Evaluating Deep Learning-based NIDS in Adversarial Settings

Hesamodin Mohammadian, Arash Habibi Lashkari and Ali A. Ghorbani
Canadian Institute for Cybersecurity, University of New Brunswick, Fredericton, New Brunswick, Canada

Keywords: Network Intrusion Detection, Deep Learning, Adversarial Attack.

Abstract: The intrusion detection systems are a critical component of any cybersecurity infrastructure. With the increase in speed and density of network traffic, the intrusion detection systems are incapable of efficiently detecting these attacks. During recent years, deep neural networks have demonstrated their performance and efficiency in several machine learning tasks, including intrusion detection. Nevertheless, recently, it has been found that deep neural networks are vulnerable to adversarial examples in the image domain. In this paper, we evaluate the adversarial example generation in malicious network activity classification. We use CIC-IDS2017 and CIC-DDoS2019 datasets with 76 different network features and try to find the most suitable features for generating adversarial examples in this domain. We group these features into different categories based on their nature. The result of the experiments shows that since these features are dependent and related to each other, it is impossible to make a general decision that can be supported for all different types of network attacks. After the group of *All* features with 38.22% success in CIC-IDS2017 and 39.76% in CIC-DDoS2019 with ε value of 0.01, the combination of *Forward*, *Backward* and *Flow-based* feature groups with 23.28% success in CIC-IDS2017 and 36.65% in CIC-DDoS2019 with ε value of 0.01 and the combination of *Forward* and *Backward* feature groups have the highest potential for adversarial attacks.

1 INTRODUCTION

Machine Learning has been extensively used in automated tasks and decision-making problems. There has been tremendous growth and dependence in using ML applications in national critical infrastructures and critical areas such as medicine and healthcare, computer security, autonomous driving vehicles, and homeland security (Duddu, 2018). In recent years, the use of Deep learning showed a lot of promising result in machine learning tasks. But recent studies show that machine learning specifically, deep learning models are highly vulnerable to adversarial example either at training or at test time (Biggio and Roli, 2018).

The first works in this domain go back to 2004 when Dalvi et al. (Dalvi et al., 2004) studied this problem in spam filtering. They said linear classifier could be easily fooled by small careful changes in the content of spam emails, without changing the readability of the spam message drastically. In 2014, Szegedy et al. (Szegedy et al., 2013) showed that deep neural networks are highly vulnerable to adversarial examples too.

In recent years deep learning showed its potential in the security area such as malware detection and intrusion detection systems (NIDS). A NIDS purpose is to distinguish between benign and malicious behaviors inside a network (Buczak and Guven, 2015). Historically there are two methods for NIDSs: signature or rule-based approaches. Compared to the traditional intrusion detection systems, anomaly detection methods based on deep learning techniques provide more flexible and efficient approaches in networks with high volume data, which makes it attractive for researchers (Tsai et al., 2009; Gao et al., 2014; Ashfaq et al., 2017).

In this paper, we evaluate the adversarial example generation in malicious network activity classification. We use CIC-IDS2017 and CIC-DDoS2019 datasets with 76 various network features and try to find the most suitable features for generating adversarial examples in this domain. We group these features into different categories based on their nature and generate adversarial examples using features in one or more categories. The result of our experiments shows that since these features are dependent and related to each other, it is impossible to make a general decision that can be supported for all different types of network attacks. We achieved the best result when we use entire features for adversarial example generation. However, in this research, we find some subsets

Mohammadian, H., Lashkari, A. and Ghorbani, A.
Evaluating Deep Learning-based NIDS in Adversarial Settings.
DOI: 10.5220/0010867900003120
In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, pages 435-444
ISBN: 978-989-758-553-1; ISSN: 2184-4356

of features that can achieve an acceptable result.

The rest of this paper is organized as follows: in section two, we review the related works. Section three discusses the background of the work. Section four describes the proposed method. Section five presents the experimental results followed by section six with analysis and discussion. Section seven concludes the paper.

2 BACKGROUND

The primary purpose of adversarial machine learning is to create inputs that can fool different machine learning techniques and force them to make wrong decisions. These crafted inputs are called adversarial examples. These examples are carefully crafted by adding small, often imperceptible perturbations to legitimate inputs to fool deep learning models to make wrong decisions. At the same time, a human observer can correctly classify these examples (Goodfellow et al., 2014b; Papernot et al., 2017).

As mentioned previously, Szegedy et al. (Szegedy et al., 2013) were the first to demonstrated that there are small perturbations that can be added to an image and force a deep learning classifier into misclassification. Let f be the DNN classifier and loss f be its associated loss function. For an image x and the target label l, in order to find the minimal perturbation r they proposed the following optimization problem:

$$min\ \|r\|_2\ s.t.\ f(x+r) = l;\ x+r \in [0,1] \quad (1)$$

By solving this problem, they found the perturbation needed to add to the original image to create an adversarial example.

To make it easier to craft an adversarial example, Goodfellow et al. proposed a fast and simple method for generating adversarial examples. They called their method Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014b). They used the sign direction of the model gradient to calculate the perturbation they wanted to add to the original example. They used the following equation:

$$\eta = \varepsilon sign(\bigtriangledown_x J(\theta, x, l)) \quad (2)$$

Where η is the perturbation, ε is the magnitude of the perturbation, and l is the target label. This perturbation can be computed easily using backpropagation.

3 RELATED WORKS

Most of the early research on adversarial attacks focus on image domain problems such as image classification or face detection. Still, with the increasing usage of DNN in security problems, the researchers realize that adversarial examples may widely exist in this domain. Grosse (Grosse et al., 2017) and Rieck (Rieck et al., 2011) have studied adversarial examples in malware detection. In (Warzyński and Kołaczek, 2018), the authors did a very simple experiment on the NSL-KDD dataset. They showed that it is possible to generate adversarial examples by using the FGSM attack in intrusion detection systems. Rigaki shows that adversarial examples generated by FGSM and JSMA methods can significantly reduce the accuracy of deep learning models applied in NIDS (Rigaki, 2017).

Wang did a thorough study on the NSL-KDD dataset in adversarial setting (Wang, 2018). He used adversarial attack methods including FGSM, JSMA, Deepfool and C&W. He also analyzed the effect of different features in the dataset in the adversarial example generation process. Peng et al. (Peng et al., 2019) evaluated the adversarial attack in intrusion detection systems with different machine learning model. They trained their four detection systems with DNN, SVM, RF, and LR and studied the robustness of these models in adversarial settings. Ibitoye studied the adversarial attacks against deep learning-based intrusion detection in IoT networks (Ibitoye et al., 2019). In (Hashemi et al., 2019), they showed how to evaluate an anomaly-based NIDS trained on network traffic in the face of adversarial inputs. They explained their attack method, which is based on categorizing network features and evaluated three recently proposed NIDSs.

