Defending the Defender: Detecting Adversarial Examples for Network Intrusion Detection Systems

Dalila Khettaf
LCSI, Ecole nationale Supérieure d'Informatique (ESI), Oued Smar
Algiers, Algeria
hd_khettaf@esi.dz

Lydia Bouzar-Benlabiod
Jodrey School of Computer Science, Acadia University, Wolfville, NS, Canada
lydia.bouzar-benlabiod@acadiau.ca

***Abstract*—The advancement in network security threats led to the development of new intrusion detection systems(IDS) that rely on deep learning (DL). Along with other systems based on DL, deep IDS suffer from adversarial examples: malicious inputs aiming to change the prediction of a model. Protecting DL against adversarial examples remains an open challenge.**

In this paper, we propose "NIDS-Defend" a framework to enhance the robustness of a network IDS against adversarial attacks. Our framework is composed of two layers: a statistical test and a classifier, together they detect adversarial examples in real-time. The detection process consists of two steps: (1) flagging flows that contain adversarial examples with a statistical test, and (2) extracting individual adversarial examples in the previously flagged flows with a classifier. Our approach is evaluated on a network IDS trained with the NSL-KDD dataset against (1) Boundary attack and (2) HopSkipJumpAttack.

***Index Terms*—Intrusion Detection Systems, adversarial examples, adversarial attacks, adversarial machine learning, statistics**

I. Introduction

Cybercrime is estimated to cost companies worldwide about $10.5 trillion in 2025, up from $3 trillion in 2015[1]. If cybercrime were a country, it would be ranked as the world's third-largest economy right behind the US and China.

To prevent cyberattacks, deep learning (DL) joined the cybersecurity's powerful tools. It has been applied to tackle security-related issues including spam detection, malware detection, intrusion detection, and detection of software vulnerabilities [15]. One particular success of DL is in intrusion detection systems (IDS) which are considered among the main tools for protecting computer systems or networks against cyberattacks.

In the context of IDS, anomaly detection consists of spotting suspicious activities in a network or a host. Different DL models have been used to implement IDSs and achieved satisfactory performances in detecting known and unseen attacks.

The development of deep IDS is accompanied by parallel progress in another field known as "adversarial machine learning", a domain of intersection between ML and cybersecurity. Adversarial ML studies the vulnerabilities of ML and DL models against the so-called "adversarial examples", and the techniques to defend against them. Adversarial examples are data instances that were slightly altered to fool an ML/DL model [14].

The existence of adversarial examples raises security issues in deep IDS. Adversaries can disguise potential attacks as normal data to cause their misclassification.

Since adversarial examples were discovered for the first time [14] and proved to be a security threat against ML/DL systems [9], adversarial ML gained more attention in the research community. However, most works tackle the detection of adversarial examples for the image classification problem and very few focus on IDS.

Deploying a Network IDS (NIDS) is crucial to keep networks safe from cyber-attacks. Despite the outstanding performance of DL-based NIDS in normal settings, they are vulnerable to adversarial attacks that can drastically reduce their accuracy [8].

This paper defines a two-layer framework that detects decision-based adversarial attacks targeting NIDS in real-time settings using statistical properties of adversarial examples [7]. We crafted adversarial examples to test our proposed solution. Other contributions include implementing a NIDS trained with the NSL-KDD dataset and highlighting its vulnerabilities in adversarial settings before demonstrating the efficiency of our approach. The rest of this paper is organized as follows: sectionII explores the two methods: the statistical test and the outlier class introduced in [7]. SectionIII introduces our main contribution (the framework) along with other secondary contributions. SectionIV introduces the experiments and results on the NIDS and the framework. Section**??** discusses further improvements of our approach and compares it to prior work, then sectionV concludes the paper.

II. Statistical Properties of Adversarial Examples

Authors in [7] used hypothesis testing to prove that adversarial examples and clean data are drawn from different probability distributions. The two-sample hypothesis test aims to determine whether two samples are drawn from the same probability distribution. Let X_1 and X_2 be two samples such that $X_1 \sim p$, $X_2 \sim q$ drawn from different distributions p and q where $\|X_1\| = n$ and $\|X_2\| = m$ respectively.

