I.J. Wireless and Microwave Technologies, **2022, 1, 12-21**
Published Online February 2022 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijwmt.2022.01.02

Adversarial Machine Learning Attacks and Defenses in Network Intrusion Detection Systems

Amir F. Mukeri
AISSMS College of Engineering, Pune, 411001, India.
E-mail: mukeriamir@gmail.com

Dwarkoba P. Gaikwad
AISSMS College of Engineering, Pune, 411001, India.
E-mail: dpgaikwad@aissmscoe.com

Received: 24 October 2021; Accepted: 15 December 2021; Published: 08 February 2022

Abstract: Machine learning is now being used for applications ranging from healthcare to network security. However, machine learning models can be easily fooled into making mistakes using adversarial machine learning attacks. In this article, we focus on the evasion attacks against Network Intrusion Detection System (NIDS) and specifically on designing novel adversarial attacks and defenses using adversarial training. We propose white box attacks against intrusion detection systems. Under these attacks, the detection accuracy of model suffered significantly. Also, we propose a defense mechanism against adversarial attacks using adversarial sample augmented training. The biggest advantage of proposed defense is that it doesn't require any modification to deep neural network architecture or any additional hyperparameter tuning. The gain in accuracy using very small adversarial samples for training deep neural network was however found to be significant.

Index Terms: Network Intrusion Detection System, Adversarial Machine Learning, Robust Machine Learning, Network Security, Deep Neural Network.

1. Introduction

Increase in sophistication of network hacks have led to use of more sophisticated and intelligent counter attack tools.

One of such tools is a Network Intrusion Detection Systems (NIDS). NIDS monitors all incoming network traffic,

inspects the packet headers and metadata, and filters out malicious data packets. Traditionally NIDS relied on attack signatures to detect the malicious content. However, as the network attacks became more and more sophisticated NIDS are now relying on machine learning algorithms to detect zero day attacks. Machine Learning(ML) algorithms used by NIDS range from simple Decision Trees, Random Forests to complex Deep Neural Networks (DNN) models. However, ML algorithms are not a panacea for all types of attacks and they themselves could be victim of Adversarial Example attacks. Adversarial examples are malicious inputs to ML models with an intention of deceiving the ML models to misclassify input examples.

Most of the current research in adversarial machine learning has been focused on computer vision tasks such image classification or object detection using Convolutional Neural Network (CNN). Our main contributions through this article is to apply adversarial machine learning attacks specifically, adversarial example attacks against NIDS based on deep neural networks. Additionally, we propose the defense mechanism against these attacks using adversarial example augmented training of neural network.

Rest of the paper is organized as follows. In section 2 we present current research in the area of applying deep learning and adversarial machine learning techniques to the problem of Network Intrusion Detection. Section 3, presents proposed methodology for adversarial example attacks. Section 4 details the dataset used for experimental evaluation of proposed methodology. Section 5 provides the results and discussion followed by conclusion.

2. Related Work

Network security is fast moving and actively researched area. Network intrusion detection systems are no exception. Use of machine learning to increase effectiveness of NIDS has been actively researched and deployed since last more than a decade. This section provides high level overview of the different machine learning techniques

proposed in research literature as well as looks at the various adversarial attacks on these systems and defenses against adversarial attacks.

For the literature survey, search was done using keywords as, “Network Intrusion Detection System”, “Machine Learning for Network Intrusion Detection System”, “Adversarial Machine Learning”, “Defenses against Adversarial Machine Learning”.

2.1. Machine Learning Based NIDS

In this section, we outline recent work done creating NIDS system using various Machine Learning(ML) approaches ranging from decision trees, Support Vector Machines(SVM), Artificial Neural Networks(ANN) as well as ensemble techniques such as Random Forests.

Sakr et al. [1] proposed using Support Vector Machine (SVM) for the classification of benign and malicious traffic based on the feature selection by using population based algorithms such as Genetic Algorithms, Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO) algorithm in combination with Principal Component Analysis (PCA), and Correlation Feature Selection (CFS) were used. They relied on NSL-KDD data for evaluating the proposed classifier against other methods of feature selection. They found that NIDS classification using ABC outperformed other methods of feature selection. In another work, authors used similar approach for selecting most relevant feature using Binary-PSO and obtained significant results [2].

Barot et al. [3] used Naïve Bayes and Decision Table classification technique for implementing NIDS. For feature selection Chisquare, Information Gain based and Correlation based methods used. Instead of binary classification they formulated as a problem involving multiple output classes. Correlation based feature selection (CFS) algorithms outperformed other methods of feature selection with decision table classifier.

Anomaly detection techniques has been popular in NIDS. Karuppanchetty et al. [4] used anomaly detection using artificially augmented dataset for training purpose.

