GAN to generate the voice of a group of people to breach the security access [13]. However, its use in generating network attacks has not been fully investigated, which is the objective of this research.

Among network-based threats, Distributed Denial-of-Service (DDoS) and Denial of Service (DoS) attacks are the most common to generate, often using simple scripting tools such as Slowloris [14], GoldenEye [15], and Hulk [16]. Additionally, these attacks mostly target specific organizations and need fewer resources. In this research, we focus on generating DoS/DDoS attacks using adversarial network techniques, namely WGAN [17]. Furthermore, we develop a framework for generating polymorphic DoS/DDoS attacks in such a way that maintains the intensity of the attack but can be misclassified by an IDS as a regular network flow. The polymorphic attack generation engine will be useful for training IDS against unknown or polymorphic attacks.

1.1. Contributions

The main contributions of this research can be summarized as:

- We propose a WGAN-based adversarial engine that can launch polymorphic attacks on a black-box IDS. To the best of our knowledge, our work is the first to introduce and study the concept of polymorphic network attacks.
- To preserve the functional behavior of an attack, we employ Shapley Additive Explanations (SHAP) [18] that identify the functional features of the attacks.
- We introduce an adversarial polymorphic training mechanism to enhance the performance of IDSs against these attacks.
- We conduct comprehensive experiments and analyze the results to compare the performance of multiple IDSs against polymorphic adversarial DDoS/DoS attacks.

1.2. Outline of the Paper

This paper is organized as follows. Section 2 provides a detailed overview of the related works, followed by Section 3, which discusses deep learning-based techniques for cybersecurity. Section 4 provides details regarding the dataset used and the feature selection technique employed. Section 5 discusses our proposed framework. Section 6 describes the experimental setup, and thereafter, we present our results and analysis in Section 7. Finally, in Section 8, we conclude the paper and provide recommendations for future work.

2. Related Works

2.1. Taxonomy of Intrusion Detection Techniques Using ML and DL

Researchers have been working on various Machine Learning (ML) and Deep learning (DL) techniques to improve the functionalities and ability of the IDS. There are three techniques used by IDSs to detect various attacks: *Signature-based detection (knowledge-based)*, *anomaly-based detection (behavior-based)*, and *stateful protocol analysis (specification-based)* [1]. The first technique analyzes a network for a specific pattern of predefined

attack. The limitation of this technique is that it is unable to detect an unknown attack and zero-day attacks. The second method classifies the network flow data into standard and anomalous data by comparing it with normal network user data. The limitation of this technique is that it results in high false positives during classification [1]. The third method is used to trace protocol states and identify unknown commands. The disadvantage of this approach is that it is unable to identify slow-rate attacks, which have similar behavior as normal data [1]. In the following subsections, we discuss several signature-based, anomaly-based, and specification-based detection techniques employed by current network security researchers.

2.1.1. Signature-Based Intrusion Detection

Various signature-based supervised machine learning classifiers, such as Support Vector Machine (SVM) [19,20], Naïve Bayes (NB) [19,20], KNN [20], Decision Tree (DT) [21], Random Forest (RF) [19,21], Logistic Regression (LR) [21], and Multi-Layer Perceptron (MLP) [22] are studied by current researchers for intrusion detection. Their results show that these algorithms have higher accuracy on known or similar attack patterns. The authors in [23] pointed towards the recent developments and shortcomings in signature-based IDS. They suggested that the main concern with the latest IDS is that they tend to alarm on fake attack data, which results in high false positives in terms of machine learning. A comparison of the detection rate and accuracy between various machine learning techniques, such as Artificial Neural Network (ANN), SVM, NB, RF, and AdaBoost has been discussed. In [24], the authors provide an overview of several single classifiers, hybrid classifiers, and ensemble classifiers consisting of multiple ML models. The authors have compared their results based on multiple network security datasets. Their research shows that hybrid and ensemble classifiers provide a better accuracy and detection rate as compared to a single classifier. The authors in [25] compared the efficacy of binary and multi-class ML classifiers. These signature-based classifiers are compared based on several performance metrics, such as True Positive Rate, False Positive Rate, Area Under the ROC Curve (AUC), and incorrect classifications. Their results depicted that reducing the number of target classes in training data helps in reducing the class imbalance for minority attack classes, thereby improving the overall IDS performance.

Likewise, several advanced DL classifiers, such as Deep Neural Network (DNN) [26,27], Convolutional Neural Network (CNN) [28,29], Recurrent Neural Network (RNN) [30,31], Long Short Term Memory (LSTM) [30,31], and Gated Recurrent Unit (GRU) [32,33] are employed by current network security researchers as signature-based intrusion detection techniques. Although these supervised DL classifiers provide better attack detection rates as compared to the traditional ML classifiers, they can identify only known attack patterns and do not perform equally well on unknown attacks or variants of known attacks [2,3].

2.1.2. Anomaly-Based Intrusion Detection

The anomaly detection technique uses several statistical methods, machine learning, and deep learning in IDS. The authors in [34] proposed a kernel-based IDS that can detect anomalies, such as DDoS attacks. It uses the ML technique named K-means clustering to classify between adversarial and standard examples. In [35], the authors employed a statistical analysis-based anomaly detection to identify normal and malicious data. An adaptive threshold profile is maintained for normal and abnormal behaviors together with a trust management scheme. An ML network anomaly detection technique based on the Isolation Forest classifier is proposed in [36]. Several anomalous behaviors are identified through statistical feature extraction methods. The authors in [37], provided a comparative analysis of several unsupervised network anomaly detection techniques, such as K-means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Gaussian Mixture Model (GMM), and Local Outlier Factor (LOF). Their results indicated that DBSCAN gives the best performance based on the detection rate. Another unsupervised ML clustering-

based technique is employed in [38]. The authors employed Sub-Space Clustering (SSC) and One Class Support Vector Machine (OCSVM) to identify both known and unknown attacks.

Several DL algorithms are employed for unsupervised network anomaly detection. In [39], the authors introduced a Deep Autoencoder model for network flow-based anomaly detection. While the detection is only based on the distribution of traffic features, investigation of other network features is left for future work. The authors in [40] introduced a hybrid anomaly detection classifier consisting of a CNN model and an unsupervised Deep Autoencoder (DAE) model. In this case, CNN is used for traffic profiling and the unsupervised DAE is used for unsupervised detection of normal and anomalous traffic. In [41], the researchers compared several anomaly detection techniques, such as K-means, Deep Autoencoding Gaussian mixture model (DAGMM), Self Organizing Maps (SOM), and Adversarial Learned Anomaly Detection (ALAD) to identify normal and anomalous network flows. Based on their analysis, ALAD is the best performing model since it learns from adversarial samples and provides stable results on minority attacks.

2.1.3. Specification-Based Intrusion Detection

Multiple specification-based intrusion detection techniques are applied by network security researchers to identify protocol-based attacks. In [42], the authors introduced a specification-based IDS engine for networks without an infrastructure. The paper mainly focused on the Ad-hoc On Demand Distance Vector (AODV) routing protocol. Experimental results showed that attacks can be detected with high detection accuracy. The authors in [43] proposed a behavior rule specification-based IDS for safety-critical medical cyber systems. In this case, the behavior of a medical device is monitored for any hidden attacks to reduce false positives and ensure the patient's safety. The authors report that their proposed technique outperforms other available techniques for intrusion detection in healthcare applications. In [44], the authors explored a specification heuristics-based IDS for Internet of Things (IoT) Networks. In this paper, the researchers focused on identifying intrusion at each device level rather than the network level. Experimental results revealed that their approach is successful in identifying abnormal sequential patterns as compared to other available approaches.

2.2. Attacks Using Generative Adversarial Networks

With the recent developments towards Machine Learning and Deep Learning techniques, Intrusion Detection Systemss are getting advanced with these methods. However, there is limited research testing the integrity of the advanced IDS against adversarial data.

A framework to generate adversarial malware using GAN to bypass the detection system is developed in [45]. The objective of this research was to use a black-box malware detector since most of the attackers are unaware of the detection techniques used in the detection system. Instead of directly attacking the black-box detector, the researchers created a model that can observe the target system with the corresponding data. Then, this model calculates the gradient computation from the GAN to create adversarial malware data. With this technique, the authors achieved a model accuracy of around 98%.

The same methodology has been employed to generate adversarial attacks for Android applications. The authors in [46] present a model to generate adversarial android malware using GAN. The model consists of the generator, the discriminator, and Malware Detector. The generator takes a random noise vector and produces the adversarial data. The discriminator gets benign data and adversarial malware and then differentiates between real and perturbed data. The discriminator provides feedback in the form of loss to the generator. If the generated sample is distinguishable, it will increase the loss, and decrease it otherwise. They have used various classifiers, such as Support Vector Machine, Random Forest, and Logistic Regression as the machine learning classifiers for the GAN model.

The Wasserstein GAN (WGAN) model was introduced in [17]. To generate a malicious file, the authors in [10] proposed a method that uses WGAN in order that a detection system signifies the adversarial malicious file as a regular file. They have achieved an

accuracy of around 99%, proving that their method can generate adversarial malicious files that can bypass the detection system.

A study in [47] used WGAN to generate simulated attack data. According to the authors, many tools can generate simulated attack data. However, this process could take a long time and a large amount of resources. Using the proposed technique, they have produced millions of connection records with only one device and within a short period. They used the KDD Cup 1999 dataset as the training set. Their experiment suggests that as compared to GAN, the WGAN learns faster and generates better results.

Ring et al. [48] proposed a method that produces flow-based attack data using WGAN. This research uses the CIDDS dataset to test and train the proposed method. They have suggested that the flow-based dataset consists of categorical features, such as IP address, port numbers, etc., where the GAN model is unable to process. They have also proposed a method to preprocess the categorical data and transform that into continuous data. They have employed several techniques to evaluate the quality standard of the adversarial data. Their results show that it is possible to generate real network data using this method.

The benefits of WGAN to improve IDS functionality has been studied in [49]. The authors proposed a technique, called IDSGAN, to generate adversarial attack data and test the attack against the Intrusion Detection System. They have utilized the NSL-KDD as the benchmark dataset to generate an adversarial attack on IDS. They have tested this technique with various machine learning classifiers, such as Support Vector Machine, Naïve Bayes, Multilayer Perceptron, Linear Regression, Decision Tree, and Random Forrest. Four types of attacks, such as Probe, DoS, User to Root, and Root to Local to generate adversarial attack data have been studied in this research.

Shahriar et al. [50] introduced a GAN-based IDS named G-IDS for the detection of attacks. In this work, G-IDS is compared with a standalone IDS (S-IDS) that is not trained with adversarial attack samples. Although the results indicate a better performance of G-IDS in terms of precision, recall, and F1-score, the model is not evaluated against polymorphic attacks.

Zhang et al. [51] proposed a hybrid model named VAE-GAN for intrusion detection. This model combines a Variational Autoencoder (VAE) and a Generative Adversarial Network (GAN) to classify attack and normal instances. While the results of three IDS models, such as CNN, MLP, and RNN show improvements after training with samples generated using the VAE-GAN model, no evaluation is provided on polymorphic attacks.