[1]https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

2835-5776/24/$31.00 ©2024 IEEE
DOI 10.1109/IRI62200.2024.00040

The statistical test $\tau(X_1, X_2)$ distinguishes between two hypotheses: the null hypothesis H_0 and the alternative hypothesis H_A. The null hypothesis H_0 states that the two samples are drawn from the same distribution, while the alternative hypothesis H_A states that the two samples are drawn from two different distributions. The statistical test takes the two samples as input and outputs the p-value. The p-value is the probability of obtaining the observed results under the assumption that the null hypothesis H_0 is true. A low p-value indicates that the observed results are very unlikely to happen, and the null hypothesis is rejected.

During the training process, a classifier intends to learn the real distribution of features $D_{real}^{C_i}$ with regard to a partition of data C_i corresponding to a class i. Due to the limited number of data points in the training dataset, a classifier cannot learn the real distribution of data, so it learns the distribution of the training data $D_{train}^{C_i}$.

Generated adversarial examples belonging to a class i constitute their own distribution $D_{adv}^{C_i}$. Since any data point belonging to this distribution is still in the class i, it is also in the real distribution of the data $D_{real}^{C_i}$. But this is not the case for the two distributions $D_{adv}^{C_i} \neq D_{train}^{C_i}$ because adversarial examples crafted from data points in the training set cannot have the same class i as the data they originated from, otherwise they would not be adversarial. This offers the possibility to statistically distinguish between adversarial examples and clean data.

A. The Statistical Test

To compare two data samples, the authors [7] used a statistical test based on a distance metric that is suitable for high dimensional data and small sample sizes. They have chosen to work with the biased estimator of the true Maximum Mean Discrepancy (MMD) introduced by [6] as follows:

$$MMD_b[\mathcal{F}, X_1, X_2] = \sup_{f \in \mathcal{F}} \left(\frac{1}{n}\sum_{i=1}^{n} f(x_{1i}) - \frac{1}{m}\sum_{i=1}^{m} f(x_{2i})\right) \quad (1)$$

x_{1i}, x_{2j} are the ith and jth data points in the samples X_1, X_2 respectively. $f \in \mathcal{F}$ is the kernel function (Gaussian kernel was used by the authors to perform the experiments) that is chosen to maximize the distance between the two samples.

To evaluate the hypothesis that adversarial examples are drawn from a different distribution than clean data, a statistical test is performed. The biased estimate of MMD is computed using the formula in *Equation 1* then 10,000 bootstrapping iterations are performed to estimate the distributions. Finally, the p-value is computed and compared with a threshold (typically 5%). For legitimate data, the p-value is expected to be high. For adversarial examples, however, it is expected to be very low (below the threshold) and thus the hypothesis H_0 is rejected.

B. Detecting Individual Adversarial Examples

The statistical test cannot be used to detect individual adversarial examples. To solve this problem, a second model N is trained with the same dataset as the NIDS. The dataset, however, is augmented with a new category called the outlier class. The model assigns the label "outlier" to all the data points that are not part of the distribution of legitimate data. A model N is trained on a dataset $D = \{X, Y\}$ augmented with adversarial examples crafted from the original dataset. Adversarial examples crafted using different algorithms belong all to the "outlier" category. The model is trained with batches of 1/3 adversarial examples and 2/3 legitimate data.

III. Contribution

In this paper, we propose "NIDS-DEFEND" a two-layer defense framework added on top of the IDS to detect adversarial examples. To introduce our contribution, we start by defining the threat model, then the victim NIDS, and finally we detail the two layers of the framework.

A. The Threat Model

The adversaries cannot get access to the NIDS' architecture or parameters. The NIDS is a black box that adversaries can only query. The adversarial attacks considered here are decision-based and more precisely the two black box attacks: boundary attack [3] and HopSkipJump attack [13].

B. The Victim IDS

The IDS to defend is a deep NIDS based on a sequence-to-sequence architecture.

For the architecture of the NIDS, we have used two layers composed of an encoder and a decoder. The encoder and the decoder are LSTM cells. The two encoders and decoders in the sequence-to-sequence model are connected via their outputs. A dense layer is used to make the final prediction.