For NSL KDD dataset Panda et al. [5] conducted various experiments on using classic machine learning algorithms. Algorithms applied include decision trees (J48), Naive Bayes and their variants as well as ensemble ML techniques such as Naive Bayes with Ada boost. With ensemble method, Discriminative Multinomial Naive Bayes + N2B got detection accuracy of 96.5%.

Considering the high dimensionality of the data much of the research work also has gone into selecting most appropriate features for training the ML model. Wang et al. used Support Vector Machines (SVM) models and for feature selection used Efficient Correlation Based Feature Selection (ECOFS) method [6]. In this method, each feature's contribution to the target class computed using Information Gain. They have found the performance of the SVM based IDS to have better performance compared to features selected using conventional Pearson Correlation Coefficient (PCC) based models. Mukeri et al. [7] used Principal Component Analysis (PCA) for feature reduction for NSL KDD data followed by SVM classifier that showed an accuracy of 85% on training data and 83% on PCA processed data.

Tao et al. [8] proposed FWP-SVM-genetic algorithm (GA) (feature selection, weight, and parameter optimization of support vector machine based on the genetic algorithm). In this work, Tao et al. have used Genetic Algorithm (GA) for feature selection using novel fitness function that includes classification accuracy, number of features and the number of support vectors to choose the features for training SVM based classifier. They found reduction in error rate and improved classification time.

Vijayanand et al. [9] devised Whale Optimization Algorithm (WOA) for feature selection. Using this algorithm, for CICIDS2017 dataset features were reduced from 77 to 35 and for ADFA-LD dataset it was reduced from 44 to 25. They obtained accuracy of 95.91% on test data on CICIDS2017 and 94.44% on ADFA-LD.

Tama et al. [10] introduced hybrid feature selection method by making use of three bio-inspired algorithms viz. Particle Swarm Optimization (PSO), Ant Colony (AC) algorithm, and genetic algorithms. For NSL KDD dataset 37 out of 42 features and for UNSW-NB15 19 features out of 42 features were selected using this hybrid approach. A two-level classifier ensemble based on rotation forest and bagging, is proposed. They obtained accuracy of 85.85 % on NSL KDD dataset and 95.28% on UNSW-NB15 dataset.

Since most of the real-world network traffic has very less to no malicious network traffic. This gives rise to an imbalanced dataset. Yulianto et al. [11] used Synthetic Minority Oversampling Technique (SMOTE) to address this issue. They have used Principal Component Analysis (PCA), and Ensemble Feature Selection (EFS) to feature selection and AdaBoost classifier for CICIDS 2017 dataset and obtained F1 Score of 81.83%.

More recently deep neural network based methods have been leveraged for NIDS. Since deep learning is representational learning it doesn’t require feature selection. This is the biggest advantage of using deep learning approach over other traditional statistical machine learning approaches. Yin et al. [12] developed an NIDS based on Recurrent Neural Networks (RNN) and tested it on the NSL KDD dataset. Unlike feed forward NN, RNN remembers prior information and applies it to the current input data.

Convolution Neural Networks (CNN) are the specialized NN that have seen great success in the area of computer vision. Vinayakumar et al. [13] modeled the TCP/IP packets as time event sequences and used more complex CNN variants such as CNN-recurrent neural network (CNN- RNN), CNN-long short-term memory (CNN- LSTM) and CNN-

gated recurrent unit (GRU). They showed that it also outperforms traditional machine learning approaches on NSL KDD dataset. Similar approach was taken by (P. Sun et al. 2020) and yielded accuracy of 98.67% on CICIDS2017.

Yuan et al. [14] proposed a system for detection of DDoS attacks. In this method, they have used Bag of Words for text fields. In order to consider the temporal aspect of the network traffic flow they argue that RNN is better architectural choice for detecting attacks. They used Bidirectional RNN with both LSTM and GRU. The dataset used was ISCX2012. They reported reduced error rate compared to other deep learning or machine learning methods. For the class of NIDS that depends on signature based malware detection Sohi et al. [15] proposed RNN to generate new examples for testing and improving NIDS.

To address the issue of high dimensionality Liu et al. [16] proposed the use of Sparse Auto Encoder for compressing the traffic flows and classification using Random Forest. They also further added the compressed flows to a database for later reconstruction of original traffic or for debugging purpose through a feedback mechanism. This system was tested using CICIDS2017 and was found to have better detection accuracy. Similar feature compression technique was employed Yue et al. [17] using CNN.

