CTU Journal of Innovation and Sustainable Development
ISSN 2615-9422 | e-ISSN 2815-5602

DOI:10.22144/ctujoisd.2023.030

An interpretable approach for trustworthy intrusion detection systems against evasion samples

Nguyen Ngoc Tai, Hien Do Hoang, Phan The Duy, and Van-Hau Pham*

Information Security Laboratory, University of Information Technology, Ho Chi Minh city, Viet Nam

Viet Nam National University Ho Chi Minh city, Viet Nam

**Corresponding author (haupv@uit.edu.vn)*

Article info.

Received 5 Aug 2023
Revised 10 Sep 2023
Accepted 27 Sep 2023

Keywords

Machine learning, XAI, intrusion detection, adversarial sample

ABSTRACT

In recent years, Deep Neural Networks (DNN) have demonstrated remarkable success in various domains, including Intrusion Detection Systems (IDS). The ability of DNN to learn complex patterns from large datasets has significantly improved IDS performance, leading to more accurate and efficient threat detection. Despite their effectiveness, DNN models exhibit vulnerabilities to adversarial attacks, where malicious inputs are specifically crafted to deceive the models and evade detection. This paper provides insights into the effectiveness of deep learning-based IDS (DL-IDS) against adversarial example (AE) attacks. We tackle the weaknesses of DNN in detecting adversarial attacks by proposing the Convolutional Neural Network (CNN), which serves as an AE detector. We also utilize one of the XAI techniques, specifically SHAP, to enhance the transparency of the AE detector. Our results show that the AE detector has obvious effects for detecting adversarial examples and achieves an impressive 99.46% accuracy in our experimental environment.

1. INTRODUCTION

According to recent studies (Sun et al., 2020; Otoum, 2022), the Deep Learning based Intrusion Detection System (DL-IDS) performs exceptionally well. However, a notable weakness of Deep Learning (DL) models is their vulnerability to adversarial attacks. These attacks manipulate or deceive DL models by adding subtle perturbations to the data input (Liang, 2022). This can cause misclassification and failure of intrusion detection to DL models, compromising security systems. To find solutions to prevent these attacks, several studies (Ko, 2021; Wang N. A., 2022) have been conducted and showed promising outcomes. Yet, there are still critical concerns regarding the lack of transparency in their adversarial example (AE) pattern detection methods.

Explainable Artificial Intelligence (XAI) is a branch of artificial intelligence research that focuses on developing models and methods capable of providing understandable and transparent explanations for the decision-making processes of machine learning models. In the cybersecurity domain, XAI can achieve transparency of models or enhance model performance, as well as explain errors made by classifiers (Capuano, 2022). There are studies (Peng, 2022) (Le, 2022) utilizing XAI to enhance the transparency of DL-IDS. However, there is still no research using XAI to increase the transparency of AE detection methods. Our work contributes primarily to addressing this knowledge gap by:

– Exploring the impact of highlight features on the model's effectiveness against AE attacks.

– Developing an AE detector that can distinguish between benign and adversarial samples.

– Applying an XAI method, specifically SHAP framework (Lundberg, 2017), to explain the AE detector predictions, aiming to enhance the transparency of the detector.

The rest of this paper is structured as follows. Section 2 introduces related studies on using XAI to explain Machine Learning/Deep Learning IDS and adversarial attack detection methods. In Section 3, we present the details of our approach. Section 4 describes experiment settings and results. Finally, we illustrate our discussion, conclusion, and future work in Section 5.

2. RELATED WORK

Ko and Lim (Ko, 2021) proposed a method that used model explanation to detect adversarial examples and did not rely on pre-generated adversarial samples. First, they used the saliency map method to generate input explanations. Second, they used generated explanations to train their reconstructor networks. Finally, they used the reconstructor networks to separate normal and adversarial examples. They conducted experiments on the MNIST handwritten digit dataset and applied attack techniques such as the Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), and Momentum Iterative Method (MIM). Their method achieved the adversarial example detection rates 99.23% on FGSM with an epsilon of 0.1, 95.73% with an epsilon of 0.2, 96.93% with an epsilon of 0.3, and 98.39% on Basic Iterative Method (BIM) with an epsilon of 0.1, 98.94% with an epsilon of 0.2, 99.01% with an epsilon of 0.3.