C. NIDS-DEFEND: A Framework To Defend NIDS Against Adversarial Examples

The framework is composed of two layers: 1) A statistical test that detects if a network flow contains adversarial examples, 2) A classifier that detects individual adversarial examples.

1) Layer 1: The Two-Sample Statistical Test: The goal of this statistical test is to flag adversarial samples in a network flow. It is performed on the following populations:

- *Population 1:* In a buffer, the NIDS inputs are saved and then arranged into samples of the same size. These samples can contain adversarial examples.
- *Population 2:* This population consists of legitimate data taken from the dataset.

The size of the samples used in the statistical test is chosen after running some experiments.

The followed steps are: 1) Choose a sample of legitimate data from the dataset. 2) Extract a sample of data from the NIDS' inputs. 3) Apply the kernel-two-sample statistical test to compare the two samples, the output of this test is the p-value. 4) Compare the p-value with a threshold set up after experiments. If the p-value is below the threshold, the null hypothesis is rejected, and thus the input sample contains adversarial examples.

2) Layer 2: The Outlier Detection: The performance of this technique depends on the results of the statistical test. A classifier is trained with data belonging to three classes. Two categories, "benign" and "attack" are used, then the dataset is augmented with a third class "adversarial". This class consists of all adversarial examples crafted using different algorithms including white-box and black-box attacks. More details about these algorithms are provided in the section IV.

3) Interaction Between the Two Layers: Figure 1 illustrates the adversarial examples' detection process with our framework: 1) After receiving the network data, samples are saved in buffers, then a statistical test is performed. If the data sample is benign, it will be sent to the NIDS. Otherwise, the data sample is sent to the second layer of the framework. 2) In the second layer, for each data point in the sample, the classifier makes a prediction. If a data point is not benign, then it is rejected. Otherwise, the data point is sent to the NIDS. 3) The final prediction is made by the NIDS to whether a network attack is present in the data. That way, our solution minimizes the number of adversarial examples that get the NIDS.

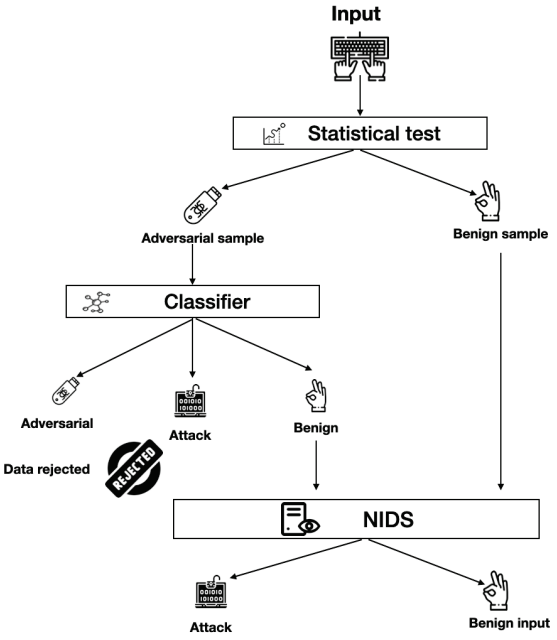

Fig. 1. NIDS-DEFEND architecture

IV. Experimental Setup & Results

In this section, we aim to answer the following questions: 1) Is our intrusion detection system vulnerable against adversarial examples? 2) What is the best population and sample size that provide better results for the statistical test? 3)How good is the detection with an outlier class? 4) How good is the framework in protecting the NIDS? and 5) Can an adversary aware of the defense attack the NIDS?

We carry out a series of tests to prove the effectiveness of our solution. We start by studying the vulnerabilities of the NIDS in the presence of black-box adversarial attacks. Then, we perform tests to choose the parameters of the statistical test and confirm its ability to flag white-box adversarial examples. After that, we test the resistance of the classifier to white-box adversarial examples. We finally perform two tests in a real-world attack scenario and an adversary aware of the defense with black-box adversarial attacks.

A. The Used Dataset

We have chosen to work with the NSL-KDD dataset [11]. It is an improved version of the KDD cup99 dataset where the redundancies were eliminated.