2.3. Problem Statement

Research works based on generating adversarial attacks using GAN mainly focus on generating adversarial data and studying IDS for the possibility of attack detection. Most of the state-of-the-art research does not focus on training the IDS with adversarial data generated by the GAN and testing if the IDS can detect similar kinds of attacks in the following cycles. The current research work also lacks the idea of polymorphic attacks, i.e., to update the attack feature profile by manipulating the features of training data and trying to generate a new variety of adversarial data to evaluate the IDS functionality gradually. In addition, research shows that the traditional variant of GAN models based on Goodfellow et al. [6] does not scale with a large dataset, and is unstable with large-scale applications [52].

In this research, we aim to generate an AI-based adversarial DDoS/DoS attack using WGAN. Moreover, this attack will be profile-based and polymorphic, which means the attack will change its feature profile periodically.

The main objective of the research is to build a model that generates feature profile-based polymorphic DDoS/DoS attacks and evaluates if it can evade the IDS. This work also includes monitoring the performance of the GAN model to study the number of cycles the polymorphic adversarial DDoS/DoS attack can evade the IDS. We also apply the SHAP feature selection technique to distinguish essential features from the entire dataset.

3. Deep Learning-Based Techniques in Cybersecurity

3.1. Generative Adversarial Networks

In this section, we provide an overview of GAN architecture. Generative Adversarial Network is a paradigm based on machine learning models that can generate synthetic data from the original input data. It consists of two neural networks known as the Generator and the Discriminator [6].

The discriminator can be simply called a classifier that distinguishes the generated data as original or fake. It takes two forms of data: Original data and the data generated by the generator. The discriminator uses original data as a positive example and generated data as negative/adversarial examples during training. L_D represents the penalty to the discriminator. When the discriminator cannot detect or correctly differentiate the data, the penalty increases and decreases otherwise. To update the weights of the discriminator, it uses backpropagation. Another loss L_G represents a loss of the generator [53]. Figure 1 shows the discriminator training process.

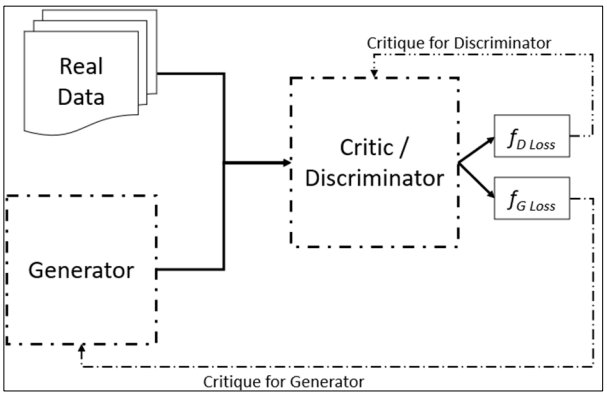

Figure 1. WGAN architecture.

The generator produces a synthetic dataset by receiving feedback from the discriminator and learns to produce data in order that the discriminator classifies the synthetic data as the original.

The training of the Generator includes the following steps:

- A random input noise.
- The generator, to produce synthetic data from the random data.
- The discriminator, to distinguish the synthetic data and original data.
- Loss L_G that fines the generator if it is unable to produce data, which can deceive the discriminator.

As discussed, there are two variants of neural networks in the GAN. Therefore, it needs to train the generator and the discriminator alternately. Moreover, it is important to check if GAN is converged or not. The alternative training works as follows:

(1) Training of the generator runs for some epochs.
(2) Training of the discriminator runs for some epochs.
(3) Continue repeating steps 1 and 2 until the GAN converges.

To train the GAN more efficiently, we need to keep either of the Neural Networks constant. For instance, while training the generator, we need to keep the discriminator constant. Otherwise, it will be difficult to converge [54]. While training the discriminator, the generator needs to be constant since it needs to learn to differentiate between the generated and fake data.

The loss function represents the difference value between the generated data and the adversarial data as follows:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}\ (x)}[log(D(x))] + E_{z\sim p_z(z)}[log(1-D(G(z)))] \quad (1)$$

The authors in [6] introduced a loss function named min-max loss. They trained the discriminator D to maximize the average of the $D(x)$, with $D(x)$ denoting the estimated probability of data as original, and the $log(1 - D(G(z)))$. $G(z)$ represents the output synthesized by the generator from the noise z. $D(G(z))$ is the approximate value of the discriminator that the generated data is real. E_X represents the expected value over the original data, $p_z(z)$ is input noise variables, and E_z represents the expected value of the random data inputs to the generator. Moreover, the authors concurrently train the generator G, which seeks to minimize the $log\ (1 - D(G(z)))$ predicted by the discriminator for synthetic data. The discriminator D and the generator G play a min-max game with the following value function $V\ (D, G)$.

To minimize the loss at the generator, we need to minimize the value of the $log\ (1 - D(G(z))$. The lower loss at G indicates that the generator produces synthetic data that can be classified as the original.

According to the research in [46,52,54], most of the common challenges in training a GAN are as follows:

- Convergence problem: Classification performance of the discriminator decreases in the following cycles. Training the GAN from this point indicates that the generator trains with less meaningful data. This state is known as the convergence problem.
- Mode collapse: In ideal conditions, a GAN can produce a good variety of data. However, if a generator learns to produce a specific set of data in order that the discriminator classifies them as the original, then the generator will only produce these sets of data and easily deceive the discriminator. This condition is called mode collapse.

3.2. Wasserstein GAN

To overcome the above-mentioned issues, Arjovsky et al. proposed a method known as Wasserstein GAN [17]. Wasserstein GAN provides a better approximation of distributed data that was given in the training set. WGAN uses a discriminator with a critic that provides a score of how real or fake generated data is. In contrast, a discriminator in traditional GAN predicts and classifies the generated data only as original or fake, a binary selection. Figure 1 shows the WGAN model architecture.

In Figure 1, $f_{D\ loss}$ represents a loss function that provides critique values for the discriminator, and $f_{G\ loss}$ represents a loss function for the generator. The following are the differences in the implementation of WGAN. A critic score < 0 depicts real data, and a score > 0 depicts fake or synthetic data.

It trains or updates the discriminator/critic multiple times as compared to the generator in each cycle. Two loss functions used by WGAN are discriminator/critic loss and generator loss, which are given below:

$$L_D = \nabla_w \frac{1}{m} \sum_{i=1}^{m} [\, f_w(x^{(i)}) - f_w(g_\theta(z^{(i)}))\,] \tag{2}$$

Equation (2) [46] specifies the discriminator/critic loss that can be simplified as a difference between the average critic score of real data and the average critic score of fake data. Here, $f_w\left(x^{(i)}\right)$ represents an average critic score on real data and $f_w(g_\theta(z^{(i)}))$ represents an average critic score on fake/generated data:

$$L_G = \nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w\left(g_\theta\left(z^{(i)}\right)\right) \tag{3}$$

Equation (3) specifies the generator loss that can be simplified as $1 - f_w\left(g_\theta(z^{(i)})\right)$, in which $f_w\left(g_\theta\left(z^{(i)}\right)\right)$ depicts the average critic score of fake data. Overall, the main advantages of WGAN include the fact that it does not suffer from the mode collapse problem, and the generator learns well even if the critic accurately discriminates the adversarial data. Both of the loss functions motivate a separation between a score for synthetic data and real data, which is not necessarily positive and negative [55].

It suffices to say that WGAN has been proven to generate high-quality synthetic attack data. We have followed a similar methodology to generate synthetic DDoS attack data in this research.

4. Dataset and Feature Selection

4.1. Dataset Properties

As detection systems evolved with time, various new attacks have also emerged that could compromise networking systems. The training dataset needs to be updated with the latest attack features. The Canadian Institute for Cyber Security has developed a dataset from real-time simulations, and has published various datasets involving numerous attacks, such as Android malware, Botnet, DoS, DDoS, etc. Their Intrusion Detection Evaluation Dataset known as CICIDS2017 [56] contains benign data and commonly known attacks, such as Brute Force, SSH, DoS, Web Attack, Botnet, and DDoS. To produce a reliable dataset, the authors have considered critical criteria, such as complete network configuration, complete traffic, labeled dataset, complete interaction, complete capture, available protocols, attack diversity, feature set, and metadata. The dataset consists of more than 80 features that are important as per the latest network standards, and most of them were not available in the previously known datasets. In addition, none of the other datasets have considered these benchmarks. The CICIDS2017 data consist of eight different files that contain regular traffic and attack traffic data. Table 1 depicts the properties of the CICIDS2017 dataset.

Table 1. Properties of the CICIDS2017 dataset.

Feature	Values
Total number of flows	2,830,540
Total number of features	83
Number of classes/labels	15

Moreover, the CICIDS2017 dataset consists of various types of attacks along with the normal network flow. The following Table 2 consists of the attack and benign labels available in the dataset.

Table 2. Labeled features in CICIDS2017.

Normal/Attack Labels	Number of Flows
BENIGN	2,359,087
BOT	1966
DDOS	41,835
DOS GOLDENEYE	10,293
DOS HULK	231,072
DOS SLOW HTTPTEST	5499
DOS SLOWLORIS	5796
FTP-PATATOR	7938
HEARTBLEED	11
INFILTRATION	36
PORTSCAN	158,930
SSH-PATATOR	5897
WEB ATTACK—BRUTE FORCE	1507
WEB ATTACK—SQL INJECTION	21
WEB ATTACK—XSS	652

4.2. Feature Selection

Feature selection is an essential aspect of machine learning. If we train the model without determining the critical features of the dataset, the predicted results will have more noise and uncertain results. Moreover, while using a dataset with a higher number of

feature sets, it is unnecessary to use all the available features since the machine learning method requires more resources and time to process a large volume of data with additional features. Among several techniques to select important features from a dataset, we have employed SHAP in this research.

Feature Selection using SHAP

Shapley **A**dditive ex**P**lanations (SHAP) [18] is a feature selection technique whose goal is to signify the contribution of each feature to the predicted value. SHAP aims to satisfy two critical measures to define feature importance, which are consistency and accuracy. The idea behind Shapley values is that the outcome of each possible combination (or coalition) of features needs to be examined to determine the importance of a single feature. The mathematical explanation of this is as follows:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{4}$$

where g represents the overall result of the Shapley values, $z' \epsilon \{0, 1\} M$ is a coalition vector, M is the max coalition size, and ϕ_j represents the presence of feature j that contributes towards the final output. In the coalition vector, 0 means the corresponding value is "not present" and 1 means it is "present".

Herein, each node represents a coalition of features. Edges represent the inclusion of a feature that was not present in the previous coalition. Equation (4) trains each coalition in the power set of the features to find the most critical feature from the dataset. The advantages of using SHAP are as follows:

- Global Interpretability: This technique provides essential features from a dataset and a contribution of each feature for a target result and effect of the feature. To calculate global importance, we need to find an average of SHAP vales.

$$I_j = \sum_{i=1}^{n} \left| \phi_j^{(i)} \right| \tag{5}$$

- Local Interpretability: With this method, we can get an impact of an individual feature across the whole dataset.

We ran the SHAP explainability model on the CICIDS2017 DDoS/DoS data. Figure 2 is known as a summary plot that can represent an effect of the feature, positive or negative, on the result. Furthermore, the dark red color represents a higher impact of a feature, and the blue color represents a lower impact of a feature on the output value.

We use the results of the SHAP analysis, such as those in Figure 2, in our feature selection stage. These results identify the functional features (attack features) that are used by IDS, and the non-functional features that would be used in creating polymorphic attacks while keeping functional features constant. SHAP analysis helps us separate functional and non-functional features by quantifying the impact of each feature on the correct identification of the attack data.

