Adversarial Examples Against the Deep Learning Based Network Intrusion Detection Systems

Kaichen Yang*, Jianqing Liu*, Chi Zhang†, Yuguang Fang*
*Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, 32611, USA
Email: ykcdxt@ufl.edu, jianqingliu@ufl.edu, fang@ece.ufl.edu
†Key Laboratory of Electromagnetic Space Information, University of Science and Technology of China
Email: chizhang@ustc.edu.cn

***Abstract*—Deep learning begins to be widely applied in security applications, but the vulnerability of deep learning in front of adversarial examples raises people's concern. In this paper, we study the practicality of adversarial example in the domain of network intrusion detection systems (NIDS). Specifically, we investigate how adversarial examples affect the performance of deep neural network (DNN) trained to detect abnormal behaviors in the black-box model. We demonstrate that adversary can generate effective adversarial examples against DNN classifier trained for NIDS even when the internal information of the target model is isolated from the adversary. In our experiment we first train a DNN model for NIDS system using NSL-KDD database and achieve a performance matching the state-of-art literature, then we show how can an adversary generate adversary examples to mislead the model without knowing the internal information.**

***Index Terms*—deep learning, intrusion detection, security**

I. Introduction

Deep learning demonstrates tremendous potential in the security area, especially in the field of network intrusion detection system (NIDS). NIDS is designed to detect malicious behaviors inside an organization's network [1]. A functional NIDS analyzes and raises alarms for the network traffic entering or exiting the network. Traditional methods of NIDS mainly depend on signature and rule-based approaches. Limited by existing known attack patterns, these approaches may fail when encountering evolving and changing attack scenarios and strategies. Meanwhile, in recent years the amount of network traffic is growing rapidly, and current NIDS based on signatures and rules is insufficient to deal with high volume data. Comparing 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 [2] [3] [4].

Though deep learning is recognized as a promising way in NIDS, recent works have revealed that deep learning models are vulnerable to deliberately crafted inputs known as adversarial examples [5]. Adversarial examples are generated by adding small but intentionally selected perturbations to the original inputs that lead the deep learning models to misclassify. The phenomenon of adversarial examples is further studied by researchers such as Goodfellow [6] and Papernot [7], but the root of adversarial examples is still a problem that is not fully understood. Fast gradient sign method (FGSM) [6], jacobian-based saliency map attack (JSMA) [7], DeepFool [8] and C&W attack [9] are representative adversarial example generation methods.

Current studies on adversarial examples mainly focus on image classification task, but recently researchers begin to realize that adversarial examples may widely exist in various application scenarios including security applications. Grosse [10] and Rieck [11] have studied adversarial examples in malware detection. Rigaki shows that adversarial examples generated by FGSM and JSMA methods can significantly reduce the accuracy of deep learning models applied in NIDS [12]. Wang extends her work to more adversarial attack methods such as Deepfool and C&W attack, he also conducts additional analysis about the relation between adversarial attacks and certain features of traffic records [13]. All these works are based on white-box model because they assume that adversary fully knows the internal information of the target model applied in security applications. However, in practical scenarios black-box model which assumes adversary has no access to the internal information of the trained neural network model is more common since the adversary as a standard user always only knows the output of the model (label or confidence score).

In this paper we try to mimic the adversarial attacks against deep neural network (DNN) model applied for NIDS in real world: adversary examples will be generated and evaluated in black-box model. Specifically, we evaluate three different algorithms in launching adversarial attacks in black-box model and test their performance of bypassing the detection in the experiment. The perturbations to the original attack traffic is constrained so that the attack function will be maintained.

The remainder of this paper is organized as follows: Section 2 reviews the background of our work. Section 3 describes the methodology. Section 4 presents the experimental results and analysis. Section 5 concludes the paper.

This work was partiall supported by the US National Science Foundation under grant IIS-1722791. The work of C. Zhang was supported by the National Key Research and Development Program of China under Grant 2017YFB0802202, and by the Natural Science Foundation of China under Grants 61702474 and 61871362.