The records in this dataset are made up of 41 attributes and a label that specifies whether the record represents an attack or not. 9 of these attributes are basic, 13 are content-related, 9 are time-related and 10 are host-based. The records in the dataset are available in 5 categories: normal and 4 attack classes: DoS, Probe, R2L, and U2R.

B. Adversarial Examples Crafting

To evaluate each layer of the proposed framework, we started by crafting white-box adversarial attacks. We have used the attack class of the dataset NSL-KDD. The goal is to transform attack records into benign ones while keeping their malicious properties. So the target class for all the attacks is the benign class of the NSL-KDD dataset. For this purpose, the following white-box crafting methods have been used:

- *Fast Gradient Sign Method(FGSM)*:[5] With the maximum perturbation magnitude $\epsilon = 0.1$ and a batch size of 128.
- *Basic Iterative Method/Projected Gradient Descent(BIM/PGD)* [9]: The number of iterations for this attack is 100 and the magnitude of the perturbation is 0.001 each iteration and the maximum perturbation magnitude $\epsilon = 0.1$ and a batch size of 128.
- *Deep Fool*[10]: This attack is generated after 100 iterations with a magnitude of 10^{-6}. We used the l_2 version of the attack.
- *Carlini & Wagner C&W:* [4] This attack was crafted with a learning rate of 0.01 and batch size of 128. The l_2 version of this attack was crafted.
- *Jacobian-Based Saliency Map Attack(JSMA)* [12]: We used the targeted version of the JSMA attack, the target class is the class normal of NSL-KDD. All the features were perturbed by the algorithm, with a perturbation of 0.1 and a batch size of 128.

We first confirm that the statistical test can distinguish between normal data and white-box adversarial examples. After that, the solution is tested on real scenarios of black-box attacks(Boundry and HopSkipJump attacks). For the Boundary attack, we crafted the adversarial examples using the targeted version of the attack with 100 iterations and a learning rate of 0.01. For the HopSkipJump attack, the targeted version was also used. We generated the adversarial examples after 40 iterations and 100 evaluations.

C. NIDS vulnerabilities

This section proves that the NIDS needs to be protected. Table I shows the performance of the used NIDS in normal settings.
The NIDS is vulnerable to adversarial examples. We ran the NIDS in adversarial settings, the success rate for the Boundary attack was 97.81% for the HopSkipJump attack was 100% . The success rate is the rate of the adversarial examples not classified in their target class (the class attack of NSL-KDD). This answers the *first question* and proves how essential our defense is.

TABLE I
NIDS PERFORMANCE IN NORMAL SETTINGS.

Metric	Accuracy	Precision	F1 score	Recall
Value(%)	90.68	99.69	99.74	99.79

D. Statistical Testing

Before performing the statistical test to detect black-box adversarial attacks, we carry out a series of tests to answer the second and fifth questions.

The first test objective is to choose the best population composition and size that can efficiently flag adversarial examples. We run the two-sample test using the following populations:

- *Population 1:* Adversarial examples crafted with the algorithms FGSM, JSMA, C&W, BIM and DeepFool.
- *Population 2:* We first run the statistical test with data belonging to the class "benign" of the NSL-KDD dataset, and then with data belonging to the class "attack" of the same dataset.

We vary the sample size each time to find the optimal value that allows distinguishing between adversarial and normal data.

The figures show the p-value as a function of the sample size for the statistical test between population 1 and the benign class of the NSL-KDD dataset in figure 2 and the attack class in figure 3. The threshold of the p-value used here is 0.005. Based on the graphs, we can conclude that: 1) Comparing adversarial data with data in their target class (benign) allows for more accurate results in the statistical test, .i.e a p-value below the threshold and, 2) the sample size necessary to perform the statistical test is 10 inputs per sample.

For the second test, the objective is to verify whether an adversary who is aware of the defense can still attack the NIDS. In such a scenario, the adversary will inject adversarial examples among legitimate data from both the benign and attack classes of NSL-KDD.

To run the test, the following populations are used:

- *Population 1:* Benign class data from the NSL-KDD dataset.
- *Population 2:* Mixture of benign class data and variable percentages of attack class (normal Mix) and variable percentages of adversarial examples.