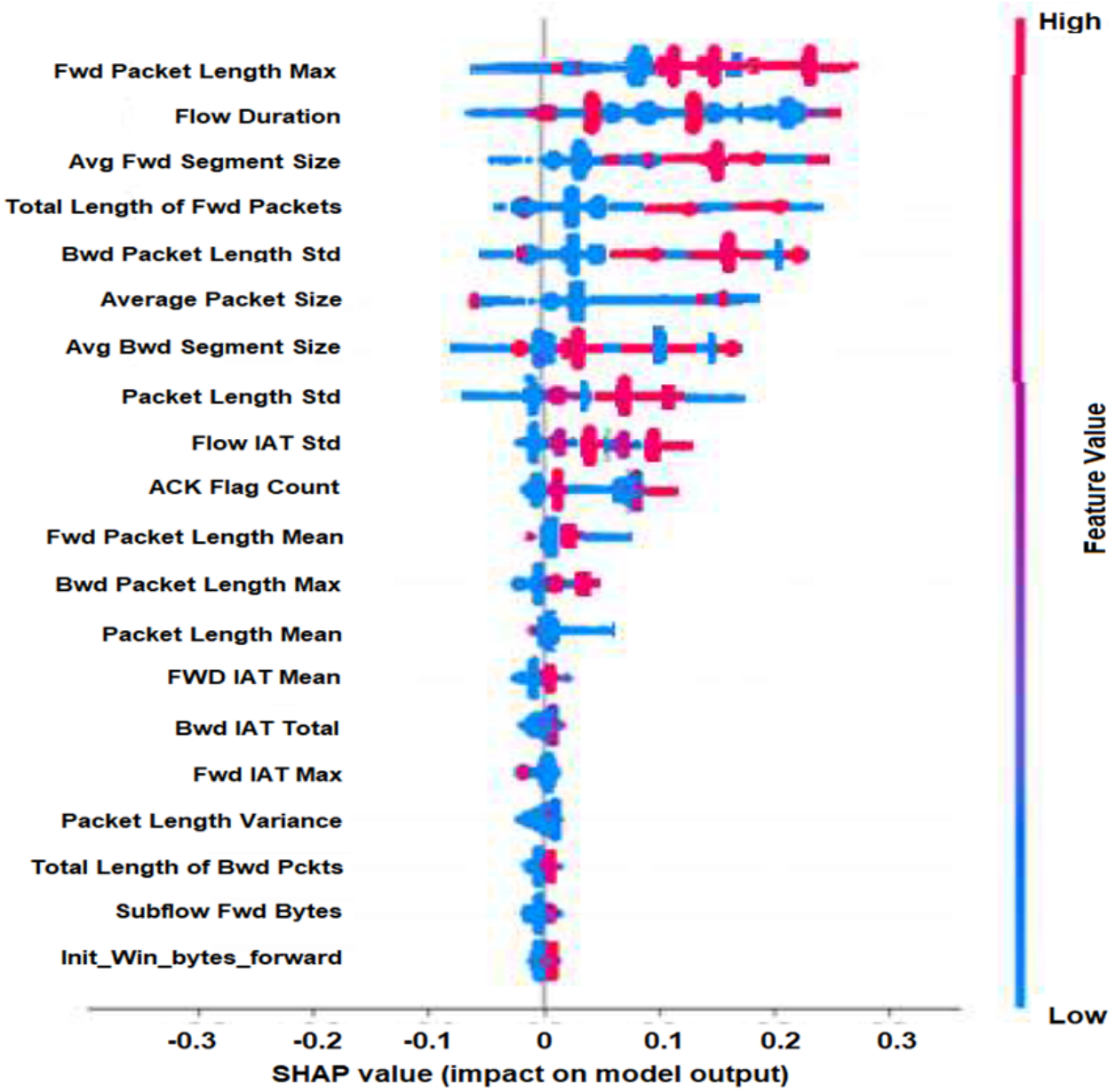

Figure 2. Summary plot with feature impact using SHAP. Red indicates a high-impact attack feature, while blue indicates a low-impact attack feature.

5. Proposed Framework

The proposed framework involves the WGAN model that produces adversarial attacks to train the IDS with previously generated adversarial data, as well as the polymorphic engine to generate polymorphic DDoS/DoS attacks using the polymorphic data to attack the IDS.

5.1. Adversarial Attack Generation Using Wasserstein GAN

We have used the DDoS/DoS attack data from the CICIDS2017 [56] to train the proposed model. To generate an adversarial attack, we considered a combination of a random noise vector of the same size as the selected features from the dataset.

The generator in this framework is a feed-forward neural network that consists of five linear layers. The input layer consists of neurons as per the selected number of features, and the output layer consists of two neurons. The input layer receives a set of features according to the experiment, and the output layer generates the desired data. The generator

consists of three hidden layers that are optimal for this scenario and prevent overfitting the training data.

In the next step, the generated adversarial attack combined with the benign network flow data will be fed to the Intrusion Detection System. The IDS will detect the attack and send predicted labels to the discriminator, which is the detection success rate, and the discriminator will send the critique to the generator using the backpropagation, in order that in the next cycle, the generator can improve the production of adversarial DDoS/DoS attack. The IDS consists of four layers, in which the input and output layer consists of two neurons each. The IDS consists of two hidden layers that are ideal since it only detects if the test data consist of an attack or are benign.

We used a signature-based black-box Intrusion Detection Systems to test the detection rate of the adversarial DDoS/DoS attacks. The reason for using this system is that most of the time, the type of attack detection system is unknown to the attackers. Attackers rely on the responses received from the detection system, and black-box IDS is the right choice for this model.

Finally, the critic or discriminator consists of four layers. The input layer accepts two types of data from the black-box IDS. The output layer provides two critics, one for the generator and one for itself.

To calculate the loss, we have used loss functions for the generator and the discriminator, which are as follows [49]:

$$P_G = E_{M \in S_{attack}\ ,\ N} \ - D(G(M, N)) \tag{6}$$

where P_G represents the penalty to the generator. M is an m-dimensional attack vector, and N is an n-dimensional noise vector. E is the estimated value over the random inputs to the generator. S_{attack} represents the original training attack data. A lower penalty to the generator means that it is performing well and produces attack data that can bypass the IDS.

$$P_D = A_{S \in B_{Benign}} D(s) + A_{S \in B_{Attack}} - E_{S \in B_{Attack}}\ D(s) \tag{7}$$

where P_D represents the penalty to the discriminator. E is the overall estimated feature values of the generated adversarial attack data. A is the actual feature value of benign and the attack data. A lower penalty to the discriminator indicates that the discriminator performs well. It calculates if the generated data are closer to the DDoS/DoS attack or benign data. Algorithm 1 shows the process that is represented in Figure 3.

Algorithm 1. Adversarial attack generation

Input:
Generator—noise vector N, DDoS/DoS Attack Data
Critic/Discriminator—Sattack, and Sbenign

Output:
Trained Critic/Discriminator and Generator

for *epochs = 1, . . . , MAX EPOCHS* ***do***
  for *G-iterations* ***do***
    Generator creates adversarial network attacks using Sattack, and updates the penalty using P_G *function once it receives the critique.*
  end
  While generating adversarial DDoS/DoS data and feeding the data to IDS to test if it detects the attack.
  for *D-iterations* ***do***
  Receive detected labels from the IDS and send a critique to the Generator. Update the penalty using P_D *function.*
  end
end

5.2. Train the Generator to Create an Adversarial Attack

The generator needs to maintain constant values for the features that have higher SHAP values. An example of how the generator produces an adversarial attack by the proposed technique is shown in Figure 4, in which the darker shade explains the values of

the features that are contributing to the attack, whereas non-highlighted values depict the feature value of a regular or non-attack feature.

As shown in Figure 4, to maintain the intensity of the attack we need to keep the functional attack features constant and only change the feature values that are not contributing to the attack. Therefore, to evade the black-box IDS, the generator changes the values of the features that are not contributing to the DDoS/DoS attack.

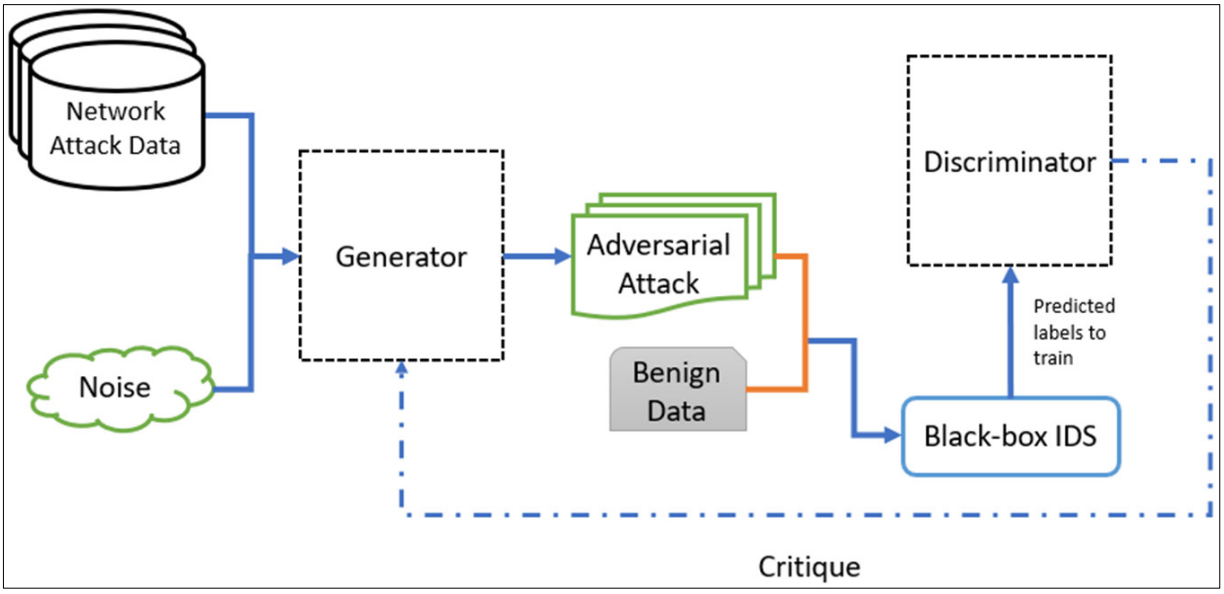

Figure 3. Generating the adversarial DDoS/DoS attack.