978-1-5386-7185-6/18/$31.00 ©2018 IEEE

II. Background

A. Network Intrusion Detection System (NIDS)

NIDS is a mechanism that detects malicious behaviors in a network [14]. Typical malicious behaviors in network includes denial of service (DoS), user to root (U2R), remote to local (R2L), Probing, etc. Normally NIDS are placed at certain entry points within the network to monitor traffic flows, and it would raise alarm once an abnormal behavior is detected. NIDS can be divided into two categories according to the detection approaches, namely: signature-based NIDS and anomaly-based NIDS. Signature-based NIDS works in a similar way comparing to the traditional anti-virus software which extracts patterns as signatures from known malwares. Signature-based NIDS detects malicious traffic by matching specific patterns of the traffic to signatures. However, it is not effective when facing unknown attacks. As the amount of data traffic grows rapidly, the types of network intrusions diversify as well. To detect unknown attacks, anomaly-based NIDS extracts patterns from benign behaviors, and any unknown behaviors deviated from this model would be classified as abnormal. It is appealing because of its ability to detect zero-day attacks. Although anomaly-based NIDS may work against unknown attacks, it suffers from potential high false positives: previously unknown legitimate activities may also be classified as malicious. Deep learning is suitable for anomaly-based NIDS due to its ability to extract and learn a good feature representation from a large amount of data. The deep learning models applied in NIDS are usually deployed in the background of the system so the potential adversary may only be able to obtain the results of the classification and fail to access the internal information.

B. DNN

DNN is composed of several inter-connected layers, each layer consists of a certain number of nodes - named neurons [15]. Each neuron serves as a basic computing unit with an activation function. The input of the activation function on a neuron is the output from the previous layer weighted by parameters, and the output of each layer is simultaneously the input of subsequent layer starting from an initial input. The layers in DNN can be categorized into three classes: input layer, hidden layer and output layer. To hierarchically extract stochastic representations from the large-scale input data for tasks such as clustering and classification, multiple hidden layers of computing nodes may be needed in order to learn the underlying correlation from the input data.

We can consider a DNN model as a function in chain:

$$f(x) = f^{(k)}(\cdots f^{(2)}(f^{(1)}(x))), \tag{1}$$

where $f^{(i)}$ denotes the function of the ith layer of the network, $i = 1, 2, \cdots, k$. The function $f(x)$ can be optimized through the training process in order to minimize the prediction errors. Commonly used optimization functions that adjust $f(x)$ according to the prediction error in DNN is stochastic gradient descent (SGD) and Adam [16] yielding to the well-known backpropagation method. The optimization algorithm repeats a two-phase cycle: propagation and weight update. For each input vector, it is propagated forward through the DNN to reach the output. The loss function is calculated by comparing the output to the expected value. The resulting error value is calculated for each of the neurons in the output layer. The error values are then propagated from the output back through the DNN, until each neuron has an associated error value that reflects its contribution to the original output. Backpropagation uses these error values to calculate the gradients of the loss function. In the second phase, the gradients are fed to the optimization method which in turn uses the gradients to update the parameters in an attempt to minimize the loss function.

C. Adversarial Examples

Though deep learning has achieved remarkable advances in many areas, an intriguing discovery made by Szegedy [5] shows that many deep learning models such as DNN, may not be as smart as they behave. Researchers find that adding slightly but intentionally generated perturbations to original correctly classified examples can cause the model to misclassify. For example, in the field of image classification Szegedy [5] shows that adding very small perturbation to an image could fool a model to misclassify with even higher confidence. The perturbation is small enough to be viewed as insignificant noise that is imperceptible to humans.

The reasons for the existence of adversarial examples is still not thoroughly understood by people. We only know some factors including the highly linear functions in deep learning models, the low flexibility of the classifier and the single unified loss function may cause this phenomenon, .

The first method to generate adversarial examples against deep neural network is introduced by Szegedy [5] in 2014. Their L-BFGS attack tries to find the adversarial example by solving the following problem:

$$\begin{aligned} \min_{x'} \quad & c\,[\eta] + J_\theta(x', l') \\ s.t. \quad & x' \in [0,1] \end{aligned} \tag{2}$$

where η is the peturbation, θ are model parameters, J is the cost function, c is a constant and l' is the target label. The generated adversarial example x' is calculated as: $x' = x + \eta$. L-BFGS is impractical since it applies computation-intensive linear search method to find the optimal value. To generate adversarial examples more efficiently, Goodfellow proposes FGSM [6]. Instead of linear search, only one step gradient update along the direction of the sign of gradient is needed in FGSM. Their perturbations can be expressed as

$$\eta = \varepsilon sign(\nabla_x J_\theta(x, l)) \tag{3}$$

where ε is the magnitude. This operation can be computed fast by backpropagation. One major drawback of FGSM is that to obtain an adversarial example x', the magnitude of disturbances it needs is considerable. In the domain of image classification, it may be acceptable. But in other scenarios it may totally destroy the functionality of the input.