Similar to feature selection for traditional ML approaches, DNN has the need for hyperparameter tuning for better generalization to unseen data. Hyperparameters are related to neural network architecture such as number of layers, number of neurons in each layer, activation function and learning rate. This field is also referred to as Neural Architecture Search (NAS). Extensive survey on this topic was done by Elesken et al. [18]. In the context of NIDS, Gaikwad et al. [19] applied various optimization techniques such as Bayesian Optimization, Hyperband, Random Search and Grid Search methods to design an optimal neural architecture for DNN based NIDS. They reported highest test accuracy of 83% on NSL KDD dataset using Bayesian Optimization.

Gurung et al. [20] used sparse autoencoder for feature selection followed by logistic regression for classification. This approach helped in avoiding entire feature selection step in the development of intelligent NIDS.

2.2. Adversarial Attack On Machine Learning Based NIDS

Adversarial machine learning attacks are intended to fool ML models. Based on whether or not attacker has an access to training data, adversarial attacks can be classified as *poisoning* or *evasion* attacks.

In case of poisoning attacks, attacker modifies the training data such that the model is trained on the incorrect data and such trained model ends up misclassifying input data. In case of evasion attacks, attacker modifies or perturbs malicious input such that already trained model ends up misclassifying the input.

In case of evasion attacks, attacker tries to modify or perturb the malicious input data examples that needs to be evaded or needs to be classified as normal traffic in such a way that the classifier classifies it as normal traffic. However, the perturbations should be done in such a way that ML model still considers it as valid input example. For example, in case of network traffic simply modifying the protocol type from TCP to UDP without changing the corresponding fields in the packet would be useless since such packets would be rendered as invalid and might get rejected at the firewall or proxy stage itself and attack will not be successful. Perturbations to input traffic needs to consider constrained perturbations while the attributes or features that can perturbed without any bounds or constraints are called as Unconstrained. Various methods can be employed to generate such adversarial examples. Following sections provide overview of the most prominent adversarial attacks against DNN models.

Alhajjar et al. [21] used evolutionary computational algorithms viz. Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) as well as Generative Adversarial Network (GAN) on NSL KDD and UNSW-NB15 dataset [22]. GAN is a generative deep learning technique where generative NN generates fake samples while another discriminator NN tries to classify between real and fake samples They found that these perturbation techniques were successful in evading most of the ML based NIDS. Especially, decision trees and SVM based classifier suffered evasion rates greater than 90%. Hence, authors recommend not relying on SVM and DT based NIDS in production deployments. They found the highest evasion rate with GANs.

Rigaki and Garica [23] used GAN to generate malicious traffic that mimics normal traffic and goes undetected by Intrusion Prevention System (IPS). They specifically used RNN GAN to generate fake Facebook chat traffic that looks like a normal chat traffic to an IPS and delivered the malware to the infected hosts. GANs learned to generate Facebook chat traffic only using 217 traffic flows. This is then used to modify the malware parameters namely timing, duration and request size. Using this parameters malware generated adversarial examples that were then replayed at the target. Also, there is a feedback mechanism build into the attack mechanism that ensures that in case the traffic was detected and blocked by IPS than GAN is retrained in short interval with new examples that were detected and new adversarial examples are generated. The malware command and control system is deployed in Amazon Web Services (AWS) cloud and infected client machine in university lab along with GAN.

WGAN (Wasserstein GAN) is a variant of GAN [24]. Unlike GAN that uses gradient of loss function, WGAN uses Wasserstein distance to calculate the loss. This prevents the mode collapse and stability issue observed with GAN. WGAN was used by Lin et al. [25] to generate completely new adversarial samples from malicious samples. They found that detection for NSL KDD data was reduced below 1% for ML models based on Naive Bayes, MLP, DT, RF and KNN.

2.3. Research Objectives

As evident from literature survey most of the research on adversarial machine learning is focused on computer vision tasks which is the clear gap in the current work. Hence, we formulate following research objectives:

- What is the impact on the performance of various white box adversarial machine learning attacks from computer vision domain on deep learning models used in NIDS?
  - Can these attacks be carried out preserving network traffic validity?
- Adversarial example augmented training is found to be an effective defense mechanism against adversarial attacks in computer vision domain. Therefore, we pose a question, is adversarial training effective against adversarial example attacks in network security specifically, for deep learning based network traffic classifiers?

3. Proposed Methodology

This section describes various white box adversarial example attacks used against deep learning based classifiers. While most of these attacks were proposed in the domain of computer vision tasks we adapt them in order to be effective against classifiers used in NIDS.

3.1. Fast Gradient Sign Method (FGSM)

FGSM attack for image classifiers was proposed by Goodfellow et. al. [26]. These attacks use the gradient of neural network. In this technique, the sign of the perturbations in the direction gradient of loss function is computed to generate the adversarial examples,

$$\eta = \epsilon(\Delta\_x J(\theta, x, y)) \tag{1}$$

In above formula, ε is denotes the amount of perturbations needed, θ is the parameter of the model, x is input and y is the label and J denotes the loss function. This technique, tries to minimize the amount of perturbations needed to fool the model. On ImageNet it produced an error rate of 99.9%, its demonstration is shown in Fig 1.