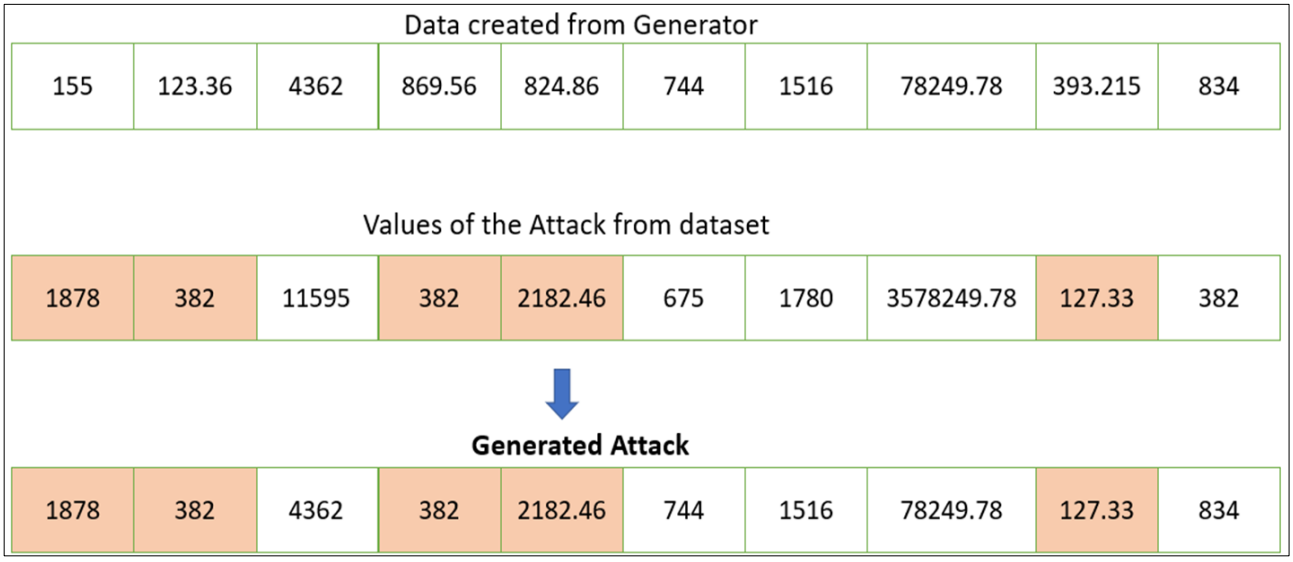

Figure 4. The process to generate the adversarial DDoS/DoS attack.

5.3. Training an IDS with the Generated Adversarial Data

In this section we will discuss the training of the IDS in order that we can evaluate the performance of the IDS with the adversarial data.

We consider three inputs to train the IDS: Normal or benign data, new adversarial data, and previously generated adversarial data. The IDS learns about the adversarial data and tries to detect the DDoS/DoS attack data.

5.4. Polymorphic Engine to Generate a Polymorphic Attack

We evaluated different methods for updating the attack feature profile to generate polymorphic adversarial DDoS/DoS attacks. The following approaches were considered:

Approach 1: Select a subset of non-functional features in a polymorphic attack cycle. Randomly increase the values of these features in the attack profile by a small percentage to generate new data for training the attacker, which in turn synthesizes a polymorphic attack each time IDS detects the previous adversarial polymorphic attack. Repeat this process until the attacker runs out of features or until the IDS can classify all of the polymorphic attacks.

Approach 2: Add a new set of features from the predefined list of features in the current attack profile to train the attacker after the IDS detects previous adversarial attacks. Keep repeating this process for each polymorphic attack until the attacker runs out of features or until the IDS can classify all of the polymorphic attacks. This approach employs both manual feature selection, as well as automated feature selection using Reinforcement learning.

In the above techniques, we assumed that an attacker could modify the feature profile and train the generator model with the new feature profile every time after the IDS detects a polymorphic attack.

Algorithm 2 shows the process for training the IDS, and Algorithms 3 and 4 are employed to synthesize a polymorphic adversarial attack. In the case of Algorithm 3, we select a set of five non-functional (non-attack) features from the training data and randomly increased their values by 5%. The functional (attack) features are kept constant to preserve the attack characteristics. The generator is then trained using the new dataset to synthesize polymorphic attacks and evade detection by the IDS. Then, the IDS is retrained with new synthesized adversarial data using Algorithm 2. The polymorphic attack generation phase is repeated by adding another set of five non-functional features and the IDS performance is evaluated again. This process is repeated until the attacker has no more features left or until the IDS can successfully identify all of the attacks.

Algorithm 2. Training IDS with the adversarial attack data.

Input:
Generator—N noise + Original Attack Data
IDS—Benign or Normal Data, Adversarial Data, and Previously Generated Adversarial Data
Critic/Discriminator—Sattack and Sbenign

Output:
Trained Critic/Discriminator, Generator, and IDS

for *epochs = 1, . . . , MAX EPOCHS* ***do***
  for *G-iterations* ***do***
    Generator creates adversarial network attacks using Sattack and updates loss using P_G *function*
  end
  for *D-iterations* ***do***
    Critic/Discriminator classifies the network data to Bbenign and Battack. Update loss using P_D *function*
    Feed Battack (Adversarial data) and Previously Generated Adversarial Data to the IDS
  end
end

Algorithm 4 works in a slightly different manner. Here, in the first cycle, a subset of three non-functional features is selected and their values are randomly increased by 5% to create new training data for the attacker, which can synthesize polymorphic attacks. The IDS performance against these polymorphic attacks is evaluated and retraining is performed to improve its detection results. In the next phase of polymorphic attack generation, we select a new subset of six non-functional features to synthesize another polymorphic attack. This process is repeated until the attacker has no more features left or until the IDS can successfully identify all of the attacks.

Algorithm 3. for generating the polymorphic adversarial attack.

Input:
Generator—Noise, S_{attack}, X_{attack}
Critic/Discriminator—S_{attack}, X_{poly}, S_{benign}

Output:
Trained Critic/Discriminator and Generator

F_tot = total number of features
Feature Set = S (1: F_tot)
N = total number of non-functional features in F_tot
nf = number of non-functional features selected for mutation in the first cycle of polymorphic attacks
steps = range (N/nf)
/ Data preparation for the Attacker and Polymorphic attack generation */*
for *i = 1 to steps* ***do***
*Choose a subset Si (0: i*nf-1) of non-functional features from S_{attack}. Randomly increase the selected feature values by a small percentage to create X_{attack}*
for *epochs = 1 to MAX EPOCHS* ***do***
for *G-iterations* ***do***
Generator creates adversarial polymorphic attacks (X_{poly}) using X_{attack} + Noise. Update loss using P_G function
end
/ Training the Discriminator model */*
for *D-iterations* ***do***
Discriminator classifies the network data to B_{benign} and B_{attack}. Update loss using P_D function
end
end
end

Algorithm 4. for generating the polymorphic adversarial attack.

Input:
Generator—Noise, S_{attack}, X_{attack}
Critic/Discriminator—S_{attack}, X_{poly}, S_{benign}

Output:
Trained Critic/Discriminator and Generator

F_tot = total number of features
Feature Set = S (1: F_tot)
N = total number of non-functional features in F_tot
nf = number of non-functional features selected for mutation in the first cycle of polymorphic attacks
steps = range (N/nf)
prev = 0
/ Data preparation for the Attacker and Polymorphic attack generation */*
for *i = 1 to steps* ***do***
if *(prev < N)*
Choose a subset Si (prev: prev + (i×nf) − 1) of non-functional features from S_{attack}. Randomly increase the selected feature values by a small percentage to create X_{attack}
Update the value of prev to {prev + (i×f)}

for *epochs = 1 to MAX EPOCHS* ***do***
for *G-iterations* ***do***
Generator creates adversarial polymorphic attacks (X_{poly}) using X_{attack} + Noise. Update loss using P_G function
end
/ Training the Discriminator model */*
for *D-iterations* ***do***
Discriminator classifies the network data to B_{benign} and B_{attack}. Update loss using P_D function
end
end
end

6. Experiment Setup

The following are the libraries used in the overall work of this research.

- PyTorch [57] is an open-source machine learning platform that is based on the Torch library. We used PyTorch library to create neural networks for Black-box IDS, the generator, and the discriminator or critic. For example, to generate random noise, we have used a "*torch.Tensor*" method.
- Scikit-learn [58] is a machine learning library for python that supports various classifications, regressions, and clustering techniques. Examples include sklearn.utils and sklearn.metrics.

- Pandas [59] is a python library that is used to read, manipulate, and analyze the dataset. For example, to read CSV files, we use the read_csv() method from this library.

6.1. Hyperparameter Adjustment

Hyperparameters are essential properties that define the characteristics of the training process of machine learning or a deep learning model. For a deep neural network, hyperparameters are variables that explain its structure. Table 3 depicts hyperparameters that are used to optimize the generator and discriminator in WGAN for this research.

Table 3. Hyperparameters.

Hyperparameter	Description
Batch_Size	Defines the number of samples to consider for one iteration
learning_rate	Controls the weights of a neural network
Critic_Iters	Critic_iters for each generator cycle
Optimizer	Methods used to update the attributes of the neural networks, e.g., Adam, Rmsprop, Adagrad
Epochs	Number of training cycles

We train the WGAN model for a total of 100 epochs with a batch_size of 512, learning_rate set to 0.001, critic_iters equal to 5, and the optimizer selected using trial and error is RMSprop. Batch_size, learning_rate, critic_iters, and epochs are optimization hyperparameters related to the optimization and training process of the model. In comparison, an optimizer is a model-specific hyperparameter.

6.2. Evaluation Metrics

The following metrics are employed to evaluate the performance of IDS in this work:

- **Detection Rate** (***DR***) **or Precision**—a ratio between the correctly detected attack samples over all the samples classified as an attack by the IDS. It is also known as a ratio between True Positives and the sum of True Positives and False Positives.

$$DR = \frac{TP}{TP + FP} \tag{8}$$

- *Recall*—a ratio between the correctly detected attack samples over the total attack sample data. It is also known as a ratio between True Positives and the sum of True Positives and False Negatives.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

- *F1_Score*—represents the harmonic mean of precision and recall.

$$F1\_Score = 2 \times \frac{(precision \times recall)}{(precision + recall)} \tag{10}$$

- True Negative Rate (TNR)—a ratio between the correctly detected benign samples over the total benign sample data. It is also known as a ratio between True Negatives and the sum of True Negatives and False Positives.

$$TNR = \frac{TN}{TN + FP} \tag{11}$$

where *TP*, *FP*, *TN*, and *FN* represent True Positives, False Positives, True Negatives, and False Negatives, respectively.

The following metric is employed to evaluate the performance of the attacker:

- IDS Evasion Success Rate (*ESR*)—the rate of increase of undetected adversarial polymorphic attack samples by IDS compared to the original attack samples.

$$ESR = 1 - \frac{(Adversarial\ polymorphic\ detection\ rate)}{(Original\ detection\ rate)} \quad (12)$$

7. Results and Discussion

This section describes the results of various experiments for different scenarios and the analyses of the findings.

The first step of the research is to generate adversarial DDoS/DoS data that can evade detection by the Black-box IDS. We have selected a tree-based Random Forest classifier as the IDS model for our experiments. Figure 5 represents the results of the IDS against adversarial attacks. As seen in the graph initially, after training the attack generator for 100 epochs, it learns to synthesize adversarial data and is successful in evading the IDS by reducing the Detection Rate (DR) close to zero. After retraining the IDS with the previously generated adversarial DDoS/DoS data, the performance improves significantly.

7.1. Polymorphic Adversarial DDoS/DoS Attack Generation

In this section, we discuss the performance of a Random Forest-based IDS in different polymorphic adversarial attack cycles based on Algorithm 3. In Figure 6, the red-colored graph suggests the polymorphic attack phase and the blue-colored graph depicts the results after retraining the IDS with previously synthesized polymorphic adversarial data. During polymorphic cycle 1, the DR for the IDS is reduced to almost 25%. After retraining the IDS in the same cycle, we observe significant improvements in the DR values. A similar behavior is observed in cycle 2. However, in cycle 3, we notice that the DR of IDS against polymorphic attacks has improved to approximately 82% as compared to cycles 1 and 2. Finally, in cycle 4, the IDS can no longer be evaded by the attacker. The DR of IDS does not drop to a lower value in the attack phase. Therefore, no retraining is required for this cycle.