To avoid the above-mentioned problem and reduce the scale of the perturbation, Papernot [7] proposes an iterative way to launch adversarial attack. Their JSMA attack evaluates the model's output sensitivity to each feature in the input domain by jacobian matrix which is given by

$$J_f(x) = \frac{\partial f(x)}{\partial x} = \left[\frac{\partial f_j(x)}{\partial x_i}\right]_{i \times j}. \tag{4}$$

In this way the influence of an individual feature on a particular class could be extracted and ranked through saliency map, and one or more features could be chosen to add perturbations until this change on the original input would result in the misclassification into the target class. JSMA adds smaller perturbations in a smaller portion of features for inducing large output variations to cause misclassifications comparing to FGSM. However, JSMA is much slower than FGSM due to its significant computational cost. Other iterative-based methods such as C&W attack [9] and Deepfool [8] are also proposed.

FGSM, JSMA, C&W attack and Deepfool are all white-box attacks since they require the internal information of the target model. They rely on the transferability to stay effective when the internal information of the target model is not accessible. Transferability is a property of adversarial examples first found by Szegedy, he shows that adversarial examples generated based on a neural network can fool other different neural networks trained for samiliar target. Transferability is critical for black-box attacks where the internal information of the target model is not accessible. Attackers can train a substitute neural network model and then generate adversarial examples against substitute model. Then the target model will be vulnerable to these adversarial examples due to transferability. However, transferability is only widely observed in image classification tasks, its effectiveness in security applications is still untested. Even for image classification tasks, Tramèr [17] claims that transferability might not be an inherent property of deep neural networks by showing a counter-example.

Several approaches are proposed to defend adversarial examples such as [18] and [19], but they will not be considered in this paper since current defense methods cannot ensure effectiveness and efficiency at the same time. Some surveys such as [20] and [21] provide detailed introduction to the existing methods of generating and defending adversarial examples, but adversarial examples in NIDS are not invloved in them.

III. Our proposed Methodology

In this section, we investigate how to craft adversarial examples against deep learning based NIDS in black-box model. First, we train a DNN model to classify malicious behaviors in a network with the performance comparable to the state-of-the-art NIDS classifiers, and then we show how to add small perturbations to the original input to lead the model to misclassify while maintaining the maliciousness of the records. We assume an adversary without the internal information of the DNN model tries to launch black-box attack. Three different black-box attacks will be tried by the adversary: attack based on training substitute model, attack based on zeroth order optimization (ZOO) and attack based on generative adversarial nets (GAN).

A. Evaluation measurement

Consider a binary classifier trained for NIDS, it divides inputs into two categories: normal or abnormal. There are four cases of the relationship between the predicted label and the true label: true positive (TP), false positive (FP), true negative (TN) and false negative (FN).

The performance of this classifier can be measured by the following criteria: accuracy (AC), precision (PC), recall (RC), false alarm (FA) and Fscore.

AC refers to the percentage of correctly labeled records over the total number of records, it is calculated by

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

RC refers to the percentage of correctly labeled malicious records over the total number of attack records, it is calculated by

$$RC = \frac{TP}{TP + FN} \tag{6}$$

PC refers to the number of correct classifications penalized by the number of incorrect classifications, it is claculated by

$$PC = \frac{TP}{TP + FP} \tag{7}$$

FA refers to the percentage of normal records labeled as malicious over the total number of normal records, it is calculated by

$$FA = \frac{FP}{FP + TN} \tag{8}$$

Fscore refers to the harmonic mean of precision and recall, which serves as a derived effectiveness measurement, it is calculated by

$$Fscore = 2 \cdot \frac{PC \cdot RC}{PC + RC} \tag{9}$$

B. Data preprocessing

The dataset we use to train our model and generate adversarial examples is the KDD-series dataset [22]. The first KDD 99 dataset was developed by MIT Lincoln Labs and was widely used by researchers to build training models for NIDS. An upgraded version of KDD dataset - NSL-KDD was created in 2009 by Tavallaee [22], which resolves some problems in the previous KDD 99 dataset such as duplicate records. NSL-KDD dataset contains records representing four categories of attacks:DoS, U2R, R2L and Probing attacks, and the rest records represent normal traffic flow. The records in the NSL-KDD dataset consist of 41 features that depict various attributes of a traffic flow in a network. In this paper, we use KDDTrain+.TXT as the training set, and the KDDTest+.TXT as the test set.

The details of the NSL-KDD dataset are described in Table 1.