Wang (2022) leveraged SHapley Additive exPlanations (SHAP) values to discriminate between normal and adversarial samples. First, they created a data set of benign and adversarial examples from popular datasets via a Generating Adversarial Example algorithm. The algorithm utilized well-known attack techniques, such as the Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM), and Carlini and Wagner (C&W), to create adversarial examples. Second, they generated SHAP values for this data set by SHAP DeepExplainer and referred to them as an XAI signature data set. Finally, they used the XAI signature data set to train a fully connected feed-forward neural network model. The model distinguished between normal and adversarial samples. They conducted experiments using each data set and model pair (i.e., [CIFAR-10, ResNet56] and [MNIST, CNN]). CIFAR's AUC ROC and AUC PR pairs were 96.6% and 95.8%, whereas MNIST's were 96.7% and 96.1%.

Peng (2022) proposed a novel trustworthy intrusion detection framework (TIDF) by applying ex-post interpretation and machine learning. The network data underwent preprocessing, balancing, and division before being fed into a model for prediction. They then performed objective and subjective tests to evaluate the performance of the TIDF framework. Global and local interpretation models were employed to increase the clarity of prediction findings based on network access data. The findings and explanations were obtained by the Network Security Management Engineer (NSME). These results showed that the suggested structure improved the transparency of the IDS. They then used the NSL-KDD data set to evaluate the effectiveness of the method, and the results showed that the proposed framework significantly enhanced the IDS's transparency.

Le et al. (2022) employed SHAP to explain and interpret Decision Tree and Random Forest decisions on three public IoT-based data sets, such as IoTID20, NF-BoT-IoT-v2, and NF-ToN-IoT-v2. They used PyCaret to implement an algorithm that selected the best model and its appropriate hyperparameters for binary and multiple classification tasks in each dataset. After running the algorithm, they trained the chosen model using its related dataset and hyperparameters. Finally, they leveraged the SHAP method to explain the models' predictions, employing the Heatmap for global explanations and the Decision Plot for local explanations.

These studies had made significant progress in using XAI frameworks to increase the transparency of ML (machine learning)/DL-IDS and have shown promising results in adversarial detection methods, bolstering the robustness of DL-IDS against Adversarial Attacks. However, the lack of transparency in these detection methods presents a challenge in comprehending the employed defense mechanisms. In response to these challenges, we integrate a XAI support into the AE detector mechanism, allowing for a more interpretable understanding of the defense strategies employed, shedding light on the decision-making process, and further enhancing the overall resilience of ML/DL-IDS against Adversarial Attacks. Table 1 compares our approach with previously proposed state-of-the-art methods.

Table 1. The comparison between the current state-of-the-art methods and our approach

Proposed Method	XAI method	Dataset	XAI for classifier	AE detector	XAI for AE detector
(Ko, 2021)	Saliency Map	MNIST	Yes	Yes	No
(Wang N. A., 2022)	-	NSL-KDD, MNIST	No	Yes	No
(Le, 2022)	SHAP	NSL-KDD	Yes	No	No
(Peng, 2022)	Mean Decrease Impurity; Partial Dependence Plot; SHAP; Local Interpretable Model-Agnostic Explanation; Explain like I'm five	NSL-KDD	Yes	No	No
Ours	**SHAP**	**NSL-KDD**	**Yes**	**Yes**	**Yes**

3. OUR APPROACH

3.1. Explanation method

SHAP introduced in (Lundberg, 2017), is one of the XAI techniques that helps to interpret and understand outcomes made by machine learning models. The primary purpose of SHAP is to calculate the contribution of each feature to the ML model's prediction. The SHAP can provide both global and local explanations. In this paper, we use the Deep SHAP method to rank significant features and explain the predictions of an AE detector.