All the previously mentioned works are in white-box settings. This means that the adversary fully knows the target model and has all the information, including the architecture and hyper-parameters of the model. In contrast, in black-box settings, an adversary has no access to the trained model's internal information and can only interact with the model as a standard user who only knows the model output. Yang et al. (Yang et al., 2018) made a black-box attack on the NSL-KDD dataset. They trained a DNN model on the dataset and used three different attacks based on substitute model, ZOO (Chen et al., 2017), and GAN (Goodfellow et al., 2014a). In (Kuppa et al., 2019) they proposed a novel black-box attack which generates adversarial examples using spherical local subspaces. They evaluated their attack against seven state-of-the-art anomaly detectors.

4 PROPOSED METHOD

In this section, we explain our method for making an adversarial attack against the NIDS. First, we train a DNN model for classifying different types of network attacks in our dataset with good performance compared to other classifiers. Since we are making a white-box attack, we assume that the attacker knows the parameter and architecture of the target DNN model. We use one of the well-known adversarial attack methods in computer vision called FGSM to craft our adversarial examples.

4.1 Training the DNN Target Model

First, we train our DNN model for classifying different network attacks. We train a multi-layer perceptron with two hidden layers, each of them has 256 neurons. We used RelU as our activation function and a Dropout layer with 0.2 probability in both hidden layers.

In this research, we use CIC-DDoS2019 (Sharafaldin et al., 2019), and CIC-IDS2017 (Sharafaldin et al., 2018) datasets to train our DNN model and perform the adversarial attack. Each dataset contains several network attacks. The CIC-DDoS2019 attacks are: DNS, LDAP, MSSQL, NetBios, NTP, SNMP, SSDP, UDP, UDP-Lag, WebDDos, SYN and TFTP. The CIC-IDS2017 includes DDoS, PortScan, Botnet, Infiltration, Web Attack-Brute Force, Web Attack-SQL Injection, Web Attack-XSS, FTP-Patator, SSH-Patator, DoS GoldenEye, DoS Hulk, DoS Slowhttp, Dos Slowloris and Heartbleed attack. They extracted more than 80 network traffic features from their datasets using CICFlowMeter (Lashkari et al., 2017) and labeled each flow as benign or attack name.

We used the data from training day of the CIC-DDoS2019 and the whole CIC-IDS2017 to train our DNN model and craft adversarial examples. During preprocessing, we removed seven features, namely Flow ID, Source IP, Source Port, Destination IP, Destination Port, Protocol and Timestamp, which are not suitable for a DNN model.

4.2 Generating Adversarial Examples

We are going to perform the adversarial attack in a white-box setting and craft adversarial examples using different feature sets while using the FGSM (Goodfellow et al., 2014b) method for generating adversarial examples.

We use 76 different features from CIC-DDoS2019, and CIC-IDS2017 as our model input and group these features into six sets and evaluate the effectiveness of using each of these sets and a combination of them to generate adversarial examples. These six sets are Forward Packet, Backward Packet, Flow-based, Time-based, Packet Header-based and Packet Payload-based features. You can find the details of these feature sets in Table 1.

In the FGSM method, after computing the magnitude of the perturbation using Equation 2, the attacker will add the perturbation to all the input features to generate the adversarial example. But, since we only change a subset of input features to craft adversarial examples, we use the following equation:

$$X' = X + mask\_vector * \eta \quad (3)$$

Where X' is the adversarial example, X is the original example, η is the magnitude of the perturbation (ε) multiplied by the sign of the model gradient, and $mask\_vector$ is a binary vector with the same size as input vector which for the features that we want to change, has the value 1 and for the other features 0.

Algorithm 1: Crafting adversarial examples.

for *each* $(x,y) \in Dataset$ **do**
    if $F(x) = y$ **then**
        $\eta = \varepsilon sign(\nabla_x J(\theta, x))$
        $x' \leftarrow x + mask\_vector * \eta$
        if $F(x') \neq y$ **then**
            return x'
        end
    end
end

Algorithm 1 shows how we generate adversarial examples using a different set of features. For each flow in the dataset, we use the FGSM method to compute the magnitude of the perturbation. Then, we multiply the mask vector of the set that we are using and add the result to the original input. If the classifier cannot make a correct prediction for the generated sample, the algorithm will return it as a new adversarial example.

5 EXPERIMENTS AND ANALYSIS

First, we train our DNN model for classifying network attack in both datasets and demonstrate the performance of the classifier. Then we use our white-boxed adversary to perform an adversarial attack on the trained classifier. Our purpose is to evaluate the effect of different feature sets on the adversarial at-

Table 1: Feature sets.

Name of the feature set	List of features
Forward Packet (24)	total Fwd Packets, total Length of Fwd Packet, Fwd Packet Length Min, Fwd Packet Length Max Fwd Packet Length Mean, Fwd Packet Length Std, Fwd IAT Min, Fwd IAT Max, Fwd IAT Mean Fwd IAT Std, Fwd IAT Total, Fwd PSH flag, Fwd URG flag, Fwd Header Length, FWD Packets/s Avg Fwd Segment Size, Fwd Avg Bytes/Bulk, Fwd AVG Packet/Bulk, Fwd AVG Bulk Rate Subflow Fwd Packets, Subflow Fwd Bytes, Init_Win_bytes_forward, Act_data_pkt_forward min_seg_size_forward
Backward Packet (22)	total Bwd Packets, total Length of Bwd Packet, Bwd Packet Length Min, Bwd Packet Length Max Bwd Packet Length Mean, Bwd Packet Length Std, Bwd IAT Min, Bwd IAT Max, Bwd IAT Mean Bwd IAT Std, Bwd IAT Total, Bwd PSH flag, Bwd URG flag, Bwd Header Length, Bwd Packets/s Avg Bwd Segment Size, Bwd Avg Bytes/Bulk, Bwd AVG Packet/Bulk, Bwd AVG Bulk Rate Subflow Bwd Packets, Subflow Bwd Bytes, Init_Win_bytes_backward
Flow-based (15)	Flow duration, Flow Byte/s, Flow Packets/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max Flow IAT Min, Active Min, Active Mean, Active Max, Active Std, Idle Min, Idle Mean Idle Max, Idle Std
Time-based (27)	Flow duration, Flow Byte/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min, Bwd IAT Min, Bwd IAT Max Bwd IAT Mean, Bwd IAT Std, Bwd IAT Total, FWD Packets/s, BWD Packets/s, Active Min Active Mean, Active Max, Active Std, Idle Min, Idle Mean, Idle Max, Idle Std
Packet Header-based (14)	Fwd PSH flag, Bwd PSH flag, Fwd URG flag, Bwd URG flag, Fwd Header Length, Bwd Header Length FIN Flag Count, SYN Flag Count, RST Flag Count, PSH Flag Count, ACK Flag Count, URG Flag Count CWR Flag Count, ECE Flag Count
Packet Payload-based (16)	total Length of Fwd Packet, total Length of Bwd Packet, Fwd Packet Length Min, Fwd Packet Length Max Fwd Packet Length Mean, Fwd Packet Length Std, Bwd Packet Length Min, Bwd Packet Length Max Bwd Packet Length Mean, Bwd Packet Length Std, Min Packet Length, Max Packet Length Packet Length Mean, Packet Length Std, Packet Length Variance, Average Packet Size