TABLE I
RECORDS IN NSL-KDD DATASET

Traffic Label	No. of Training Records	No. of Test Records
DOS	45927	7548
U2R	52	67
R2L	995	2887
Probe	11656	2421
Normal	67343	9711

Records in the NSL-KDD dataset need to be preprocessed before training. Since there are three kinds of data in the NSL-KDD dataset: symbolic, continuous and binary, we need to transform all symbolic and binary variables to numerical using one-hot encoding proposed by Verhoeff [23]. We also apply Min-Max scaler to normalize all features to avoid the case that very large values dominate the training set. After pre-processing the 41-feature NSL-KDD dataset is transformed into a new dataset with 122 numeric features falling into the range between 0 and 1. All attack records are re-labeled as “abnormal” while the rest records are labeled as “normal”.

C. Training the classifier for intrusion detection

In this section, we construct a DNN model as an intrusion classifier. We train a binary DNN model with three hidden layers and 512 hidden units (128-256-128) based on the NSL-KDD training and test sets. In the training process, the inputs x of the model are the vectors representing the 122 features of the records in the NSL-KDD dataset after preprocessing. The output of the model will be the confidence vector $[a, b]$ satisfying $a+b = 1$. The input will be classified as “abnormal” if $a > b$, otherwise the input will be classified as“normal”.

D. Generating adversarial examples in black-box model

In this section, we assume an adversary who tries to generate adversarial examples from abnormal records in black-box model so that they would be misclassified as normal records by our DNN based classifier. For easy comparison, adversarial examples will be generated from the abnormal records from the test set of NSL-KDD. Specifically, the adversary will try three different black-box attacks: attack based on training substitute model, attack based on ZOO and attack based on GAN.

1) Attack based on substitute model: In this section, the adversary will first train a substitute model which serves as the intrusion classifier. The substitute model has a similar structure to the target model, constructed with two hidden layers and each layer contains 128 neural units. The substitute model will also be trained with the NSL-KDD dataset. Then the adversary will launch attacks on the substitute model in the white-box model. Consider the transferability the adversary examples generated on the substitute model will be used to further attack the target model. C&W attack proposed by Carlin [9] will be applied to generate adversary examples on the substitute model due to its high effectiveness. In C&W attack the process of generating adversarial examples are defined as an optimization problem of finding an adversarial perturbation η for an input x as follows:

$$\begin{aligned} &\min_{\eta} \|\eta\|_p + c \cdot g(x') \\ &s.t.\ x' \in [0,1]^n, \end{aligned} \tag{10}$$

where $x' = x + \eta$, $g(x')$ is the objective function satisfying $g(x') \geq 0$ if and only if $f(x') = l'$, f denotes the function of the model, l' denotes the target class, c is a constant, p represents the norm, $[0,1]^n$ represents the format of the input. Normally g is defined as:

$$g(x') = \max\left(\max_{i \neq l'}(Z(x')_i) - Z(x')_t, -k\right), \tag{11}$$

where Z presents the softmax function, k is a constant to control the confidence, t is the index of the target class. ℓ_2 norm is chosen as the distance measurements of perturbations. The optimization problem can be further described as

$$\min_{w} \left\| \frac{1}{2}(\tanh(w)+1) \right\|_2 + c \cdot g\left(\frac{1}{2}\tan g(w) + 1\right) \tag{12}$$

to avoid the box constraint by introducing a new variant w where w satiefies $\eta = \frac{1}{2}(\tanh(w)+1) - x$.

2) Attack based on ZOO: In this section, the adversary will try to launch the attack based on ZOO proposed by Chen [24]. Different from gradient-based adversarial generating approaches which require internal gradient information of the target model, this attack can be launched without gradient information, so it can be applied in black-box model. Inspired by [9], ZOO modified $g(\cdot)$ in Equation (11) as a new hinge-like loss function:

$$g(x') = \max\left(\max_{i \neq l'}(\log[f(x)_i]) - \log[f(x)]_{l'}, -k\right), \tag{13}$$

and uses symmetric difference quotient to estimate the gradient and Hessian:

$$\begin{aligned} \frac{\partial f(x)}{\partial x_i} &\approx \frac{f(x+he_i) - f(x-he_i)}{2h} \\ \frac{\partial^2 f(x)}{\partial x_i^2} &\approx \frac{f(x+he_i) - 2f(x) + f(x-he_i)}{h^2} \end{aligned} \tag{14}$$

where e_i denotes the standard basis vector with the ith component as 1, and h is a small constant.