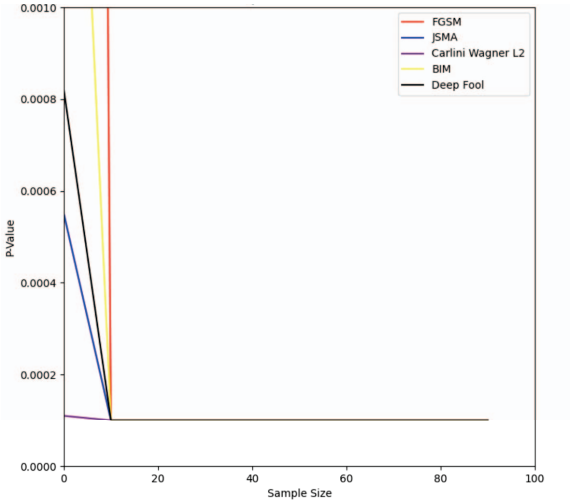

Fig. 2. Statistical test: benign data *vs.* adversarial data.

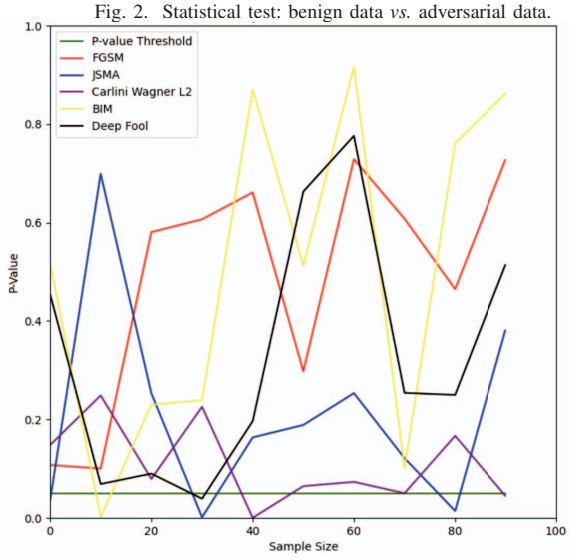

Fig. 3. Statistical test: attack data *vs.* adversarial data.

The figures represent the p-value as a function of the attack class percentage and the adversarial examples' ratio.
For the C&W attack (figure 5), the p-value exceeds the threshold three times for some percentages of the class attack and adversarial examples. For the two attacks, DeepFool (figure 6) and FGSM (figure 4), the p-value exceeds the threshold about 3 times in the whole graphs. For the attack BIM/PGD , when the percentage of the class attack is between 80% and 90% and the adversarial examples' percentage between 10% and 80%, the p-value exceeds the threshold and thus the statistical test fails to detect adversarial samples. For the JSMA attack for all the percentages of the class attack or adversarial examples, the p-value stays below the threshold.
Through these two tests, we answered the *second and fifth question* and confirmed that in most cases, the statistical test is capable of distinguishing between adversarial samples and