Table 2: Results of the Classifiers.

Machine Learning Techniques	DDOS			IDS		
	F1-score	PC	RC	F1-score	PC	RC
DT	97.09	98.54	96.26	99.84	99.76	99.92
Naive Bayes	50.55	60.20	62.03	28.47	32.43	73.75
LR	69.53	70.68	68.94	36.71	39.76	34.96
RF	95.65	98.55	94.60	96.58	99.79	94.24
DNN (Our)	98.97	99.00	98.96	98.18	98.27	98.22

tack against NIDS and also find the most vulnerable type of network attacks against adversarial attacks.

5.1 The DNN Classifier Performance

We train a DNN model on both datasets and compare its performance with other machine learning techniques. The DNN model is a simple multi-layer perceptron. Table 2 shows the results that demonstrate that our model's performance is comparable with other machine learning models.

5.2 The Adversarial Attack Results

After training the DNN model, we use the proposed method to generate adversarial examples for the two selected datasets. To perform the adversarial attack, we use the FGSM method with two different values for ε. In order to choose the suitable ε values for our detailed experiments, first we perform the attack using 6 different values including: 0.1, 0.01, 0.001, 0.0001, 0.00001, and 0.000001 for ε. Based on the results, 0.001 and 0.01 were chosen as the preferred values for ε. Also, we generate the adversarial examples using different feature sets and present the result for each dataset.

5.2.1 CIC-IDS2017

After training the model on CIC-IDS2017, we start generating adversarial examples. We only use those original samples that the model detected correctly. The number of these samples are 2,777,668. As the model could not detect the *Web Attack-SQL Injection*, we do not use them for adversarial sample generation.

Table 3 contains the result of adversarial sample generation on CIC-IDS2017 dataset with 0.001 and 0.01 as values for ε. The table shows the number of adversarial examples generated using different feature sets. The first column is the result when we use all features in the dataset.

With 0.001 the attack cannot generate any adversarial examples for *Infiltration, Web Attack-XSS* and *Heartbleed*, so we remove them from the results table. As we expected, the best result in both cases is when all the features were used. With $\varepsilon = 0.001$, the attack is able to generate adversarial examples for 9.05% of the original samples and with $\varepsilon = 0.01$ for 38.22% of the actual samples.

In both cases the second-best set of features is the combination of *Forward, Backward* and *Flow-based* features with 8.89% for $\varepsilon = 0.001$ and 23.28% for $\varepsilon = 0.01$. The third and fourth-best feature sets are also the same for both ε values. The combination of *Forward* and *Backward* features is third and the combination of *Forward* and *Flow-based* features is

Table 3: CIC-IDS2017 results for $\varepsilon = 0.01$ and $\varepsilon = 0.001$.