3) Attack based on GAN: In this section, the adversary will try to generate adversary examples based on GAN. GAN is a type of generative model first introduced by Goodfellow [25]. GAN is designed to generate fake samples that cannot be distinguished from original samples. GAN is composed of two components: discriminator and generator. The generator is a generative neural network used to generate samples. The discriminator is a binary classifier used to determine the generated samples are real or fake. The discriminator and generator are alternately trained so that the generator can generate valid adversarial records. Assuming we have the original sample set x with distribution p_r and input noise variables z with distribution p_z. Model G is a generative multilayer perception function with parameter θ_g that generates fake samples $G(z)$; another model D is a discriminative multilayer perception function with parameter θ_d that outputs $D(x)$ which represents

TABLE II
TEST SET RESULTS OF THE CLASSIFIERS

Technique	AC	PC	RC	FA	Fscore
Naive Bayes	0.760	0.750	0.780	0.260	0.765
Random forest	0.785	0.771	0.810	0.240	0.790
SVM	0.685	0.683	0.690	0.320	0.686
Ours	0.890	0.897	0.900	0.100	0.898

TABLE III
IMPACT OF ATTACK BASED ON SUBSTITUTE MODEL

Technique	AC	PC	RC	FA	Fscore
Substitute model	0.810	0.800	0.815	0.204	0.807
Substitute model under attack	0.533	0.570	0.271	0.204	0.367
Target model	0.890	0.897	0.900	0.100	0.898
Target model under attack	0.738	0.852	0.575	0.10	0.687

TABLE IV
IMPACT OF ATTACK BASED ON ZOO

Technique	AC	PC	RC	FA	Fscore
Target model	0.890	0.897	0.900	0.100	0.898
Target model under attack	0.537	0.634	0.173	0.100	0.273

the probability that model D correctly distinguishes fake samples from the original samples. D and G play the following two-player minimax game with the value function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x\sim p_r}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]. \tag{15}$$

In this competing fashion, GAN is capable of generating raw data samples that look close to the real data.

A more stable GAN called Wasserstein GAN (WGAN) is proposed by Arjovsky [26], which minimizes the objective function with Wasserstein-1 distance:

$$W(p_r, p_z) = \max_{\|f\|_L \leq 1} (E_{x\sim p_r}[D(x)] - E_{x\sim p_z}[D(G(z))]). \tag{16}$$

The classifier of discriminator D satisfies 1-Lipschitz.

In this case, the adversary will use WGAN to generate adversary examples for the target intrusion detection classifiers. The whole process of the algorithm does not require internal information of the target model so it could work in black-box model.

IV. Experiments and Analysis

In this section, we first train a DNN model to classify the records in the NSL-KDD dataset and evaluate its performance through the test set. Then we assume an adversary who uses the three black-box attack methods described above to attack this model. Through the experiments and analysis, we want to verify whether adversarial examples generated in black-box model is effective in the NIDS domain. We also want to determine which black box attack method is most appropriate and effective for NIDS. All experiments are conducted on a lintel i7 laptop with Ubuntu environment and GTX1060 gpu acceleration. We also get help from tensorflow1.7 [27] and cleverhans [28] libraries.

A. The performance of the DNN classifier for intrusion detection

We train a DNN model based on the NSL-KDD dataset and compare its performance with other machine learning techniques. This DNN model is a regular feed forward neural network. The results can be seen in table 2.

B. Attack based on substitute model

In this section the adversary first trains a substitute model with two layers (128-128) from NSL-KDD dataset, then the adversary use C&W algorithm to generate adversary examples target on the substitute model. The scaling factor for the second penalty term is set to 3.0. The Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$ is applied to minimize the loss with learning rate 0.1. The maximum disturbance amplitude under ℓ_2 norm is limited to 0.1 to maintain the functionality. The minimum confidence of adversarial examples is set to 0. The maximum epoch is set to 100. Table 3 shows the adversary impacts of the attack based on substitute model on target DNN classifier.

C. Attack based on ZOO

In this section the adversary will launch the black-box attack based on ZOO algorithm. The Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$ is applied to minimize the loss with learning rate 0.1. The minimum confidence of adversarial examples is set to 0. The maximum epoch is set to 100. The maximum disturbance amplitude under ℓ_2 norm is also limited to 0.1. The minimum confidence of adversarial examples is set to 0. Table 4 shows the adversary impacts of the attack based on ZOO on target DNN classifier.