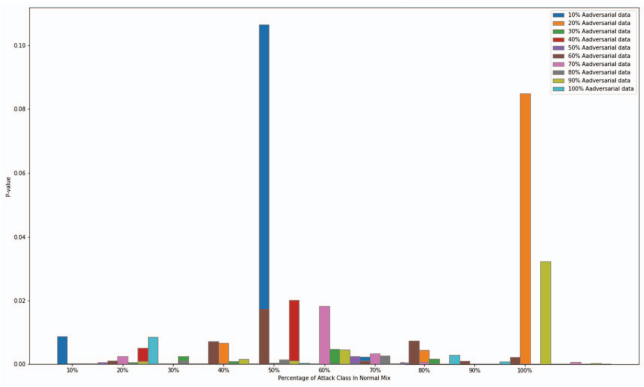

Fig. 4. FGSM attack

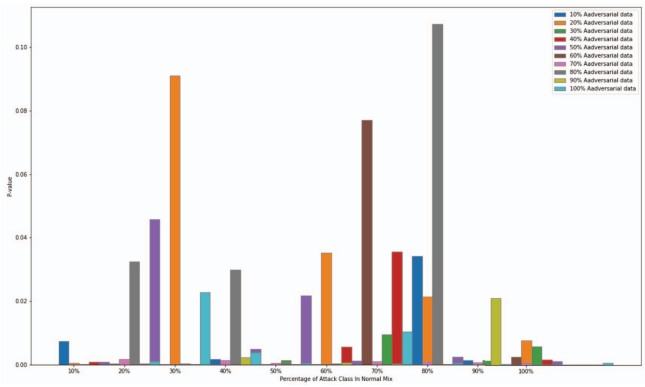

Fig. 5. C&W attack

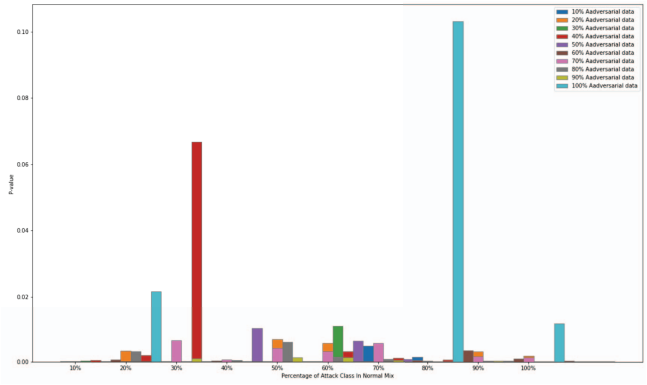

Fig. 6. DeepFool attack

benign data even when the adversary is aware of the defense.

E. The Outlier Detection

To implement this technique, we kept the two classes of the original dataset and added an outlier class with the label "adversarial". The data points in this class are adversarial examples crafted using the training set of NSL-KDD. The algorithms used to generate the adversarial examples are: FGSM, JSMA, Deep Fool, PGD et C&W. We make new training and test sets with equal amounts of the instances for each of the classes: benign, attack and adversarial.

We used a Tabnet [2] classifier, a deep neural network for tabular data. Tabnet could be used with raw data, without the need to preprocess the dataset. The training of our classifier auto-stopped after 22 epochs with an accuracy of 72% for test data.

TABLE II
PERFORMANCE OF TABNET TRAINED WITH AN OUTLIER CLASS

Attack	Detection Rate(%)	Recovery Rate(%)	Error Rate(%)
JSMA	83.51%	8.46%	8.02%
C&W	52.20%	20.16%	27.63%
Deep Fool	73.43%	9.32%	17.23%

Table II shows the results after training, we tested the classifier with the 3 adversarial attacks: JSMA, C&W and DeepFool. Detection rate is the rate of adversarial examples classified in the adversarial class. The recovery rate is the rate of adversarial examples classified on their original class (attack). Error rate is the rate of adversarial examples classified as benign.

The C&W attack has the highest error rate, followed by DeepFool and finally JSMA attack which has the highest detection rate. Overall, the classifier performs well for all the attacks, which answers our *third question* and confirms that the outlier class technique is efficient in detecting and correcting adversarial examples.

F. Real-World Attack Scenario

We run a series of tests using adversarial examples crafted with the algorithms Boundary and HopSkipJump and the NIDS as the victim model.

The first test consists of running a statistical test between the benign data and adversarial examples, the sample size for this test is 10 (already proven in section IV-D).

TABLE III
RESULTS OF THE STATISTICAL TEST.

Attack	Detection Rate(%)	Error Rate(%)
Boundary attack	100%	0%
HopSkipJump attack	95.94%	4.05%

According to table III, the NIDS with the statistical test as a defense has a very small error rate of 4.05% for HopSkipJumpAttack and 0% for the Boundary attack. The detection rate is the rate of the samples that contain adversarial examples and detected as adversarial. The error rate is the rate of adversarial samples detected as benign.

The second test we run consists of using the classifier trained with an outlier class as a defense.

Table IV shows the test on the black-box attacks that record lower error rates compared to white-box attacks. These results show how efficient the outlier classifier is and thus answers the *third question*.

The third test consists of a scenario where an adversary adds adversarial examples within normal data (from both classes:

TABLE IV
PERFORMANCE OF TABNET TRAINED WITH AN OUTLIER CLASS TO DETECT: BOUNDARY AND HOPSKIPJUMP ATTACKS.