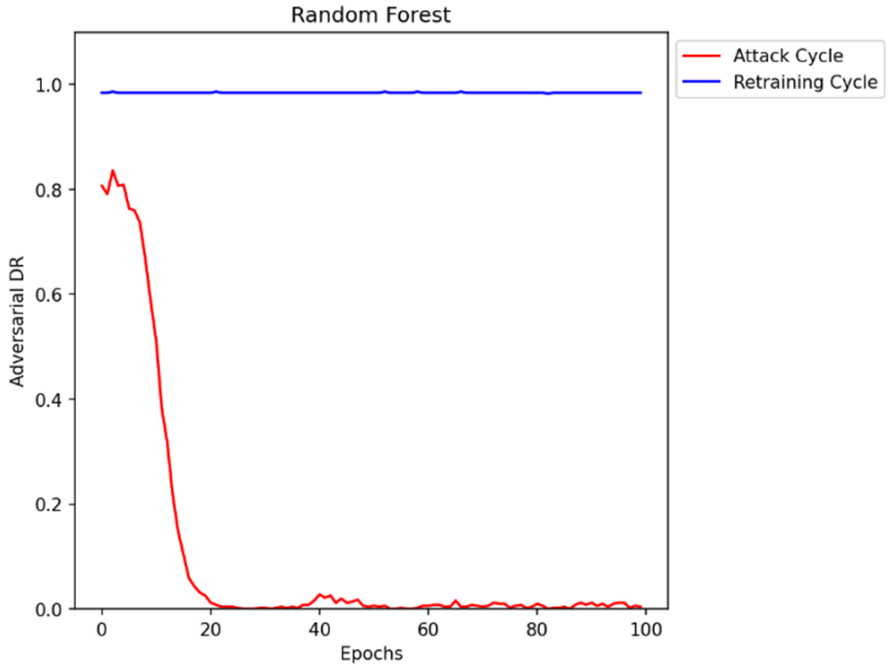

Figure 5. Adversarial DDoS/DoS attack generation and retraining.

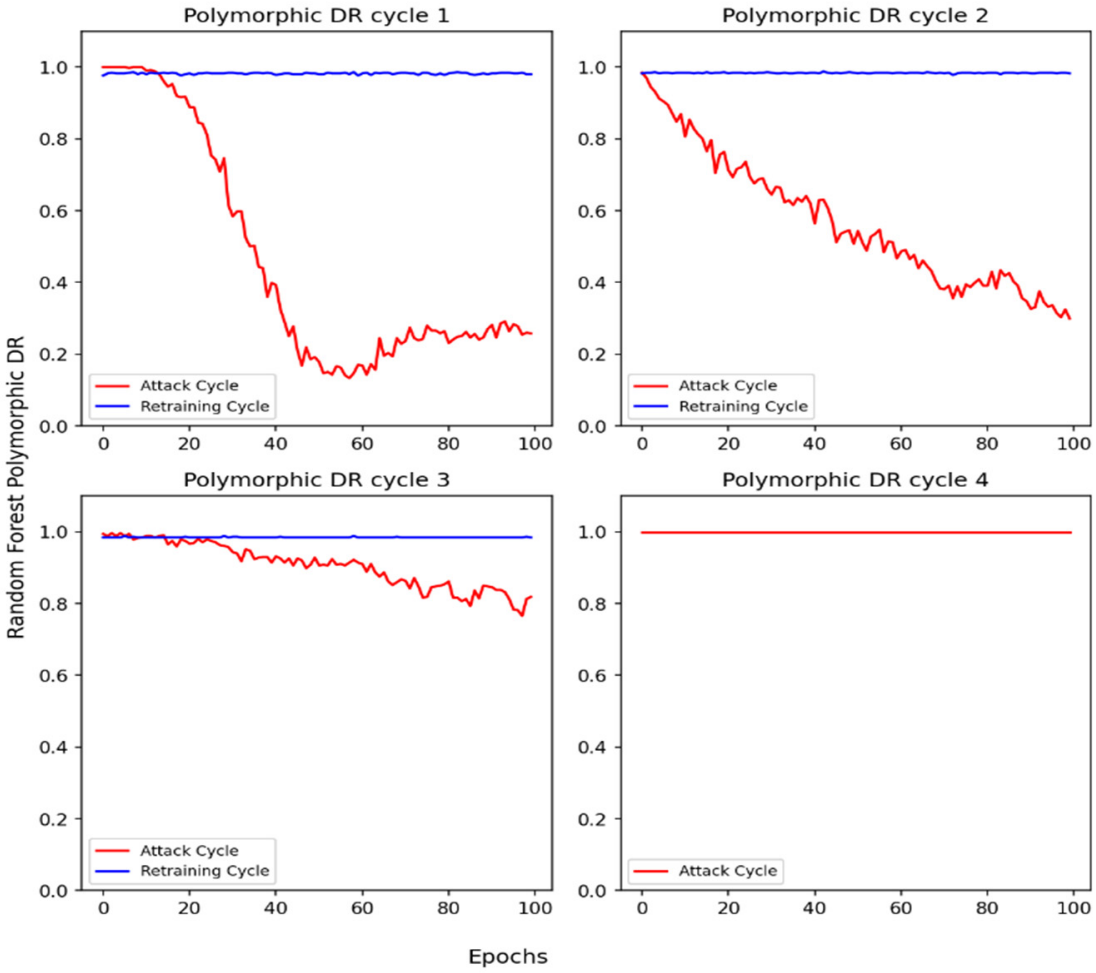

Figure 6. Polymorphic adversarial DDoS/DoS attack using Algorithm 3.

Figure 7 depicts the DR results of the Random Forest IDS against polymorphic attacks synthesized by implementing Algorithm 4. We observe that the DR is dropped to almost zero after 100 epochs in cycle 1. After retraining, the DR improves to 70% which is lower as compared to cycle 1 in Figure 6. Gradually, after multiple attacks and retraining cycles (2–3), the IDS can capture the pattern of polymorphic attacks and can no longer be evaded.

Figure 8 represents the evaluation of the attacker model based on the IDS Evasion Success Rate (ESR). It measures the success of the polymorphic adversarial attack in evading detection by the IDS. As shown in Figure 8, ESR in the adversarial cycle and attack cycles 1–2 is higher indicating that the attacker can successfully evade detection in these attack cycles. However, ESR reduces gradually after continuous updating and retraining of the IDS, indicating that it has learned the pattern and can now identify the attack with higher detection rates.

Table 4 shows the performance evaluation and comparison of overall metric values, such as Recall (TPR), TNR, and F1-score for the Random Forest IDS model on different adversarial polymorphic attacks. The results show that during the attack cycle, the attacker is successful in evading detection by the IDS as depicted by the low TPR values. After each retraining phase, the IDS eventually learns the attack pattern until the attacker can no longer launch successful attacks on the victim.

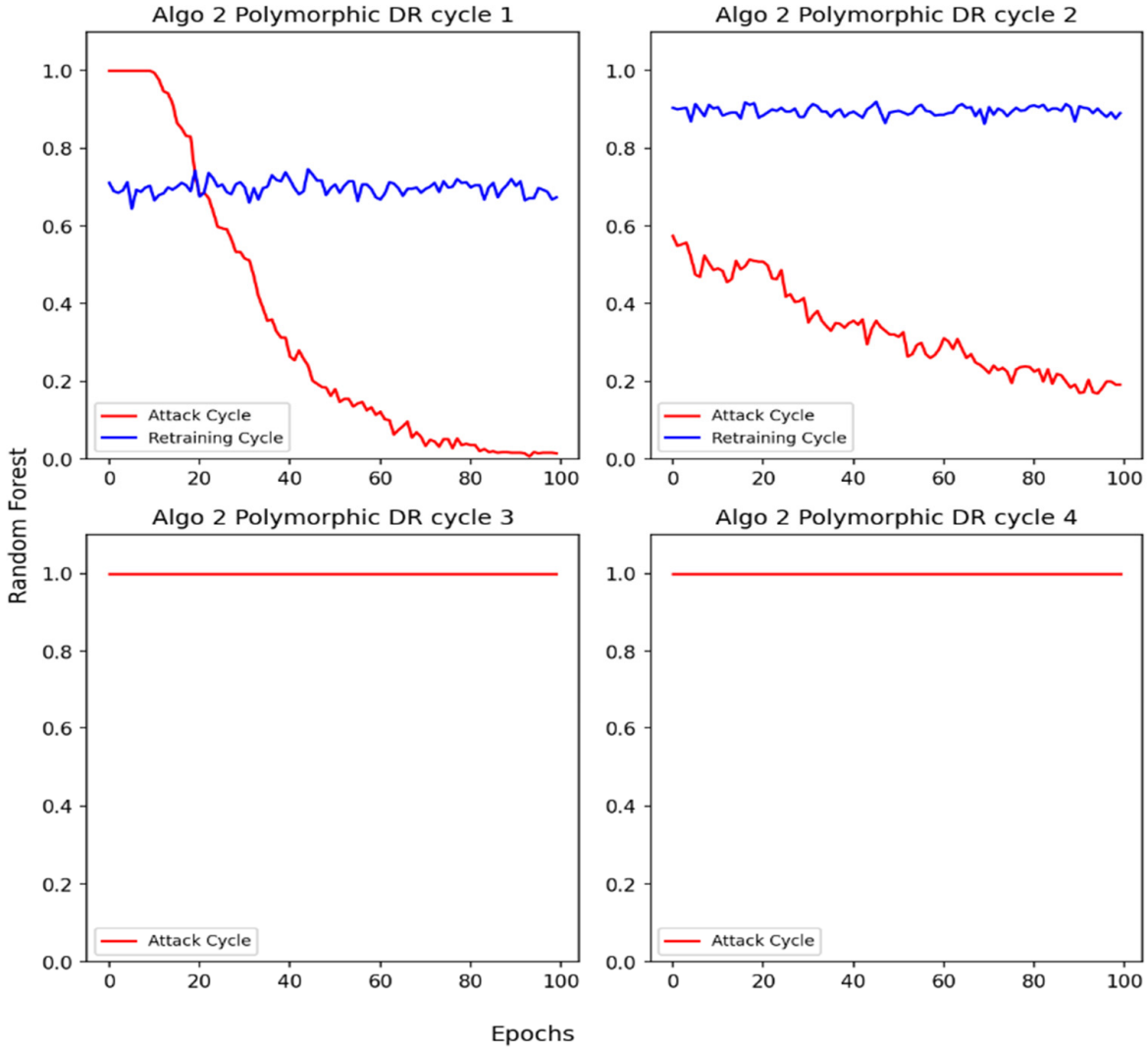

Figure 7. Polymorphic adversarial DDoS/DoS attack using Algorithm 4.

The Random Forest IDS is also evaluated against the polymorphic attacks generated using Approach 2 [60]. The strength of these polymorphic attacks is measured using the Evasion Success Rate metric. In this case, multiple experimental scenarios are studied with different training data feature combinations (both manual and automatic feature selection). Six different experiments are performed with a different number of selected features for generating a polymorphic attack. For manual feature selection, 10 features are selected during the first test, followed by 20 features in the second test. Automated tests using Reinforcement learning were conducted with 40 features, 50 features, 60 features, and 76 features.