Attack Type	ε	All	FWD (F)	BWD (B)	Flow (FL)	F+B	F+Fl	B+FL	F+B+FL	Time (T)	Packet Header (PH)	Packet Payload (PP)	PH+PP	T+PH+PP	Samples
Benign	0.01	585398	102109	31552	31903	143058	109848	12940	172927	29308	7097	63103	65740	76836	2239270
		26.14	**4.55**	**1.40**	**1.42**	**6.38**	**4.90**	**0.57**	**7.72**	**1.30**	**0.31**	**2.81**	**2.93**	**3.43**	
	0.001	61354	56864	19117	23789	55926	58217	29973	57936	42826	1099	13967	14174	51669	
		2.73	**2.53**	**0.85**	**1.06**	**2.49**	**2.59**	**1.33**	**2.58**	**1.91**	**0.04**	**0.62**	**0.63**	**2.3**	
DDoS	0.01	126278	103981	61040	46574	120764	112552	88763	123044	67991	789	89729	91053	117519	127351
		99.15	**81.64**	**47.93**	**36.57**	**94.82**	**88.37**	**69.69**	**96.61**	**53.38**	**0.61**	**70.45**	**71.49**	**92.27**	
	0.001	46408	12641	1960	2497	39023	21059	15629	45182	12173	61	4847	5115	38635	
		36.44	**9.92**	**1.53**	**1.96**	**30.64**	**16.53**	**12.27**	**35.47**	**9.55**	**0.04**	**3.8**	**4.01**	**30.33**	
PortScan	0.01	151032	90279	135567	63349	151032	144544	141332	151032	130072	2439	72248	73116	147790	151480
		99.70	**59.59**	**89.49**	**41.82**	**99.70**	**95.42**	**93.30**	**99.70**	**85.86**	**1.61**	**47.69**	**48.26**	**97.56**	
	0.001	66222	15377	29964	566	66538	21105	32368	66606	2757	1	2823	3823	21468	
		43.71	**10.15**	**19.78**	**0.37**	**43.92**	**1393**	**21.36**	**43.97**	**1.82**	**0**	**1.86**	**2.52**	**14.17**	
Botnet	0.01	699	696	686	690	699	699	698	699	699	1	685	686	699	699
		100	**99.57**	**98.14**	**98.71**	**100**	**100**	**99.85**	**100**	**100**	**0.14**	**97.99**	**98.14**	**100**	
	0.001	676	12	3	2	671	584	559	675	13	0	2	3	672	
		96.7	**1.71**	**0.42**	**0.28**	**95.99**	**83.54**	**79.97**	**96.56**	**1.85**	**0**	**0.28**	**0.42**	**96.13**	
Infiltration	0.01	5	2	5	0	5	4	5	5	4	0	4	4	5	5
		100	**40**	**100**	**0**	**100**	**80**	**100**	**100**	**80**	**0**	**80**	**80**	**100**	
Web Attack-Brute Force	0.01	107	36	36	107	38	107	107	107	107	9	36	36	107	107
		100	**33.64**	**33.64**	**100**	**35.51**	**100**	**100**	**100**	**100**	**8.41**	**33.64**	**33.64**	**100**	
	0.001	35	9	0	0	35	35	5	35	0	0	0	0	9	
		32.71	**8.41**	**0**	**0**	**32.71**	**32.71**	**4.67**	**32.71**	**0**	**0**	**0**	**0**	**8.41**	
Web Attack-XSS	0.01	16	0	10	3	15	3	14	16	3	0	2	7	16	16
		100	**0**	**62.5**	**18.75**	**93.75**	**18.75**	**87.50**	**100**	**18.75**	**0**	**12.50**	**43.75**	**100**	
FTP-Patator	0.01	7771	7722	7732	3847	7771	7767	7771	7771	4211	14	7771	7771	7771	7771
		100	**99.36**	**99.49**	**49.50**	**100**	**99.94**	**100**	**100**	**54.18**	**0.18**	**100**	**100**	**100**	
	0.001	3810	3809	1396	3810	3810	3810	3810	3810	3810	4	3809	3809	3810	
		49.02	**49.01**	**17.96**	**49.02**	**49.02**	**49.02**	**49.02**	**49.02**	**49.02**	**0.05**	**49.01**	**49.01**	**49.02**	
SSH-Patator	0.01	2936	2830	1939	2472	2936	2935	2936	2936	2935	0	67	219	2936	2936
		100	**96.38**	**66.04**	**84.19**	**100**	**99.96**	**100**	**100**	**99.96**	**0**	**2.28**	**7.45**	**100**	
	0.001	4	0	0	0	0	1	2	3	2	0	0	0	3	
		0.13	**0**	**0**	**0**	**0**	**0.03**	**0.06**	**0.1**	**0.06**	**0**	**0**	**0**	**0.1**	
DoS GoldenEye	0.01	7281	4781	4948	598	6643	5129	5443	6676	2809	54	5365	5421	6336	9955
		73.13	**48.02**	**49.70**	**6.00**	**66.73**	**51.52**	**54.67**	**67.06**	**28.21**	**0.54**	**53.89**	**54.45**	**63.64**	
	0.001	710	104	197	22	446	129	305	573	96	7	161	167	467	
		7.13	**1.04**	**1.97**	**0.22**	**4.48**	**1.29**	**3.06**	**5.75**	**0.96**	**0.07**	**1.61**	**1.67**	**4.69**	
DoS Hulk	0.01	174567	165145	81397	58921	173204	167679	81257	173565	59457	2469	105626	105687	149214	227131
		76.85	**72.70**	**35.83**	**25.94**	**76.25**	**73.82**	**35.77**	**76.41**	**26.17**	**1.08**	**46.50**	**46.53**	**65.69**	
	0.001	71477	71185	18027	23375	71423	71226	27507	71645	27256	291	23730	23755	27311	
		31.46	**31.34**	**7.93**	**10.29**	**31.44**	**31.35**	**12.11**	**31.54**	**12.00**	**0.12**	**10.44**	**10.45**	**12.02**	
DoS Slowhttp	0.01	1356	453	462	99	3712	463	483	3734	304	5	528	529	929	5328
		25.45	**8.50**	**8.67**	**1.85**	**69.66**	**8.68**	**9.06**	**70.08**	**5.70**	**0.09**	**9.90**	**9.92**	**17.43**	
	0.001	69	49	46	7	56	50	47	56	41	1	53	53	59	
		1.29	**0.91**	**0.86**	**0.13**	**1.05**	**0.93**	**0.88**	**1.05**	**0.76**	**0.01**	**0.99**	**0.99**	**1.10**	
Dos Slowloris	0.01	4303	4076	2397	2138	4276	4262	2478	4297	2381	10	3693	4211	4245	5609
		76.71	**72.66**	**42.73**	**38.11**	**76.23**	**75.98**	**44.17**	**76.60**	**42.44**	**0.17**	**65.84**	**75.07**	**75.68**	
	0.001	688	522	147	9	659	530	170	682	181	0	253	267	499	
		12.26	**9.3**	**2.62**	**0.16**	**11.74**	**9.44**	**3.03**	**12.15**	**3.22**	**0**	**4.51**	**4.76**	**8.89**	
Heartbleed	0.01	1	0	0	0	1	0	1	1	1	0	0	0	1	10
		10	**0**	**0**	**0**	**10**	**0**	**10**	**10**	**10**	**0**	**0**	**0**	**10**	
Sum	0.01	1061750	482110	327771	210701	614154	555992	344228	646810	300282	12887	348857	354480	514404	2777668
		38.22	**17.35**	**11.80**	**7.58**	**22.11**	**20.01**	**12.39**	**23.28**	**10.81**	**0.46**	**12.55**	**12.76**	**18.51**	
	0.001	251453	160572	70857	54077	238587	176746	110375	247023	89193	1464	50645	51166	144602	
		9.05	**5.78**	**2.55**	**1.94**	**8.58**	**6.36**	**3.97**	**8.89**	**3.21**	**0.05**	**1.82**	**1.84**	**5.2**	

fourth one.

The worst results for both cases are when we use *Packet header-based* features. The reason could be that the number of features in this set is the lowest and almost all the features in this set are based on the packet flags which may not have much effect on detecting the attack types.

If we only compare the results for the main feature sets, the best results for both ε values are when the *Forward* features were used. This result supports our previous findings that show the best feature sets combination are the ones with the *Forward* features present. There is a difference in the second-best features set between two ε values. For value 0.001 the second-best set is *Time-based* features but for value 0.01 is *Packet Payload-based* set. This result shows that increasing the magnitude of the perturbation increases the effect of *Packet Payload-based* features more than *Time-based* features. The third best feature set is *Backward* features.

In Table 3, we also show the number of generated samples with two values of the ε for each network attack type in the dataset.

Comparing the results for the Benign samples, shows that, in all cases increasing the value of the ε will increase the percentage of generated samples, except for the combination of *Backward*, *Flow-based* features and *Time-based* features.

For DDoS attack we are able to generate adversarial examples for 99.15% of original samples when we use *All* the features with $\varepsilon = 0.01$. Unlike Benign samples, the results for all feature sets got better when the ε value is increased.