Attack	Detection Rate (%)	Recovery Rate(%) Rate(%)	Error
Boundary attack	72.96%	15.07%	11.96%
HopSkipJump attack	81.52%	12.12%	6.34%

attack and benign of the NSL-KDD dataset). To generate this mixture of data, we've randomly taken samples from each class. We followed the steps as in figure 1. Table V shows

TABLE V
NIDS-DEFEND PERFORMANCE IN A REAL-WORLD ATTACK SCENARIO.

Metric	Value (%)
True positive (Statistical test)	64.56%
False positive rate (Statistical test)	19.26%
True negative rate (Statistical test)	12.56%
False negative rate (Statistical test)	3.60%
Detection rate (Outlier class)	57.95%
Recovery rate (Outlier class)	14.39%
Error rate (Outlier class)	23.83%

the results of both layers as a defense. The first layer detects about 64.56% of the adversarial samples. 57.95% of adversarial examples within these samples have been detected as adversarial by the second layer, 14.39% have been recovered and 23.83% misclassified.

This last test shows that our framework is capable of detecting adversarial attacks even when the adversary is aware of the defense and injects benign data among adversarial examples, this answers both *questions four and five*.

As to our knowledge, the work by [16] is the only one that proposes a defense against decision-based attacks. Authors evaluated their solution with a different dataset CSE-CIC-IDS2018[1]. The lowest success rate they obtained for boundary attack was 5.86%, our lowest success rate of the same attack is 0%. Regarding HopSkipJump attack, the lowest success rate they obtained was 5.88% whereas ours is 4.05%. This shows that our framework performs better for decision-based attacks.

V. CONCLUSION

After confirming that the NIDS is not resistant to adversarial examples, we proposed "NIDS-DEFEND", a framework composed of two layers: a statistical test and a multi-classifier trained with the NSL-KDD dataset. We started by evaluating the performance of our NIDS in both normal and adversarial settings. Then, the effectiveness of each layer has been evaluated. As a final step, the framework has been assessed in a scenario where the adversary is aware of the defenses. We crafted the adversarial examples that were used for testing.

Our solution could be further improved by increasing the accuracy of the NIDS and also speeding up the runtime performance of the entire framework. The framework can also be generalized to other decision-based attacks like Pointwise, NES, or OPT.

REFERENCES

[1] Mohammed Hasan Ali et al. "A new intrusion detection system based on fast learning network and particle swarm optimization". In: *IEEE Access* 6 (2018), pp. 20255–20261.

[2] Sercan Ö Arik and Tomas Pfister. "Tabnet: Attentive interpretable tabular learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 8. 2021, pp. 6679–6687.

[3] Wieland Brendel, Jonas Rauber, and Matthias Bethge. "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models". In: *arXiv preprint arXiv:1712.04248* (2017).

[4] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *2017 ieee symposium on sec and privacy*. Ieee. 2017, pp. 39–57.

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[6] Arthur Gretton et al. "A kernel two-sample test". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 723–773.

[7] Kathrin Grosse et al. "On the (statistical) detection of adversarial examples". In: *arXiv preprint arXiv:1702.06280* (2017).

[8] Olakunle Ibitoye, Omair Shafiq, and Ashraf Matrawy. "Analyzing adversarial attacks against DL for intrusion detection in IoT networks". In: (2019), pp. 1–6.

[9] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[10] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal 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, pp. 2574–2582.

[11] *NSL-KDD*. https://www.unb.ca/cic/datasets/nsl.html.

[12] Nicolas Papernot et al. "The limitations of deep learning in adversarial settings". In: *2016 IEEE European symposium on sec and privacy*. IEEE. 2016, pp. 372–387.

[13] Lukas Schott et al. "Towards the first adversarially robust neural network model on MNIST". In: *arXiv preprint arXiv:1805.09190* (2018).

[14] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[15] Tony Thomas, Athira P Vijayaraghavan, and Sabu Emmanuel. "Machine learning and cybersecurity". In: *Machine Learning Approaches in Cyber Security Analytics*. Springer, 2020, pp. 37–47.

[16] Chaoyun Zhang, Xavier Costa-Pérez, and Paul Patras. "Tiki-taka: Attacking and defending deep learning-based intrusion detection systems". In: *2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pp. 27–39.