Figure 9 depicts the success rate for polymorphic adversarial attacks created using manual feature selection (20 features). Figure 10 depicts the polymorphic adversarial attack synthesized using automated feature selection (40 features) successfully evading the IDS. The red bars indicate the success of the polymorphic attack engine in evading IDS. As expected, each training session (yellow bar) improves the IDS detection rate against the attack. The number of cycles indicates how many times the attack generator is able to evade the IDS until the IDS is fully trained with all of the features or the attack engine runs out of features. Throughout our study, we also observed that using fewer features (e.g., five instead of 10) improves the evasion rates, but offers fewer options for synthesizing more polymorphic attacks.

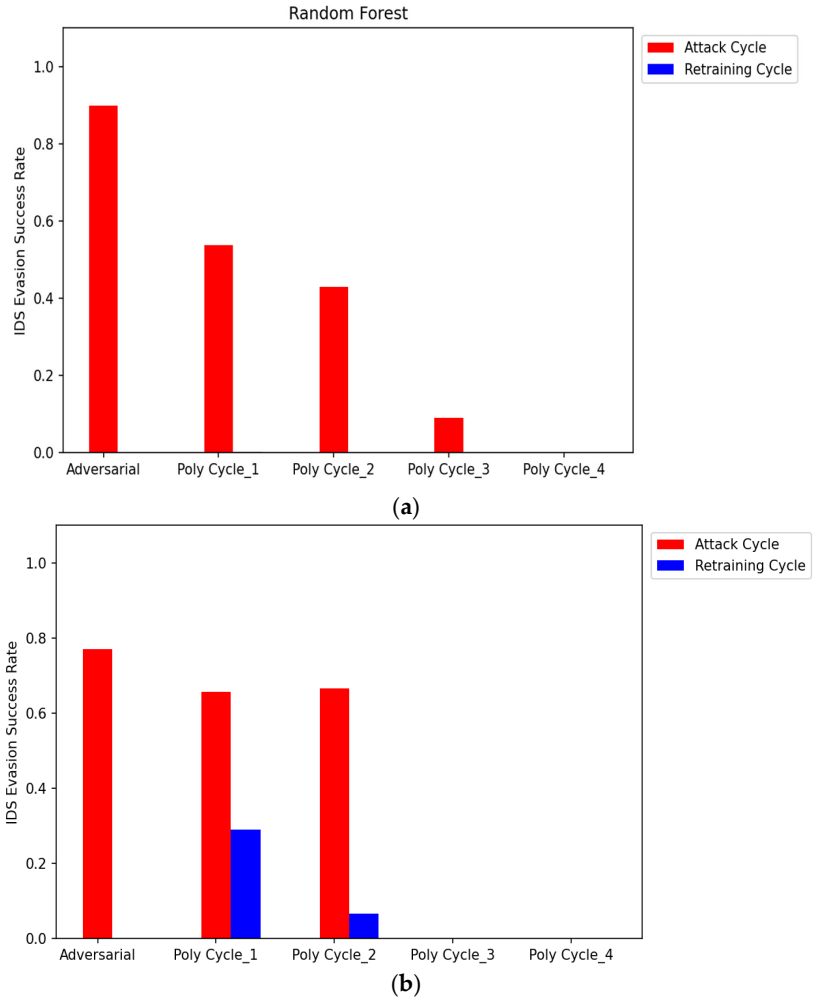

Figure 8. (**a**) Comparing the IDS evasion success rate for polymorphic adversarial attacks. The figure represents the results using Algorithm 3. (**b**) Comparing the IDS evasion success rate for polymorphic adversarial attacks. The figure represents the results using Algorithm 4.

Table 4. Performance evaluation and comparison for random forest IDS on different adversarial polymorphic attacks using Algorithms 3 and 4 (Approach 1).

Performance	Attack Type	Random Forest IDS	
	(Algorithm 3)	Attack Cycle	Retrain Cycle
Recall (TPR)	Adversarial	9.91%	98.40%
	Polymorphic 1	46%	98.20%
	Polymorphic 2	56.70%	98.30%
	Polymorphic 3	90.60%	98.40%
	Polymorphic 4	100%	N/A
TNR	Adversarial	98.60%	95.60%
	Polymorphic 1	98.60%	95.60%
	Polymorphic 2	98.60%	95.60%
	Polymorphic 3	98.60%	95.70%
	Polymorphic 4	98.60%	N/A
F1_Score	Adversarial	12.30%	98.60%
	Polymorphic 1	56.80%	98.50%
	Polymorphic 2	70.00%	98.50%
	Polymorphic 3	94.30%	98.60%
	Polymorphic 4	99.30%	N/A

Table 4. *Cont.*

Performance	Attack Type	Random Forest IDS	
	(Algorithm 4)	Attack Cycle	Retrain Cycle
Recall (TPR)	Adversarial	22.50%	98.40%
	Polymorphic 1	34.10%	69.70%
	Polymorphic 2	33.20%	89.60%
	Polymorphic 3	100%	N/A
	Polymorphic 4	100%	N/A
TNR	Adversarial	98.60%	95.50%
	Polymorphic 1	98.60%	95.70%
	Polymorphic 2	98.70%	95.60%
	Polymorphic 3	98.70%	N/A
	Polymorphic 4	98.60%	N/A
F1_Score	Adversarial	24.70%	98.50%
	Polymorphic 1	41.30%	81.60%
	Polymorphic 2	48.30%	93.80%
	Polymorphic 3	99.30%	N/A
	Polymorphic 4	99.30%	N/A

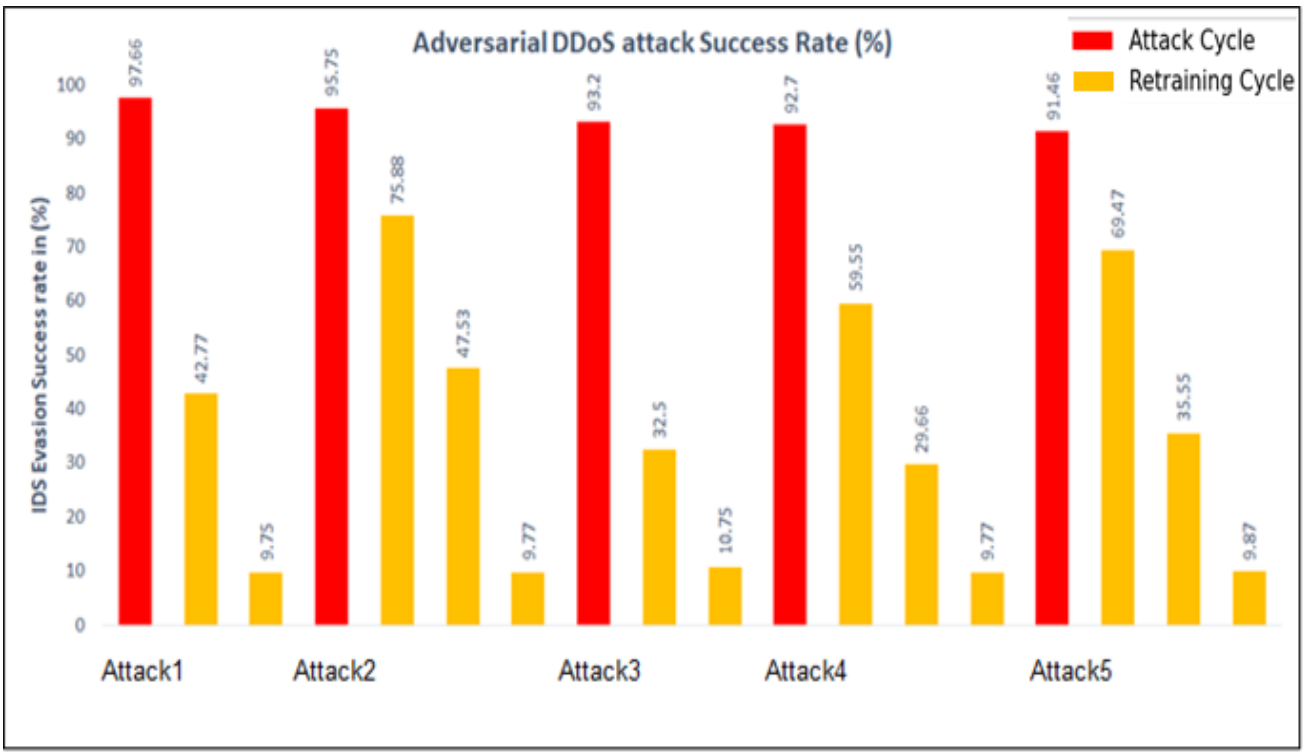

Figure 9. Polymorphic adversarial attacks using the manual feature selection (20 features).

7.2. Analysis and Discussion

To evaluate the performance of our WGAN-based framework comprehensively, several state-of-the-art machine learning models are employed as black-box IDS. The adopted machine learning models include Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Naïve Bayes (NB).

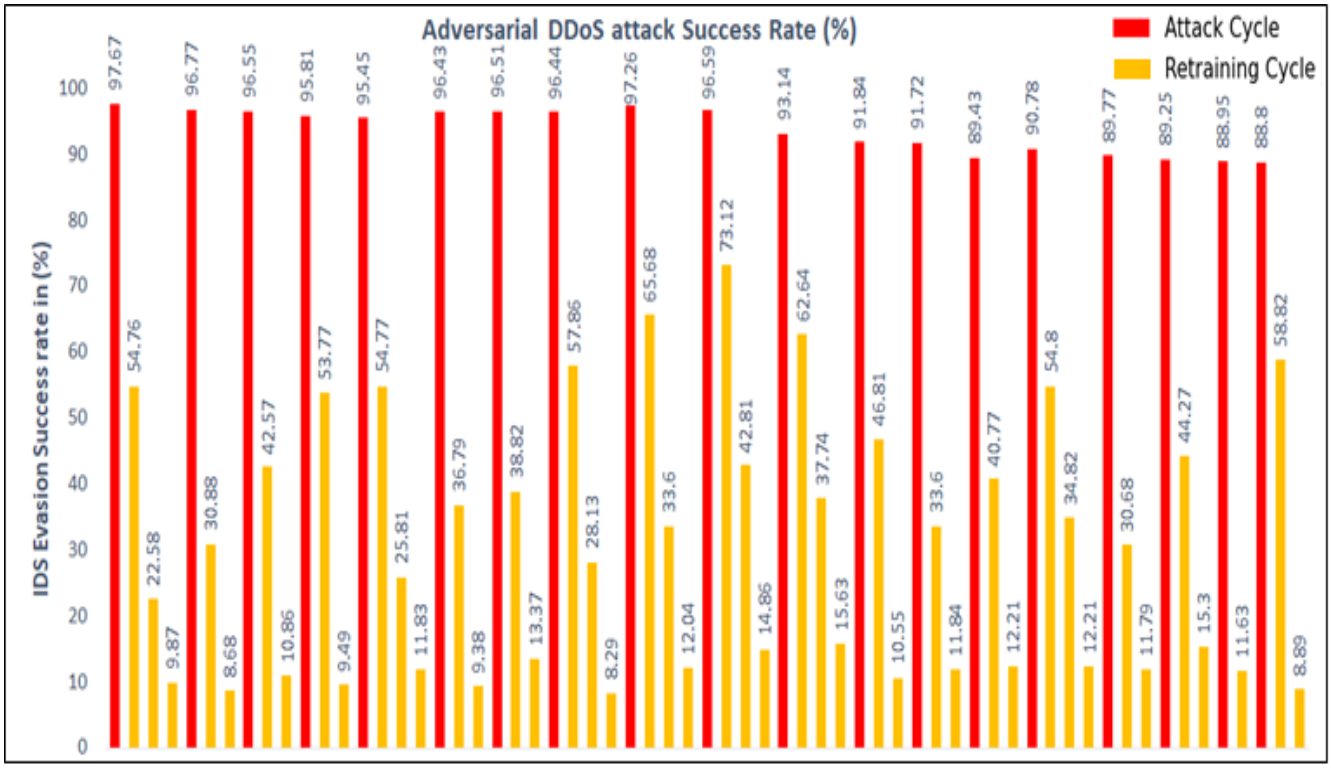

Figure 10. Polymorphic adversarial attacks using the automated feature selection (40 features).