Fig.1. Fast Gradient Sign method (FGSM): Adversarial example generated by FSGM using ImageNet was able to fool GoogLeNet [4]

As shown in Fig. 1, while the perturbations to the image are imperceptible to a human eye GoogLeNet misclassifies panda as gibbon with high confidence.

3.2 Jacobian based Saliency Map Attack (JSMA)

JSMA was proposed by Papernot et al. [27] as a way to generate adversarial examples using saliency values. These values denote how much the classification error will be introduced by modifying the input features. Saliency values are sorted in descending order and feature with highest saliency value is modified to check if it causes desired class change in model output. If not, then next feature is chosen for perturbing.

Rigaki [23] used both FSGM and JSMA attacks and found that Random Forest was most robust to these kinds of attacks on NSL KDD dataset [28]. However, in this work, this attack is tested against deep neural network based NIDS.

3.3. Carlini & Wagner Attack

Carlini & Wagner [29] proposed attack using the output of logit layer i.e. the layer before the final *softmax* layer. They essentially formulated generation of adversarial examples as an optimization problem as,

$$minimize\left|\left|\delta\right|\right|_p + c * f(x+\delta) \;\; such\ that\ x+\delta \in [0,1]^n \tag{2}$$

In (2), δ is the small perturbation to the original input example x and p is the norm of original and perturbed example. L2 norm i.e. Euclidean distance which is the preferred norm. The function f is label returned by the logit layer for the adversarial example. x + δ has the box constraint such that overall perturbation is bounded between 0 and 1. Using change of variable technique this box constrain is expressed as following and optimization is done for the variable w,

$$\delta_i = \frac{1}{2}(tanh(w_i)+1) - x_i \tag{3}$$

According to authors, this change of variables has a smoothing effect and reduces the problem of getting stuck in local optima and the value of c chosen to be small. Also, function f is reformulated to make the resulting image have minimum and imperceptible changes. Overall, L2 attack is formulated as,

$$minimize\ \left|\left|\frac{1}{2}(tanh(w)+1) - x\right|\right|_2^2 + c * f(\frac{1}{2}(tanh(w)+1) \tag{4}$$

$$with\ f(x') = max(max\{Z(x')_i : i \neq t\} - Z(x')_t - k$$

Carlini & Wagner attack is still considered to be one of the most reliable way of determining robustness of a neural network.

While there are many more methods to generate adversarial examples, it has been found that the examples that successfully attacked one ML model are most likely successful in fooling other ML models as well [9]. This is referred as transferability of adversarial examples.

3.4. Adversarial Training as a Defense against Adversarial Attacks

Goodfellow et al. [26] proposed the use adversarial examples to enhance the robustness of the ML models. They used adversarial examples generated using FGSM for training the models and obtained better performance for MNIST dataset. Similar approach was taken by Yin et al. [12] to enhance the performance of the intrusion detection classifier on NSL KDD dataset. In this method, a multiclass classifier is trained on both real training examples as well as adversarial fake examples generated by GAN. Authors call this enhanced framework as ID-GAN. Along with five original classes, the classifier is trained to classify the sixth class as fake example or not. They have found improved detection rate for such classifier. In order to enable this framework, authors also derive new loss function that takes into account modification to standard GAN architecture. Authors reported improved classification accuracy and F1 score.

Transferability blocking approach of training ML models using adversarial examples was suggested for image data by Hosseinin et al. [30]. Authors specifically address the issue of transferability of adversarial examples wherein adversarial examples that fool one classifier is able to fool other ML model irrespective of the architecture or training of the other model. In this approach, classifier is trained on normal and adversarial example with NULL label assigned to adversarial examples. The output of the classifier output shows the confidence of the model in that sample being an adversarial example. They have obtained zero error rate for adversarial attacks based on FGSM for MNIST digit data and significant improvement for GTSRB dataset.

For generating the adversarial examples Fast Gradient Sign Method (FGSM), Jacobian based Siliency Map Attack (JSMA), and Carlini and Wagner attack was used. For adversarial defense, adversarial training using all of these attack examples was used. Overall procedure for the adversarial training is illustrated in Algorithm 1. Also, while generating the adversarial examples care was taken so as not to make the examples into an invalid network traffic flow. To achieve these columns such as *protocol_type*, *service* and *flag* were not changed.

The optimization function for this algorithm is the value of probability that, the given example is malicious as returned by the trained neural network. The goal of the optimization is to reduce the probability of malicious example being classified correctly.