The third comparison is for PortScan attack. The highest percentage of generated examples is 99.7% with $\varepsilon = 0.01$ for three different feature sets. This result shows we can completely fool our model without even using all the features during adversarial samples generation.

These results for Botnet attack show even with $\varepsilon = 0.001$ in four cases; we were able to generate adversarial examples for more than 95% of the original samples, which means Botnet attack is vulnerable to adversarial attack.

Infiltration, Web Attack-XSS and Heartbleed rows only contain values for $\varepsilon = 0.01$, because the attack cannot generate any adversarial examples with $\varepsilon = 0.001$.

In 7 cases, we were able to generate adversarial examples for all the original examples with $\varepsilon = 0.01$ for Web Attack-Brute Force. Two interesting results are for *Flow-based* and *Time-based* features. The number of generated samples were 0 with $\varepsilon = 0.001$ for these two sets, but with $\varepsilon = 0.01$ the success rate was 100%.

For FTP-Patator with $\varepsilon = 0.001$ the results for all the feature sets are same (94%) except for *Backward* and *Packet Header-based* features. Also, with $\varepsilon = 0.01$ the success was more than 99% for almost all the feature sets.

When we perform the adversarial attack against SSH-Patator samples with $\varepsilon = 0.01$ the success rate is almost zero for all the feature sets. But after increasing the value of the ε we had perfect results except when packet related features were used.

The next four rows are for different types of DoS attacks. It seems the best result with $\varepsilon = 0.001$ is for DoS Hulk and with $\varepsilon = 0.01$ is for DoS GoldenEye. With both ε DoS Slowhttp has the worst result, with success less than 2% for all sets with $\varepsilon = 0.001$.

The last row is the comparison for all the generated samples using different feature sets. As we mentioned earlier, the best and worst feature sets for adversarial sample generation in this dataset are *All* and *Packet Header-based* features.

5.2.2 CIC-DDoS2019

The number of detected samples for CIC-DDoS2019 is 48197029, and we use them for performing our adversarial attack. Since the model is not able to detect any of the Web-DDoS attack samples, we do not use them for adversarial sample generation. The results of adversarial attack on CIC-DDoS 2019 dataset with values 0.001 and 0.01 for ε are shown in Table 4. In this tables, you can see the number of generated adversarial examples and their respective percentage.

With both values for ε, we were able to generate some adversarial examples for all the attacks and feature sets. Same as before the best result is when we use all the features for performing the attack. The percentage of generated sample with $\varepsilon = 0.001$ is 1.14% and with $\varepsilon = 0.01$ is 39.76%. Also, the worst result is for *Packet Header-based* features for both ε values: 0.003% for $\varepsilon = 0.001$ and 0.01% for $\varepsilon = 0.01$. For both ε values the top 5 feature sets are almost same, except for 3rd and 4th place that are changed between *Forward* plus *Backward* features and *Forward* plus *Flow-based* features.

Again, like before we also compare the results for the main feature sets. For both ε values, the best performance is for *Forward* features. The second and third place are visa-versa for two ε values. With $\varepsilon = 0.001$ the second best is *Time-based* features with 0.04% and the third is *Packet Payload-based* features with 0.02%. *Packet Payload-based* result is 4.62% and *Time-based* result is 4.60% for $\varepsilon = 0.01$.

In Table 4, the first row shows the result for Benign samples. Even with $\varepsilon = 0.01$ and using *All* the features, the percentage of generated adversarial samples is less than 7%, which means making an adversarial attack on Benign samples is a tough task.

The next row is for DNS attack. The percentage of generated adversarial samples with $\varepsilon = 0.001$ for all different sets are less 0.4%. But when the value of the ε is increased, we hada better results with 32.58% for *All* the features, 26.16% for combination of *Forward, Backward* and *Flow-based* features, and 21.87% for *Forward* and *Flow-based* features.

The success of the adversarial attack on LDAP samples with $\varepsilon = 0.001$ is almost zero for all the different feature sets with 0.04% as the highest for *All* features. The interesting finding here is that after increasing the ε value to 0.01 we got better result with *Forward* and *Backward* features combination than *All* features.

The next four attacks are MSSQL, NET, NTP and SNMP. The attack performance for all of them is really low with $\varepsilon = 0.001$. But with $\varepsilon = 0.01$ all of them have results more than 64% and up to 86% when using *All* the features and combination of *Forward, Backward* and *Flow-based* features. The next two best feature sets are *Forward, Backward* combination and *Forward, Flow-based* combination, which means using *Forward* features have a great effect on our attack performance.

Amongst all the different attack types, the best results with $\varepsilon = 0.001$ are for SSDP, and UDP. For SSDP when we use *All* the features, *Forward* features or a set that contains *Forward* features we are able to generate adversarial examples for at least 9% percent of original samples. This finding also apply to UDP, but with less percentage of success.

Next is the result comparison for SYN attack sam-

Table 4: CIC-DDoS2019 results for ε = 0.01 and ε = 0.001.