Multiple IDS models are initially trained using CICIDS2017 data one by one, and then evaluated against adversarial polymorphic attacks using our framework. Figure 11 provides a comparative analysis of these models against adversarial polymorphic attacks. During the first attack cycle, the models are trained using the original training dataset and their performance is evaluated against adversarial attacks. As seen in all of the subfigures above, the adversarial detection rates of all the models are very low in the first phase. For SVM and NB, the adversarial detection rates are close to zero. The models are then retrained with previous adversarial data from the attack cycle and again evaluated in the retraining cycle. The results indicate improvement in IDS performances. In the next attack cycle, the attacker mutates the feature profile using the value-based approach (approach 1) and relaunches the attack on the IDS. During this phase, the detection rates go down again, yet show better results than phase 1. This process is repeated until the IDS cannot be evaded by the polymorphic attacker. Figure 11 shows that RF, DT, and KNN models do not need retraining for the last two phases since these IDSs can easily identify the attack, whereas SVM and NB still have lower detection rates in the last phase of the polymorphic attack. From Figure 11, we notice that the proposed WGAN framework for polymorphic adversarial training is effective in improving the performance of the machine learning-based black-box IDSs against rapidly evolving polymorphic attacks. Further performance improvement is expected in the future when the model is trained with other classes of polymorphic network attacks.

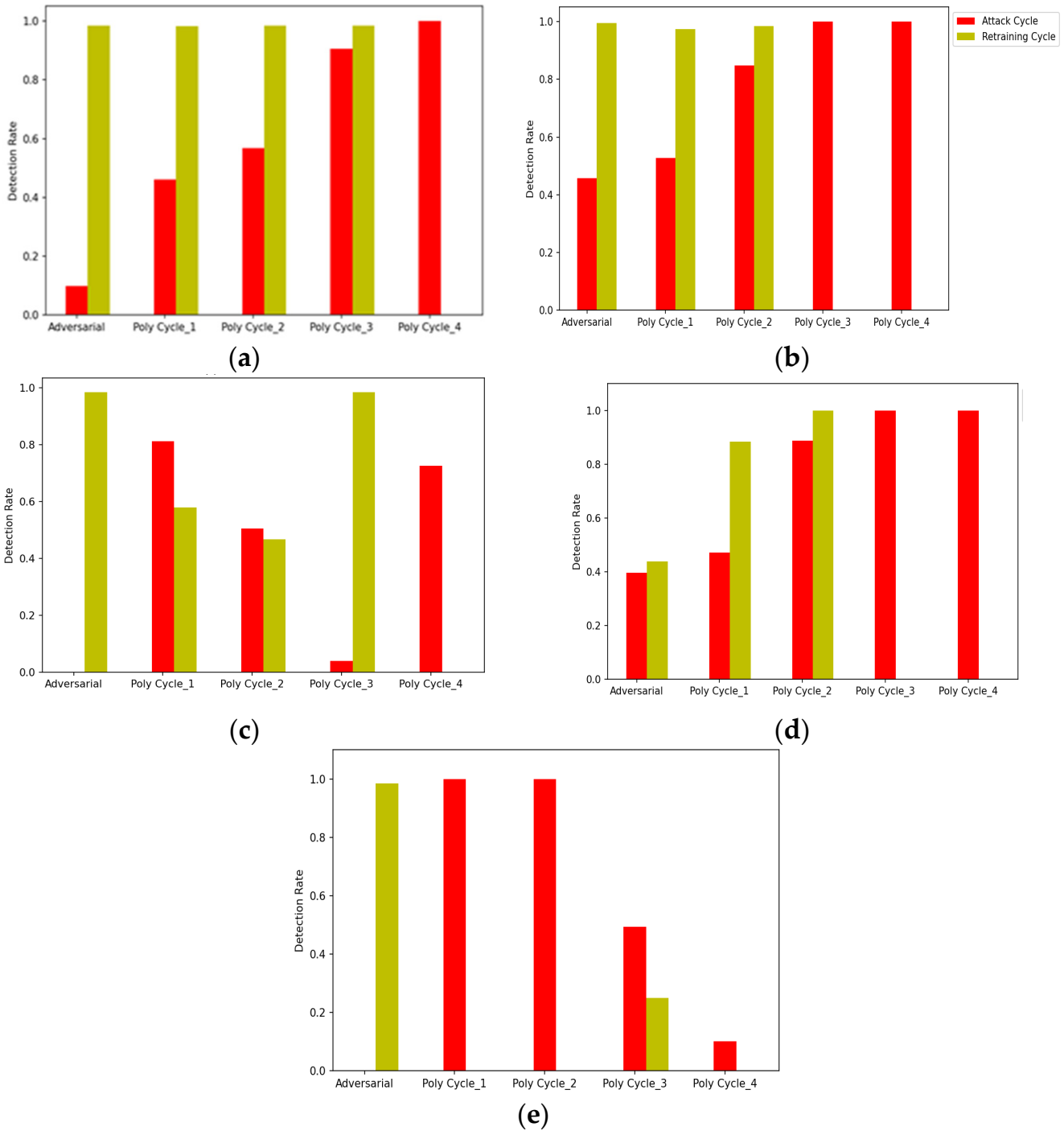

Figure 11. Comparative analysis of adversarial polymorphic attack detection rate for multiple IDS models. The adversarial polymorphic attacks are synthesized using Algorithm 3. (**a**) Random Forest (RF). (**b**) Decision Tree (DT). (**c**) Support Vector Machine. (**d**) K-Nearest Neighbor (KNN). (**e**) Naïve Bayes (NB).

8. Conclusions and Future Work

In this paper, we proposed a Wasserstein GAN-based framework to generate polymorphic adversarial DDoS/DoS attacks using a CICIDS2017 dataset. In order to synthesize these attacks, we applied two different approaches to change the feature profile of the attack. For the first approach, we employed a value-based feature mutation to synthesize new data for the attacker, which then generates a polymorphic attack. In the second approach, the feature selection-based method was employed to generate new data for the

attacker, which can in turn synthesizes a polymorphic attack. The second approach is based on both the manual as well as the automated feature selection.

From the results, we demonstrated that the Generator can produce polymorphic adversarial DDoS/DoS attacks to effectively evade the IDS during the initial phases. However, after several retraining steps, the IDS learns the pattern of attacks and eventually identifies all of the attacks. Additionally, we have provided an extensive comparison of multiple machine learning models against these attacks. The results indicate that the WGAN based adversarial training has improved the performance of several IDSs against polymorphic attacks.

In this paper, the work represented the introduction of a novel concept in network security, namely, polymorphic network attacks. Therefore, this work can be extended and built upon extensively for application to different security attack types, and IDS systems. While the proposed model has employed WGAN, there are several other variants of GAN, such as DCGAN [61], Conditional GAN [62], BiGAN [63], and Cycle GAN [64] that can be used to generate adversarial network attack data. On the IDS side, our results indicate that the polymorphic attackers can evade detection, albeit for a limited number of cycles. It is imperative to develop new techniques and models for the IDS that could, if not eliminate, at least shorten the window of vulnerability against polymorphic attacks.

Our work has focused on the signature-based IDS, primarily since the anomaly-based IDS is considered to have a high false positive rate, which would unnecessarily reduce the network throughput. It would be interesting to run our attack generator against the anomaly-based IDS, compare the results with the signature-based IDS, and evaluate the advantages and disadvantages.

Additionally, while our proposed model is generic enough to be used with any type of network attack, our performance evaluation and results were focused on a selected number of DoS/DDoS attacks. Furthermore, we would extend our research to employ multiple generators for each type of attack, and evaluate the performance of the IDS against all types of polymorphic adversarial network attacks.

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

Funding: This research was funded by NSERC Engage grant number EGP 529452-18, and start-up funding from California State University- Dominguez Hills. The APC was funded by California State University- Dominguez Hills.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Acknowledgments: The authors acknowledge and thank Deepa Kishore Malani for her contribution to this research.

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

References

1. Liao, H.-J.; Richard Lin, C.-H.; Lin, Y.-C.; Tung, K.-Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
2. Sabeel, U.; Heydari, S.S.; Mohanka, H.; Bendhaou, Y.; Elgazzar, K.; El-Khatib, K. Evaluation of Deep Learning in Detecting Unknown Network Attacks. In Proceedings of the 2019 International Conference on Smart Applications, Communications and Networking (SmartNets), Sharm El Sheik, Egyp, 17–19 December 2019; pp. 1–6.
3. Sabeel, U.; Heydari, S.S.; Elgazzar, K.; El-Khatib, K. Building an Intrusion Detection System to Detect Atypical Cyberattack Flows. *IEEE Access* **2021**, *9*, 94352–94370. [CrossRef]