Algorithm 1. Adversarial Example Generation

Algorithm 1 Adversarial Example Generation
Input:training_examples, neural_network, adv_method
Output:adv_training_examples adv_training_examples = empty(training_examples)
for example in training_example
adv_example = empty()
if example == malicious
adv_example = get_adv_example(example, adv_method)
adv_example['protocol_type'] = example['protocol_type']
adv_example['service_type'] = example['service_type']
adv_example['flag'] = example['flag']
append_example(adv_training_examples, adv_example)
else
append_example(adv_training_example, example)
return adv_training_examples

4. Dataset

Most of the past and even some recent research work is based on testing effectiveness of machine learning algorithms on NSL KDD dataset [28]. It is refined version of dataset derived from KDD Cup 1999 dataset. This dataset has separate training and test dataset. There are network traffic flows of total 5 attack flows and normal traffic flow.

NSL KDD dataset has following different attacks:

1. Denial of Service(DoS): Includes attacks such as SynFlood and UDP Storm.
2. Probe: Includes attacks such as Nmap, Portsweep intended to gather information from target machine.
3. R2L: Includes attacks such as Xsnoop, Spy with an aim to gain remote administrative level access of the target machine.
4. U2R: Includes attacks such as Buffer overflow, Rootkit for privilege escalation attacks.

For this research, the problem was translated into a binary classification problem with all above four attack types classified in single attack class and rest as normal traffic with final distribution as shown in Table 1.

This dataset has total of 41 features such as Duration, Protocol Type, Src Byte, Destination Byte, etc. Table 2 shows all the features available in this dataset. More detailed analysis of the NSL KDD dataset is provided by Dhanabal et al. [31].

Table 1. NSL KDD Binary Class distribution

File	Normal Flows	Attack Flows
KDD Train+	67433 (53.45%)	58630 (46.54%)
KDD Test+	9710 (43.07%)	12833 (56.92%)

Table 2. Features in NSL KDD dataset

No.	Feature Name	Feature Description
1	Duration	Connection duration in seconds
2	Protocol_type	Type of the protocol used, e.g. TCP and UDP
3	Service	Network service on the destination, e.g. HTTP or telnet
4	Flag	Normal or error flag status of the connection
5	Src_bytes	Number of data bytes from source to destination
6	Dst_bytes	Number of data bytes from destination to source
7	Land	1 if the connection is from/ to the same host/port; 0 otherwise
8	Wrong_fragment	Number of ―wrong‖ fragments
9	Urgent	Number of urgent packets
10	Hot	Number of ―hot‖ indicators in the content such as: entering a system directory
11	Num_failed_logins	Number of failed login attempts
12	Logged_in	1 if successfully logged in; 0 otherwise

13	Num_compromised	Number of ―compromised conditions
14	Root_shell	1 if the root shell is obtained; 0 otherwise
15	Su_attempted	1 if ―su root‖ command attempted; 0 otherwise
16	Num_root	Number of ―root‖ accesses
17	Num_file_creations	Number of file creation operations
18	Num_shells	Number of logins of normal users
19	Num_access_files	Number of operations on access control files
20	Num_outband_cmds	Number of outbound commands in FTP session
21	Is_hot_login	1 if the login belongs to the ―hot‖ list i.e., root or admin; 0 otherwise
22	Is_guest_login	1 if the login is a ―guest‖ login; 0 otherwise
23	Count	Number of connections to the same destination host in the past 2 seconds
24	Srv_count	Sum of connections to the same service in the past 2 seconds
25	Serror_rate	% of connections that have ―SYN errors among the connections aggregated in count (23)
26	Srv_serror_rate	% of connections that have ―SYN errors among the connections aggregated in srv_count (24)
27	Rerror_rate	% of connections that have ―REJ errors among the connections aggregated in count (23)
28	Srv_rerror_rate	% of connections that have ―REJ errors among the connections aggregated in srv_count (24)
29	Same_srv_rate	% of connections to the same service
30	Diff_srv_rate	% of connections to different services
31	Srv_diff_host_rate	% of connections to different hosts
32	Dst_host_count	Sum of connections to the same destination IP address
33	Dst_host_srv_count	Sum of connections to the same destination port number
34	Dst_host_same_srv_rate	% of connections that were to the same service, among the connections aggregated in dst_host_count
35	Dst_host_diff_srv_rate	% of connections that were to different services, among the connections aggregated in dst_host_count
36	Dst_host_same_src_port_rate	% of connections that were to the same source port, among the connections aggregated in dst_host_srv_count
37	Dst_host_srv_diff_host_rate	% of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count
38	Dst_host_serror_rate	% of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in dst_host_count
39	Dst_host_srv_serror_rate	% of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count
40	Dst_host_rerror_rate	% of connections that have activated the flag REJ, among the connections aggregated in dst_host_count
41	Dst_host_srv_rerror_rate	% of connections that have activated the flag REJ, among the connections aggregated in dst_host_srv_count
42	Class label	Connection behaviour label

The NIDS itself is based on DNN. Hyperparameters tuning was done using various optimization methods to arrive at optimal neural network architecture [19].