Attack Type	ε	All	FWD (F)	BWD (B)	Flow (FL)	F+B	F+Fl	B+FL	F+B+FL	Time (T)	Packet Header (PH)	Packet Payload (PP)	PH+PP	T+PH+PP	Samples
Benign	0.01	3564	836	357	259	1490	1213	679	1978	582	46	384	412	1045	55008
		6.47	**1.51**	**0.64**	**0.47**	**2.70**	**2.20**	**1.23**	**3.59**	**1.05**	**0.08**	**0.69**	**0.74**	**1.89**	
	0.001	208	100	29	36	150	131	80	188	61	9	35	37	120	
		0.37	**0.18**	**0.05**	**0.06**	**0.27**	**0.23**	**0.14**	**0.34**	**0.11**	**0.01**	**0.06**	**0.06**	**0.21**	
DNS	0.01	1598815	704487	41047	32002	989166	1073303	135588	1283905	66605	59	86070	110125	244175	4907132
		32.58	**14.35**	**0.83**	**0.65**	**20.15**	**21.87**	**2.76**	**26.16**	**1.35**	**0.001**	**1.75**	**2.24**	**4.97**	
	0.001	17208	570	49	59	4615	10551	178	12015	146	14	106	112	4655	
		0.35	**0.01**	**0.0009**	**0.001**	**0.09**	**0.21**	**0.003**	**0.24**	**0.002**	**0.0002**	**0.002**	**0.002**	**0.09**	
LDAP	0.01	1065335	1066028	24793	2315	1094897	1051191	548354	1053778	10311	161	3081	23993	556359	2051711
		51.92	**51.95**	**1.20**	**0.11**	**53.36**	**51.23**	**2.67**	**51.36**	**0.50**	**0.007**	**0.15**	**1.16**	**27.11**	
	0.001	912	612	19	182	701	689	213	714	228	8	65	89	519	
		0.04	**0.02**	**0.0009**	**0.008**	**0.03**	**0.03**	**0.01**	**0.03**	**0.01**	**0.0003**	**0.003**	**0.004**	**0.02**	
MSSQL	0.01	3788705	3000737	52849	297000	3469790	3513326	1751083	3726728	473755	527	45925	117196	2085899	4360932
		86.87	**68.80**	**1.21**	**6.81**	**79.56**	**80.56**	**40.15**	**85.45**	**10.86**	**0.01**	**1.05**	**2.68**	**47.83**	
	0.001	6581	577	323	317	599	622	360	5243	346	151	327	344	613	
		0.15	**0.01**	**0.007**	**0.007**	**0.01**	**0.01**	**0.008**	**0.12**	**0.007**	**0.003**	**0.007**	**0.007**	**0.01**	
NET	0.01	2961679	1063064	58919	1233	1888731	1854728	131697	2512890	60557	439	7504	27656	366333	3915126
		75.64	**27.15**	**1.50**	**0.03**	**48.24**	**47.37**	**3.36**	**64.18**	**1.54**	**0.01**	**0.19**	**0.70**	**9.35**	
	0.001	1314	892	306	44	794	992	450	987	440	2	308	314	507	
		0.03	**0.02**	**0.007**	**0.001**	**0.02**	**0.02**	**0.01**	**0.02**	**0.01**	**0.00005**	**0.007**	**0.008**	**0.01**	
NTP	0.01	870746	351687	201293	140910	781971	671155	496620	833804	500797	331	110267	120348	736063	1191583
		73.07	**29.51**	**16.89**	**11.82**	**65.62**	**56.32**	**41.67**	**69.97**	**42.02**	**0.02**	**9.25**	**10.09**	**61.77**	
	0.001	12052	1590	279	374	1657	5770	778	8842	776	2	23	289	878	
		1.01	**0.13**	**0.02**	**0.03**	**0.13**	**0.48**	**0.06**	**0.74**	**0.06**	**0.0001**	**0.001**	**0.02**	**0.07**	
SNMP	0.01	3979833	3172172	38138	3811	3513740	3380244	434229	3868078	4575	388	147012	148733	943566	5143895
		77.37	**61.66**	**0.74**	**0.07**	**68.30**	**65.71**	**8.44**	**75.19**	**0.08**	**0.007**	**2.85**	**2.89**	**18.34**	
	0.001	1868	757	202	537	1085	1432	720	1806	866	69	338	382	1241	
		0.03	**0.01**	**0.003**	**0.01**	**0.02**	**0.02**	**0.01**	**0.03**	**0.01**	**0.001**	**0.006**	**0.007**	**0.02**	
SSDP	0.01	1666660	684692	197760	113637	1252951	1029881	751695	1338533	461774	2504	637482	641470	1409026	2529104
		65.89	**27.07**	**7.81**	**4.49**	**49.54**	**40.72**	**29.72**	**52.92**	**18.25**	**0.09**	**25.20**	**25.36**	**55.71**	
	0.001	246873	248284	517	2647	253946	245951	14209	245770	11981	287	7075	7960	16671	
		9.76	**9.81**	**0.02**	**0.1**	**10.04**	**9.72**	**0.56**	**9.71**	**0.47**	**0.01**	**0.27**	**0.31**	**0.65**	
UDP	0.01	2677590	2168051	250950	337025	2324650	2061859	1323504	2548356	634744	1355	1124650	1427406	1893083	2958574
		90.50	**73.28**	**8.48**	**11.39**	**78.57**	**69.69**	**44.73**	**86.13**	**21.45**	**0.04**	**38.01**	**48.26**	**63.98**	
	0.001	264463	24371	4139	3147	82019	191851	8193	260576	6404	1084	3079	4831	9908	
		8.93	**0.82**	**0.13**	**0.1**	**2.77**	**6.48**	**0.27**	**8.80**	**0.21**	**0.03**	**0.10**	**0.16**	**0.33**	
SYN	0.01	113763	277	5055	366	110582	606	84612	113691	572	252	847	860	52126	1379129
		8.24	**0.02**	**0.36**	**0.02**	**8.01**	**0.04**	**6.13**	**8.24**	**0.04**	**0.01**	**0.06**	**0.06**	**3.77**	
	0.001	428	135	239	162	328	271	297	334	268	12	104	122	298	
		0.03	**0.009**	**0.01**	**0.01**	**0.02**	**0.01**	**0.02**	**0.02**	**0.01**	**0.0008**	**0.007**	**0.008**	**0.02**	
TFTP	0.01	328122	1124476	2313	5175	579659	584671	8411	284379	7007	369	59033	31529	71214	19375587
		1.69	**5.80**	**0.01**	**0.02**	**2.99**	**3.01**	**0.04**	**1.46**	**0.03**	**0.001**	**0.30**	**0.16**	**0.36**	
	0.001	1382	492	253	179	633	748	355	733	306	76	260	279	574	
		0.007	**0.002**	**0.001**	**0.0009**	**0.003**	**0.003**	**0.001**	**0.003**	**0.001**	**0.0003**	**0.001**	**0.001**	**0.002**	
UDP-Lag	0.01	112420	52988	34434	42	94423	57464	45798	102476	99	25	4510	10426	51591	329248
		34.14	**16.09**	**10.45**	**0.01**	**28.67**	**17.45**	**13.90**	**31.12**	**0.03**	**0.007**	**1.36**	**3.16**	**15.66**	
	0.001	34	22	1	8	29	24	16	31	9	0	2	6	26	
		0.01	**0.006**	**0.0003**	**0.002**	**0.008**	**0.007**	**0.004**	**0.009**	**0.002**	**0**	**0**	**0.001**	**0.007**	
Sum	0.01	19167232	13389495	907908	933775	16102052	15279641	5712270	17668596	2221351	6456	2226765	2660157	8410480	48197029
		39.76	**27.78**	**1.88**	**1.93**	**33.40**	**31.70**	**11.85**	**36.65**	**4.60**	**0.01**	**4.62**	**5.51**	**17.45**	
	0.001	553323	278402	6356	7692	346556	459032	25849	537239	21831	1714	11722	14765	36010	
		1.14	**0.57**	**0.01**	**0.01**	**0.71**	**0.95**	**0.05**	**1.11**	**0.04**	**0.003**	**0.02**	**0.03**	**0.07**	

ples. For ε = 0.001 all the results are almost zero. The feature sets that have *Backward* features have the best result with ε = 0.01, which means they are effective for performing an adversarial attack on SYN samples.