4. Gadelrab, M.; Kalam, A.A.E.; Deswarte, Y. Manipulation of Network Traffic Traces for Security Evaluation. In Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops, Bradford, UK, 26–29 May 2009; pp. 1124–1129.
5. Skopik, F.; Settanni, G.; Fiedler, R.; Friedberg, I. Semi-synthetic data set generation for security software evaluation. In Proceedings of the 2014 Twelfth Annual International Conference on Privacy, Security and Trust, Toronto, ON, Canada, 23–24 July 2014; pp. 156–163.
6. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; Volume 2, pp. 2672–2680.
7. Yu, S.; Dong, H.; Liang, F.; Mo, Y.; Wu, C.; Guo, Y. SIMGAN: Photo-Realistic Semantic Image Manipulation Using Generative Adversarial Networks. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 734–738.
8. Wan, C.-H.; Chuang, S.-P.; Lee, H.-Y. Towards Audio to Scene Image Synthesis using Generative Adversarial Network. In Proceedings of the ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2018.
9. Yang, Y.; Dan, X.; Qiu, X.; Gao, Z. FGGAN: Feature-Guiding Generative Adversarial Networks for Text Generation. *IEEE Access* **2020**, *8*, 105217–105225. [CrossRef]
10. Zhang, J.; Yan, Q.; Wang, M. Evasion Attacks Based on Wasserstein Generative Adversarial Network. In Proceedings of the 2019 Computing, Communications and IoT Applications (ComComAp), Shenzhen, China, 26–28 October 2019; pp. 454–459.
11. Fogla, P.; Sharif, M.; Perdisci, R.; Kolesnikov, O.; Lee, W. Polymorphic Blending Attacks. In Proceedings of the USENIX Security Symposium, Berkeley, CA, USA, 11–13 August 2006.
12. Best, R. How AI Is Leading to More Business Phishing Attacks. Available online: https://www.infotech.co.uk/blog/how-ai-is-leading-to-more-business-phishing-attacks (accessed on 10 December 2021).
13. Mezic, A. Hacking the Hackers: Adversarial AI and How to Fight It. Available online: https://securityboulevard.com/2020/01/hacking-the-hackers-adversarial-ai-and-how-to-fight-it/ (accessed on 10 December 2021).
14. Yaltirakli, G. Slowloris: Low Bandwidth DoS tool. Available online: https://github.com/gkbrk/slowloris (accessed on 10 December 2021).
15. Seidl, J. GoldenEye DDoS Attack. Available online: https://github.com/jseidl/GoldenEye (accessed on 10 December 2021).
16. Hulk DDoS Attack. Available online: https://github.com/Mr4FX/Hulk-ddos-attack (accessed on 10 December 2021).
17. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN. *arXiv* **2017**, arXiv:1701.07875.
18. Lundberg, S.M.; Lee, S.-I. A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 4768–4777.
19. Rokade, M.D.; Sharma, Y.K. MLIDS: A Machine Learning Approach for Intrusion Detection for Real Time Network Dataset. In Proceedings of the 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 5–7 March 2021; pp. 533–536.
20. Singhal, A.; Maan, A.; Chaudhary, D.; Vishwakarma, D. A Hybrid Machine Learning and Data Mining Based Approach to Network Intrusion Detection. In Proceedings of the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 25–27 March 2021; pp. 312–318.
21. Srivastava, A.; Agarwal, A.; Kaur, G. Novel Machine Learning Technique for Intrusion Detection in Recent Network-based Attacks. In Proceedings of the 2019 4th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 21–22 November 2019; pp. 524–528.
22. Ertam, F.; Kilincer, L.F.; Yaman, O. Intrusion detection in computer networks via machine learning algorithms. In Proceedings of the 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 16–17 September 2017; pp. 1–4.
23. Bharathy, A.M.V.; Umapathi, N.; Prabaharan, S. An Elaborate Comprehensive Survey on Recent Developments in Behaviour Based Intrusion Detection Systems. In Proceedings of the 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), Gurgaon, India, 21–23 February 2019; pp. 1–5.
24. Musa, U.S.; Chhabra, M.; Ali, A.; Kaur, M. Intrusion Detection System using Machine Learning Techniques: A Review. In Proceedings of the 2020 International Conference on Smart Electronics and Communication (ICOSEC), Tamilnadu, India, 10–12 September 2020; pp. 149–155.
25. Acharya, T.; Khatri, I.; Annamalai, A.; Chouikha, M.F. Efficacy of Machine Learning-Based Classifiers for Binary and Multi-Class Network Intrusion Detection. In Proceedings of the 2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), Shah Alam, Malaysia, 26–26 June 2021; pp. 402–407.
26. Lei, M.; Li, X.; Cai, B.; Li, Y.; Liu, L.; Kong, W. P-DNN: An Effective Intrusion Detection Method based on Pruning Deep Neural Network. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–9.
27. Li, L.H.; Ahmad, R.; Tsai, W.C.; Sharma, A.K. A Feature Selection Based DNN for Intrusion Detection System. In Proceedings of the 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, Korea, 4–6 January 2021; pp. 1–8.

28. Chen, L.; Kuang, X.; Xu, A.; Suo, S.; Yang, Y. A Novel Network Intrusion Detection System Based on CNN. In Proceedings of the 2020 Eighth International Conference on Advanced Cloud and Big Data (CBD), Taiyuan, China, 5–6 December 2020; pp. 243–247.
29. Ho, S.; Jufout, S.A.; Dajani, K.; Mozumdar, M. A Novel Intrusion Detection Model for Detecting Known and Innovative Cyberattacks Using Convolutional Neural Network. *IEEE Open J. Comput. Soc.* **2021**, *2*, 14–25. [CrossRef]
30. Park, S.H.; Park, H.J.; Choi, Y. RNN-based Prediction for Network Intrusion Detection. In Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, 19–21 February 2020; pp. 572–574.
31. Sivamohan, S.; Sridhar, S.S.; Krishnaveni, S. An Effective Recurrent Neural Network (RNN) based Intrusion Detection via Bi-directional Long Short-Term Memory. In Proceedings of the 2021 International Conference on Intelligent Technologies (CONIT), Karnataka, India, 25–27 June 2021; pp. 1–5.
32. Hao, Y.; Sheng, Y.; Wang, J. Variant Gated Recurrent Units With Encoders to Preprocess Packets for Payload-Aware Intrusion Detection. *IEEE Access* **2019**, *7*, 49985–49998. [CrossRef]
33. Liu, C.; Liu, Y.; Yan, Y.; Wang, J. An Intrusion Detection Model With Hierarchical Attention Mechanism. *IEEE Access* **2020**, *8*, 67542–67554. [CrossRef]
34. Ali, K.; Boutaba, R. Applying kernel methods to anomaly based Intrusion Detection Systemss. In Proceedings of the 2009 Global Information Infrastructure Symposium, Hammamet, Tunisia, 23–26 June 2009; pp. 1–4.
35. Chae, Y.; Katenka, N.; DiPippo, L. An Adaptive Threshold Method for Anomaly-based Intrusion Detection Systems. In Proceedings of the 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 26–28 September 2019; pp. 1–4.
36. Chun-Hui, X.; Chen, S.; Cong-Xiao, B.; Xing, L. Anomaly Detection in Network Management System Based on Isolation Forest. In Proceedings of the 2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC), Wuhan, China, 19–21 April 2018; pp. 56–60.
37. Nguyen, T.Q.; Laborde, R.; Benzekri, A.; Qu'hen, B. Detecting abnormal DNS traffic using unsupervised machine learning. In Proceedings of the 2020 4th Cyber Security in Networking Conference (CSNet), Lausanne, Switzerland, 21–23 October 2020; pp. 1–8.
38. Pu, G.; Wang, L.; Shen, J.; Dong, F. A hybrid unsupervised clustering-based anomaly detection method. *Tsinghua Sci. Technol.* **2021**, *26*, 146–153. [CrossRef]
39. Kotani, G.; Sekiya, Y. Unsupervised Scanning Behavior Detection Based on Distribution of Network Traffic Features Using Robust Autoencoders. In Proceedings of the 2018 IEEE International Conference on Data Mining Workshops (ICDMW), Singapore, 17–20 November 2018; pp. 35–38.
40. Hwang, R.; Peng, M.; Huang, C.; Lin, P.; Nguyen, V. An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection. *IEEE Access* **2020**, *8*, 30387–30399. [CrossRef]
41. Kabir, M.A.; Luo, X. Unsupervised Learning for Network Flow Based Anomaly Detection in the Era of Deep Learning. In Proceedings of the 2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService), Oxford, UK, 3–6 August 2020; pp. 165–168.
42. Panos, C.; Xenakis, C.; Kotzias, P.; Stavrakakis, I. A specification-based intrusion detection engine for infrastructure-less networks. *Comput. Commun.* **2014**, *54*, 67–83. [CrossRef]
43. Mitchell, R.; Chen, I. Behavior Rule Specification-Based Intrusion Detection for Safety Critical Medical Cyber Physical Systems. *IEEE Trans. Dependable Secur. Comput.* **2015**, *12*, 16–30. [CrossRef]
44. Babu, M.J.; Reddy, A.R. SH-IDS: Specification Heuristics Based Intrusion Detection System for IoT Networks. *Wirel. Pers. Commun.* **2020**, *112*, 2023–2045. [CrossRef]
45. Kawai, M.; Ota, K.; Dong, M. Improved MalGAN: Avoiding Malware Detector by Leaning Cleanware Features. In Proceedings of the 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Okinawa, Japan, 11–13 February 2019; pp. 040–045.
46. Shahpasand, M.; Hamey, L.; Vatsalan, D.; Xue, M. Adversarial Attacks on Mobile Malware Detection. In Proceedings of the 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile), Hangzhou, China, 24–24 February 2019; pp. 17–20.
47. Xie, H.; Lv, K.; Hu, C. An Effective Method to Generate Simulated Attack Data Based on Generative Adversarial Nets. In Proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 1777–1784.
48. Ring, M.; Schlör, D.; Landes, D.; Hotho, A. Flow-based Network Traffic Generation using Generative Adversarial Networks. *Comput. Secur.* **2018**, *82*, 156–172. [CrossRef]
49. Lin, Z.; Shi, Y.; Xue, Z. IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection. *arXiv* **2021**, arXiv:1809.02077.
50. Shahriar, M.H.; Haque, N.I.; Rahman, M.A.; Alonso, M. G-IDS: Generative Adversarial Networks Assisted Intrusion Detection System. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 376–385.

51. Zhang, J.; Zhao, Y. Research on Intrusion Detection Method Based on Generative Adversarial Network. In Proceedings of the 2021 International Conference on Big Data Analysis and Computer Science (BDACS), Kunming, China, 25–27 June 2021; pp. 264–268.
52. Hui, J. GAN—DCGAN (Deep Convolutional Generative Adversarial Networks). Available online: https://jonathan-hui.medium.com/gan-dcgan-deep-convolutional-generative-adversarial-networks-df855c438f (accessed on 10 December 2021).
53. Google. Overview of GAN Structure—Generative Adversarial Networks. Available online: https://developers.google.com/machine-learning/gan/gan_structure (accessed on 10 December 2021).
54. Zhang, Z.; Li, M.; Yu, J. On the convergence and mode collapse of GAN. In Proceedings of the SIGGRAPH Asia 2018 Technical Briefs, Tokyo, Japan, 4–7 December 2018; p. 21.
55. Brownlee, J. How to Implement Wasserstein Loss for Generative Adversarial Networks. Available online: https://machinelearningmastery.com/how-to-implement-wasserstein-loss-for-generative-adversarial-networks/ (accessed on 10 December 2021).
56. Sharafaldin, I.; Lashkari, A.; Ghorbani, A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Madeira, Portugal, 22–24 January 2018.
57. PyTorch 1.6.0 Documentation. Available online: https://pytorch.org/docs/stable/torch.html (accessed on 10 December 2021).
58. Scikit-Learn Documentation: Machine Learning in Python. Available online: https://scikit-learn.org/stable/user_guide.html (accessed on 10 December 2021).
59. Pandas 1.1.2 Documentation. Available online: https://pandas.pydata.org/docs/ (accessed on 10 December 2021).
60. Chauhan, R. *Polymorphic Adversarial DDoS Attack on IDS Using GAN*; University of Ontario Institute of Technology: Oshawa, ON, Canada, 2020.
61. Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv* **2015**, arXiv:1511.06434.
62. Mirza, M.; Osindero, S. Conditional Generative Adversarial Nets. *arXiv* **2014**, arXiv:1411.1784.
63. Mutlu, U.; Alpaydın, E. Training bidirectional Generative Adversarial Network with hints. *Pattern Recognit.* **2020**, *103*, 107320. [CrossRef]
64. Zhu, J.; Park, T.; Isola, P.; Efros, A.A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2242–2251.