5. Experimental Results

NIDS classifier was built using DNN using Keras package on Google Colab environments. GPU instances from this environment were used for all the experiments. Also, tuning of the hyperparameters was done to obtain the most effective DNN. Data was pre-processed using Standard Scaling method as shown in following equation,

$$x_i = \frac{x_i - \mu}{\sigma} \tag{5}$$

Here, x_i is a training example, μ is mean of the x column and σ is the standard deviation. For categorical columns such as Protocol type, Service and Flag, Categorical En- coding was used that basically assigns numeric codes to the categorical values, for example, tcp as 1, udp as 2, etc. Class labels were assigned values as 0 for normal traffic flow and 1 for an intrusion attack

5.1. Performance Against Adversarial Attacks

After training the DNN on clean training data, it was subjected to adversarial attacks. The performance of this DNN based model on unseen test data for various adversarial attacks is shown in Table 3.

Table 3. Performance of ANN on test data based NIDS against various adversarial example attacks

Attack	Accuracy	Drop in Accuracy	AUC Score	Drop in AUC Score
Clean (No Attack)	85.945	(baseline)	0.881	(baseline)
FGSM	71.892	**14.054**	0.877	0.005
JSMA	77.297	8.649	**0.954**	-0.072
Carlini & Wagner L2	82.162	4.324	0.018	**0.863**

As shown in Table 3, adversarial example attacks using FGSM, JSMA and Carlini & Wagner L2 attack causes considerable drop in accuracy of the DNN classifier. Also, similar effect was seen on the AUC score as expected. This is further visualized in Fig. 2. FGSM attacks had the most drops in the accuracy and is also fastest among these attacks types.

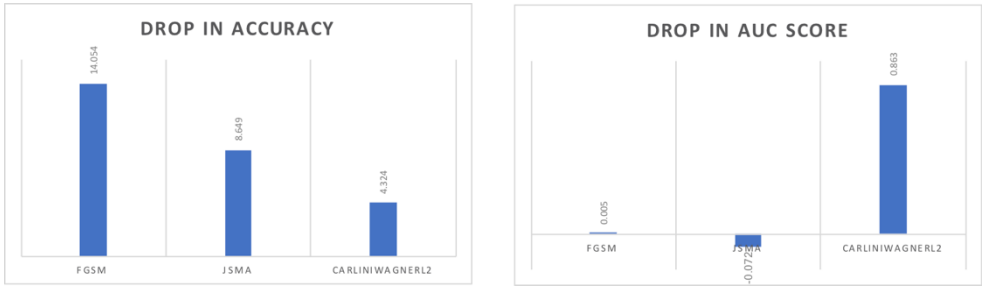

Fig.2. Drop in Accuracy and AUC score as a result of various adversarial example attacks

5.2. Adversarial Training

In order to create a NIDS which is robust to adversarial examples, combination of original clean training examples and small number of adversarial examples are used to train the DNN. The adversarial examples used for training are very small in number i.e. only 85 malicious examples that are *adversarially* generated using various attack types. DNN is trained using the adversarial examples and then it is evaluated against clean test dataset as well adversarial examples. As a result of the adversarial training, DNN accuracy improves as shown in Table 4. Most gain in accuracy is obtained by training with Carlini and Wagner L2 adversarial examples. Fig. 3 shows gain in accuracy after training DNN using various adversarial examples.

Table 4. Performance of ANN on test data based NIDS against various adversarial example attacks

Attack	Gain in accuracy Post FGSM Adv. Training	Gain in accuracy Post FGSM Adv. Training	Gain in accuracy Post FGSM Adv. Training
Clean (No Attack)	1.081	2.162	**2.162**
FGSM	12.973	4.865	**15.135**
JSMA	12.973	13.514	**14.595**
Carlini & Wagner L2	5.405	3.243	**6.486**