The performance of the adversarial attack on TFTP samples is really low even with ε = 0.01. The unusual finding here is that the result when the attack only uses *Forward* features is the best, even better than using *All* the features. When we use *Forward* features with *Backward* or *Flow-based* features the performance dropped almost to half. This means changing *Backward* or *Flow-based* features is not good for creating adversarial samples.

One to the last attack is for UDP-Lag. Again, the result with ε = 0.001 is not good and close to zero for all the feature sets. With ε = 0.01, the results get better and get up to 34.14% when we perform an adversarial attack with *All* the features. Also, as it is evident in the sub-figure using feature sets containing *Forward* features have the best results.

The last row shows the whole number of generated adversarial samples using each feature sets. As expected, the best result with both values of ε is when *All* the features were used. The next three best results were when we use feature sets containing *Forward* features. Also, the worst result is when we used *Packet Header-based* features in both cases of ε values.

5.3 Perturbation Magnitude Analysis

In the previous section, we provided a comprehensive description and analysis for all illustrated results. We talked about each attack group's results in the two datasets one by one and compared the effect of different feature sets and ε values on the adversarial examples generation results.

Before we go forward with the detailed experiments, we did some experiments with more values for ε. Table 5 and Table 6 contain the re-

Table 5: CIC-IDS2017 results for different ε values.

Epsilon	All	F	B	FL	F+B	F+Fl	B+FL	F+B+FL	T	PH	PP	PH+PP	T+PH+PP
0.1	1666564	1396501	482203	501943	1638785	1481315	571554	1641154	530359	194026	759937	924751	672617
	59.99	**50.27**	**17.35**	**18.07**	**58.99**	**53.32**	**20.57**	**59.08**	**19.09**	**6.98**	**27.35**	**33.29**	**24.21**
0.015	1482685	650795	326461	217063	853691	710699	377653	919083	322101	19500	412036	467719	545263
	53.37	**23.42**	**11.75**	**7.81**	**30.73**	**25.58**	**13.59**	**33.08**	**11.59**	**0.70**	**14.83**	**16.83**	**19.63**
0.01	1061750	482110	327771	210701	614154	555992	344228	646810	300282	12887	348857	354480	514404
	38.22	**17.35**	**11.80**	**7.58**	**22.11**	**20.01**	**12.39**	**23.28**	**10.81**	**0.46**	**12.55**	**12.76**	**18.51**
0.0015	266418	196423	92176	62471	260403	218363	170243	255986	112350	2363	77408	80244	189854
	9.59	**7.07**	**3.31**	**2.24**	**9.37**	**7.86**	**6.12**	**9.21**	**4.04**	**0.08**	**2.78**	**2.88**	**6.83**
0.001	251453	160572	70857	54077	238587	176746	110375	247023	89193	1464	50645	51166	144602
	9.05	**5.78**	**2.55**	**1.94**	**8.58**	**6.36**	**3.97**	**8.89**	**3.21**	**0.05**	**1.82**	**1.84**	**5.2**
0.0001	37195	15055	5461	5674	21066	23628	10197	34413	15626	81	5332	5367	29851
	1.33	**0.54**	**0.19**	**0.20**	**0.75**	**0.85**	**0.36**	**1.23**	**0.56**	**0.002**	**0.19**	**0.19**	**1.07**
0.00001	2878	1960	932	956	2217	2551	1694	2762	1484	3	1114	1117	2269
	0.10	**0.07**	**0.03**	**0.03**	**0.07**	**0.09**	**0.06**	**0.09**	**0.05**	**0.0001**	**0.04**	**0.04**	**0.08**
0.000001	447	202	34	46	213	418	77	434	204	1	24	24	398
	0.01	**0.007**	**0.001**	**0.001**	**0.007**	**0.01**	**0.002**	**0.01**	**0.007**	**0.00003**	**0.0008**	**0.0008**	**0.01**

Table 6: CIC-DDoS2019 results for different ε values.

Epsilon	All	F	B	FL	F+B	F+Fl	B+FL	F+B+FL	T	PH	PP	PH+PP	T+PH+PP
0.1	28106617	28294444	20000389	13087263	29214995	28371739	22846392	27418180	19289663	7806957	27075487	25257239	25783112
	58.31	**58.70**	**41.49**	**27.15**	**60.61**	**58.86**	**47.40**	**56.88**	**40.02**	**16.19**	**56.17**	**52.40**	**53.49**
0.015	22447454	16889691	2863840	1606726	18398603	18475018	9876491	20710804	4984834	13397	4406963	5499886	14502162
	46.57	**35.04**	**5.94**	**3.33**	**38.17**	**38.33**	**20.49**	**42.97**	**10.34**	**0.02**	**9.14**	**11.41**	**30.08**
0.01	19167232	13389495	907908	933775	16102052	15279641	5712270	17668596	2221351	6456	2226765	2660157	8410480
	39.76	**27.78**	**1.88**	**1.93**	**33.40**	**31.70**	**11.85**	**36.65**	**4.60**	**0.01**	**4.62**	**5.51**	**17.45**
0.0015	1429969	576454	13931	18927	723719	641808	31189	916182	27456	2261	19582	23048	57424
	2.96	**1.19**	**0.02**	**0.03**	**1.50**	**1.33**	**0.06**	**1.90**	**0.05**	**0.004**	**0.040**	**0.047**	**1.19**
0.001	553323	278402	6356	7692	346556	459032	25849	537239	21831	1714	11722	14765	36010
	1.14	**0.57**	**0.01**	**0.01**	**0.71**	**0.95**	**0.05**	**1.11**	**0.04**	**0.003**	**0.02**	**0.03**	**0.07**
0.0001	4280	1340	816	290	1829	1657	1317	2364	1221	152	712	809	2029
	0.008	**0.002**	**0.001**	**0.0006**	**0.0037**	**0.0034**	**0.002**	**0.004**	**0.002**	**0.0003**	**0.001**	**0.001**	**0.004**
0.00001	205	173	135	136	194	194	149	202	147	0	144	147	163
	0.00042	**0.0003**	**0.0002**	**0.0002**	**0.00040**	**0.00040**	**0.0003**	**0.00041**	**0.0003**	**0**	**0.0002**	**0.0003**	**0.0003**
0.000001	133	9	0	1	9	9	1	133	1	0	1	1	9
	0.0002	**0.00001**	**0**	**0.000002**	**0.00001**	**0.00001**	**0.000002**	**0.0002**	**0.000002**	**0**	**0.000002**	**0.000002**	**0.00001**

sults of these experiments for CIC-IDS2017 and CIC-DDoS2019 datasets. We started the experiments with $\varepsilon = 0.000001$ and multiplied it by 10 each time for the next ε value until 0.1.