D. Attack based on GAN

In this section the adversary will launch the black-box attack based on Wasserstein GAN. The generator is a neural-network with 4 linear layers and Relu activation function. The output layer has 122 units to meet the formula of the input vector after preprocessing. The input of the generator is the 122-feature vector representing the attack record combined with a noise vector z. The dimension of the noise vector is 10. The output of the generator is a 122-feature vector representing the adversary example. The discriminator is a multi-layer neural network used to classify the abnormal traffic and the normal traffic, which is trained by the normal records in NSL-KDD dataset and adversary examples generated by generator. The learning rates of the generator and the discriminator are both 0.001. The Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$ is applied to update the generator and discriminator. The weight clipping threshold for the discriminator training is set as 0.01. The maximum disturbance amplitude under ℓ_2 norm is also limited to 0.1. The maximum epoch is set to 100. Table 5 shows the adversary impacts of the attack based on GAN on target DNN classifier.

TABLE V
IMPACT OF ATTACK BASED ON GAN

Technique	AC	PC	RC	FA	Fscore
Target model	0.890	0.897	0.90	0.10	0.898
Target model under attack	0.567	0.700	0.234	0.100	0.350

E. Analysis and discussions

We have the following conclusions beyond the results of the experiment:

- The accuracy, precision, recall and Fscore of the target DNN model are significantly decreased under the black-box attack, which means adversary attacks against DNN model of NIDS without internal information is possible.
- Attack based on substitute model and transferability seems to be less devastating than the other two black-box attacks, it may be due to the fact that the substitute model cannot fully express the function and structure of the target model. The instability of transferability may be another explanation.
- Attack based on ZOO achieves the best results among three black-model attack methods. However, it requires expensive computation to query and estimate the gradients. In actual scenarios NIDS may limit the number of quires the adversary can have. It may reduce the effect of attack based on ZOO.
- Attack based on GAN also achieves good performance, but currently the training of GAN is still unstable and suffered from problems such as convergence failure and model collapse. The application of GAN in the adversary attacks of NIDS may still require further research.

V. Conclusion

In this paper, we study the practicality of crafting adversarial examples against network intrusion detection systems based on deep neural networks. Our results show that despite the fact that the adversary may fail to access the internal information of the target model applied for NIDS when generating adversarial examples, the adversary can still lead the classifier based on DNN to misclassify the attack input as normal input. Our study also reveals the performance of different black-box attack algorithms. From a practical perspective, these results show potential in regards to algorithm selection that could be used by adversary who wishes to adapt her attacks to fool an intrusion detection classifier.

References

[1] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[2] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[3] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *Proceedings of the Advanced Cloud and Big Data (CBD) on 2014 Second International Conference*. IEEE, 2014.

[4] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484–497, 2017.

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

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[7] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the Security and Privacy (EuroS&P) on 2016 IEEE European Symposium*. IEEE, 2016.

[8] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[9] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the Security and Privacy (SP) on 2017 IEEE Symposium*. IEEE, 2017.

[10] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *Proceedings of the European Symposium on Research in Computer Security*. Springer, 2017.

[11] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[12] M. Rigaki, "Adversarial deep learning against intrusion detection classifiers," Master's thesis, Luleå University of Technology, Sweden.

[13] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38 367–38 384, 2018.

[14] T. F. Lunt, "A survey of intrusion detection techniques," *Computers & Security*, vol. 12, no. 4, pp. 405–418, 1993.

[15] R. HECHT-NIELSEN, "Theory of the back propagation neural network," in *Proceeding of the International Joint Conference on Neural Networks (IJCNN)*, 1989.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," *arXiv preprint arXiv:1704.03453*, 2017.

[18] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[19] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Security and Privacy (SP) on 2016 IEEE Symposium*. IEEE, 2016.

[20] X. Yuan, P. He, Q. Zhu, R. R. Bhat, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *arXiv preprint arXiv:1712.07107*, 2017.

[21] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[22] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD cup 99 data set," in *Proceedings of the Computational Intelligence for Security and Defense Applications on 2009 CISDA IEEE Symposium*. IEEE, 2009.

[23] T. Verhoeff, "Delay-insensitive codes—an overview," *Distributed computing*, vol. 3, no. 1, pp. 1–8, 1988.

[24] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "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*. ACM, 2017.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the Advances in neural information processing systems*, 2014.

[26] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the International Conference on Machine Learning*, 2017.

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[28] I. G. R. F. F. A. M. K. H. Y.-L. J. A. K. R. S. A. G. Y.-C. L. Nicolas Papernot, Nicholas Carlini, "cleverhans v2.0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, 2017.