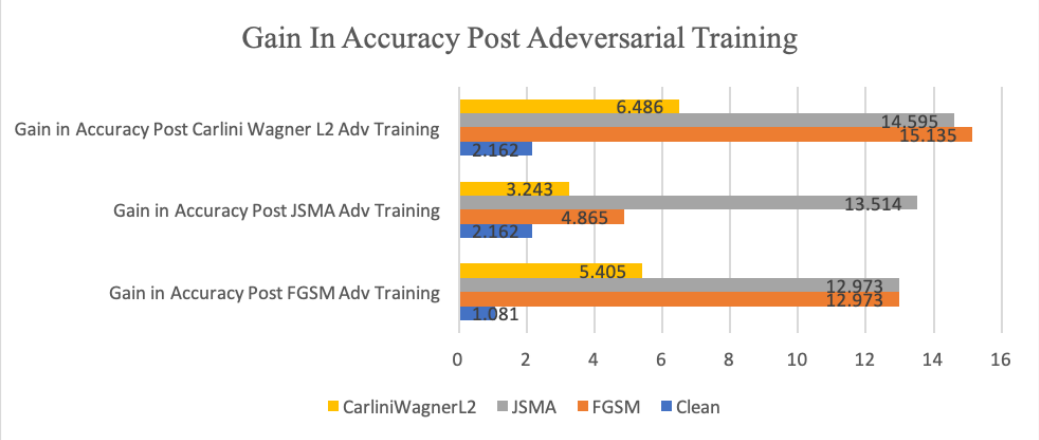

Fig.3. Comparison of accuracy gains after adversarial training.

This gain in accuracy is close the accuracy reported by other more complex method involving Generative Adversarial Network (GAN) which the proposed method achieves by addition of only 85 training samples to original training set.

Above results show the simplicity and effectiveness of the adversarial training.

6. Conclusion

Network intrusion detection system is an integral part of defense against ever increasing cyber threats. Deep Neural Network based NIDS is a critical component for combating zero day attacks that can't be detected with traditional signature based methods. However, as DL models are used in these systems they also become potential victim to adversarial machine learning attacks. The objective of these attacks is to fool Deep Learning based classifiers and make them commit mistakes. Currently most of the adversarial ML attacks and defenses are focused on image based models. In this work, we evaluated DL based NIDS with various state of the art ML attacks such as FGSM, JSMA, and Carlini & Wagner attacks.

Secondly, we trained the neural network based NIDS on very small number of adversarial examples generated using these algorithms. We found that, the neural networks trained on the adversarial examples are more robust to adversarial attacks and are difficult to compromise. Also, adversarial training examples generated using one type of algorithm provide effective robustness against other attacks as well. Specifically, we found that the training data augmented with adversarial examples generated using Carlini and Wagner L2 attack are most effective in making NIDS more robust against adversarial evasion attacks.

References

[1] Mahmoud M. Sakr, Medhat A. Tawfeeq, Ashraf B. El-Sisi, "An Efficiency Optimization for Network Intrusion Detection System", International Journal of Computer Network and Information Security(IJCNIS), Vol.11, No.10, pp.1-11, 2019.DOI: 10.5815/ijcnis.2019.10.01.

[2] Mahmoud M. Sakr, Medhat A. Tawfeeq, Ashraf B. El-Sisi,"Network Intrusion Detection System based PSO-SVM for Cloud Computing", International Journal of Computer Network and Information Security(IJCNIS), Vol.11, No.3, pp.22-29, 2019.DOI: 10.5815/ijcnis.2019.03.04.

[3] Virendra Barot, Sameer Singh Chauhan, Bhavesh Patel,"Feature Selection for Modeling Intrusion Detection", International Journal of Computer Network and Information Security(IJCNIS), vol.6, no.7, pp.56-62, 2014. DOI: 10.5815/ijcnis.2014.07.08.

[4] Chockalingam Karuppanchetty, William Edmonds, Sun-il Kim, Nnamdi Nwanze,"Artificially Augmented Training for Anomaly-based Network Intrusion Detection Systems", International Journal of Computer Network and Information Security(IJCNIS), vol.7, no.10, pp.1-14, 2015.DOI: 10.5815/ijcnis.2015.10.01.

[5] M. Panda, A. Abraham, S. Das and M. R. Patra, "Network intrusion detection system: A machine learning approach," Intelligent Decision Technologies, vol. 5, p. 347–356, 2011.

[6] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," IEEE access, vol. 6, p. 1792–1806, 2017.

[7] A. Mukeri and D. P. Gaikwad, "Support Vector Machine and Principal Component Analysis for Intrusion Detection System," i-Manager's Journal on Software Engineering, vol. 14, p. 42, 2020.

[8] P. Tao, Z. Sun and Z. Sun, "An improved intrusion detection algorithm based on GA and SVM," Ieee Access, vol. 6, p. 13624–13631, 2018.

[9] R. Vijayanand and D. Devaraj, "A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network," IEEE Access, vol. 8, p. 56847–56854, 2020.

[10] B. A. Tama, M. Comuzzi and K.-H. Rhee, "TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," IEEE Access, vol. 7, p. 94497–94507, 2019.

[11] A. Yulianto, P. Sukarno and N. A. Suwastika, "Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset," in Journal of Physics: Conference Series, 2019.