By increasing the value of ε by a factor of 10 each time, it is evident in both tables that the number of generated examples increase for all the different feature groups. But there is no relation between how much we increase the values of the ε and how much more adversarial samples we can generate. Also, the increase between different feature groups is not equal for the same ε value. For example, after increasing the value of ε from 0.01 to 0.1 for CIC-IDS2017 dataset, the percentage of generated adversarial samples with *Forward* features went up from 17.35% to 50.27% which is almost multiplied by 2.9 but samples with *Backward* features increased from 11.80% to 17.35% which is an increase by a factor of 1.5.

After choosing the two final ε values, we did another experiment. We add a small amount to these ε values to evaluate the effect of these small changes. This time we use 0.0015 and 0.015 as the ε values. Results for these two values are also in Tables 5 and 6. As you can see in all cases, the number of generated adversarial examples increased, sometimes by a factor of more than 2.

When we compare our findings for both datasets, we are not able to make a general conclusion on the most influential feature sets for an adversarial attack. For example, we expect to have the best results when using *All* the features, but for DoS Slowhttp in CICIDS-2017 and TFTP in CIC-DDoS2019 we do not get the best result with *All* the features.

The next key finding is that the rankings of feature sets for two datasets are almost the same. The first six best feature sets are the same for both datasets with a slight difference in their ranking for different ε values. Also, the worst feature set for both datasets with both ε values is *Packet Header-based* features. This means it would be better to focus on these feature sets for evaluating and enhancing adversarial attacks performance in network intrusion detection and network traffic classification domain.

In average, it seems that the CIC-DDoS2019 dataset is more robust to adversarial attacks than CIC-IDS2017. With $\varepsilon = 0.001$, the average percentage of generated adversarial samples are 0.36% and 4.55%, which is low for CIC-DDoS2019. For $\varepsilon = 0.01$, they both have averaged around 16%, but since we are trying to make changes as small as possible during our attack, these results show that CIC-DDoS2019 is more robust.

6 CONCLUSION AND FUTURE WORKS

In this paper, we investigate the problem of adversarial attack on deep learning models in the network domain. We chose two famous and well-known datasets: CIC-DDoS2019 (Sharafaldin et al., 2019) and CIC-IDS2017 (Sharafaldin et al., 2018) for our experiments. Since CIC-DDoS2019 has more than 49 millions records and it is more than 16 times the records in CIC-IDS2017, using these two datasets we can verify the scalability of our method. We use CICFlowMeter (Lashkari et al., 2017) to extract more than 80 features from these datasets. From these extracted features, 76 features are used to train our deep learning model. We group these selected features into six different categories based on their nature: *Forward, Backward, Flow-based, Time-based, Packet Header-based* and *Packet Payload-based* features. We use each of these categories and a combination of them to generate adversarial examples for our two datasets. Two different values are used as the magnitude of adversarial attack perturbations: 0.001 and 0.01.

The reported results show that it is tough to make a general decision for choosing the best groups of features for all different types of network attacks. Also, by comparing the results for two datasets, we found out that the adversarial sample generation is harder for CIC-DDoS2019 than CIC-IDS2017.

While the topic of adversarial attack on deep learning model in network domain has been gaining a lot of attention, there is still a big problem comparing these kinds of attack in the image domain. The main point in adversarial attack is to make sure that the attacker did not change the nature of the original sample completely. This is easily done in the image domain by using a human observer. But in the network domain, we cannot use a human expert, and it is tough to make sure the changes we made to the features of a flow did not change the nature of that flow. For future works, the researcher should work on this problem in the network domain.

REFERENCES

Ashfaq, R. A. R., Wang, X.-Z., Huang, J. Z., Abbas, H., and He, Y.-L. (2017). Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences*, 378:484–497.

Biggio, B. and Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331.

Buczak, A. L. and Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017). Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26.

Dalvi, N., Domingos, P., Sanghai, S., and Verma, D. (2004). Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108.

Duddu, V. (2018). A survey of adversarial machine learning in cyber warfare. *Defence Science Journal*, 68(4).

Gao, N., Gao, L., Gao, Q., and Wang, H. (2014). An intrusion detection model based on deep belief networks. In *2014 Second International Conference on Advanced Cloud and Big Data*, pages 247–252. IEEE.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Grosse, K., Papernot, N., Manoharan, P., Backes, M., and McDaniel, P. (2017). Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, pages 62–79. Springer.

Hashemi, M. J., Cusack, G., and Keller, E. (2019). Towards evaluation of nidss in adversarial setting. In *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, pages 14–21.

Ibitoye, O., Shafiq, O., and Matrawy, A. (2019). Analyzing adversarial attacks against deep learning for intrusion detection in iot networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Kuppa, A., Grzonkowski, S., Asghar, M. R., and Le-Khac, N.-A. (2019). Black box attacks on deep anomaly detectors. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–10.

Lashkari, A. H., Draper-Gil, G., Mamun, M. S. I., and Ghorbani, A. A. (2017). Characterization of tor traffic using time based features. In *ICISSp*, pages 253–262.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519.

Peng, Y., Su, J., Shi, X., and Zhao, B. (2019). Evaluating deep learning based network intrusion detection system in adversarial environment. In *2019 IEEE 9th International Conference on Electronics Information* *and Emergency Communication (ICEIEC)*, pages 61–66. IEEE.

Rieck, K., Trinius, P., Willems, C., and Holz, T. (2011). Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4):639–668.

Rigaki, M. (2017). Adversarial deep learning against intrusion detection classifiers.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116.

Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. (2019). Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)*, pages 1–8. IEEE.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., and Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *expert systems with applications*, 36(10):11994–12000.

Wang, Z. (2018). Deep learning-based intrusion detection with adversaries. *IEEE Access*, 6:38367–38384.

Warzyński, A. and Kołaczek, G. (2018). Intrusion detection systems vulnerability on adversarial examples. In *2018 Innovations in Intelligent Systems and Applications (INISTA)*, pages 1–4. IEEE.

Yang, K., Liu, J., Zhang, C., and Fang, Y. (2018). Adversarial examples against the deep learning based network intrusion detection systems. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 559–564. IEEE.