[12] C. Yin, Y. Zhu, S. Liu, J. Fei and H. Zhang, "Enhancing network intrusion detection classifiers using supervised adversarial training," The Journal of Supercomputing, p. 1–30, 2019.

[13] R. Vinayakumar, K. P. Soman and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017.

[14] X. Yuan, C. Li and X. Li, "DeepDefense: identifying DDoS attack via deep learning," in 2017 IEEE International Conference on Smart Computing (SMARTCOMP), 2017.

[15] S. M. Sohi, J.-P. Seifert and F. Ganji, "RNNIDS: Enhancing network intrusion detection systems through deep learning," Computers & Security, vol. 102, p. 102151, 2021.

[16] L. Liu, J. Lin, P. Wang, L. Liu and R. Zhou, "Deep Learning-Based Network Security Data Sampling and Anomaly Prediction in Future Network," Discrete Dynamics in Nature and Society, vol. 2020, 2020.

[17] W. Yue, J. Yiming and L. Julong, "A Fast Deep Learning Method for Network Intrusion Detection Without Manual Feature Extraction," in Journal of Physics: Conference Series, 2021.

[18] T. Elsken, J. H. Metzen, F. Hutter and others, "Neural architecture search: A survey.," J. Mach. Learn. Res., vol. 20, p. 1–21, 2019.

[19] D. P. Gaikwad and A. Mukeri, "Fine Tuned Deep Neural Networks for Intrusion Detection System," Journal of Network Security Computer Networks, vol. 6, p. 9, 2020.

[20] S. Gurung, M. K. Ghose and A. Subedi, "Deep learning approach on network intrusion detection system using NSL-KDD dataset," International Journal of Computer Network and Information Security, vol. 11, p. 8–14, 2019.
[21] E. Alhajjar, P. Maxwell and N. D. Bastian, "Adversarial Machine Learning in Network Intrusion Detection Systems," arXiv preprint arXiv:2004.11898, 2020.
[22] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in 2015 military communications and information systems conference (MilCIS), 2015.
[23] M. Rigaki, "Adversarial deep learning against intrusion detection classifiers," 2017. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1116037/FULLTEXT01.pdf. [Accessed 1 July 2021].
[24] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein generative adversarial networks," in International conference on machine learning, 2017.
[25] Z. Lin, Y. Shi and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," arXiv preprint arXiv:1809.02077, 2018.
[26] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.
[27] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik and A. Swami, "The limitations of deep learning in adversarial settings," in 2016 IEEE European symposium on security and privacy (EuroS&P), 2016.
[28] "NSL-KDD dataset," 2009. [Online]. Available: www.unb.ca/cic/datasets/nsl.html. [Accessed 1 July 2021].
[29] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in 2017 ieee symposium on security and privacy (sp), 2017.
[30] H. Hosseini, Y. Chen, S. Kannan, B. Zhang and R. Poovendran, "Blocking transferability of adversarial examples in black-box learning systems," arXiv preprint arXiv:1703.04318, 2017.
[31] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, p. 446–452, 2015.
[32] C. Yin, Y. Zhu, J. Fei and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," Ieee Access, vol. 5, p. 21954–21961, 2017.

Authors' Profiles

Amir F. Mukeri received the Diploma in Computer Engineering from AISSMS Polytechnic, Pune, Maharashtra, India in 2002 and B.E. degree in Information Technology from P.V.G.'s College of Engineering and Technology, Pune, Maharashtra, India in 2005. He has more than 15 years of experience working in the software products & SaaS industry in the domain of cloud computing, data storage and protection, virtualization and IOT in India and USA. He is currently pursuing M.E. (Computer Engineering) from AISSMS College of Engineering, Pune, Maharashtra, India. He is a member of IEEE & ACM.

Dr. Dwarkoba. P. Gaikwad received his B.E. degree in Computer Science and Engineering from Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, Maharashtra, India and M.S. degrees in Electrical Engineering from College of Engineering, Pune, Maharashtra, India in 1996 and 2006 respectively. He has been awarded Ph.D. degree in Computer Science and Engineering in 2017. Currently, he is working as an Associate Professor and Head of Department of Computer Engineering in AISSMS College of Engineering, Pune, Maharashtra, India. He has published more than 40 papers in International journal and conferences. He received best researcher award in International Scientist Award Conference held at Vishakhapatnam, India.

How to cite this paper: Amir F. Mukeri, Dwarkoba P. Gaikwad, " Adversarial Machine Learning Attacks and Defenses in Network Intrusion Detection Systems", International Journal of Wireless and Microwave Technologies(IJWMT), Vol.12, No.1, pp. 12-21, 2022.DOI: 10.5815/ijwmt.2022.01.02