3.2. Ranking the features

Wilson and Danilo proposed a feature selection mechanism in (Wilson, 2020) using SHAP values. We assume that $x \in \mathbb{R}^{c \times m \times n}$ where x is SHAP values, c is the number of classes, n is the number of instances and m is the number of features. For each class k $(k \in \mathbb{N}, 1 \leq k \leq c)$, its SHAP value of feature j $(j \in \mathbb{N}, 1 \leq j \leq n)$ and instance i $(i \in \mathbb{N}, 1 \leq i \leq m)$ is $x_{k,i,j}$, the average SHAP value of class c and feature j is:

$$s_j^k = \frac{\sum_{i=1}^{m} x_{k,i,j}}{m}$$

A SHAP sum value of feature j is:

$$S_j = \sum_{k=1}^{c} s_j^k$$

The SHAP sum is a set of $\{S_1, S_2, \ldots, S_n\}$. Based on the SHAP sum, we infer a list of most important features in any order.

3.3. Proposed model

Figure 1 illustrates our interpretable approach for trustworthy intrusion detection systems against evasion samples. There were two stages: the primary goal of stage 1 was to create a list of predefined features, while the primary goal of stage 2 was to detect attacks, test evasion samples, and provide result explanations.

In stage 1, network traffic was first collected from the internet and then preprocessed to have an appropriate input format for the trained DNN model to produce an output. The outputs were then sent to DeepExplainer to obtain SHAP values. A list of predefined features was created by sending the SHAP values to a Feature Rank and Selector mechanism. The mechanism comprised of 2 steps, the first one evaluated and arranged important features in order, as detailed in Section 3.2. The second one extracted a number of the most important features with conditions that exclude the features generated from One Hot Encoder and the features that do not significantly influence the model's predictions, for which a SHAP sum of the feature is 0. For features generated by One Hot Encoder, they needed to be retained because, in AE samples created by FGSM with an epsilon of 0.2, for example, they may contain a value of 0.2 instead of just zero or one, which is illogical. Although it is not mandatory to remove features that do not significantly influence the model's predictions, we removed them to focus on the most influential features.

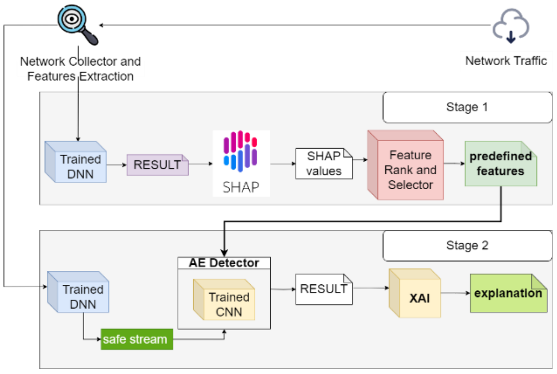

Figure 1. Proposed method

Stage 2 was only be deployed after the list of predefined features had been defined. In stage 2, when a network stream arrives, its information was first extracted and pre-processed. Next, the trained DNN classified the network stream into an attack class, such as distribution denial of service, malware, privilege escalation, or a safe class. If a DNN prediction was a safe class, the network stream was sent to an AE detector. The AE detector extracted information from the network stream based on the list of predefined features and feeds extracted information to a CNN (Convolutional Neural Network) model for a prediction. The prediction result from the AE detector was sent to an XAI to provide an explanation.

4. EXPERIMENTAL RESULT

4.1. Experimental Environment

Our experiments were conducted on a Linux operating system with an 8-core CPU and 16 GB of RAM and Python version 3.10. Adversarial Robustness Toolbox (ART, n.d.) was utilized to generate adversarial samples. Additionally, SHAP DeepExplainer (SHAP, n.d.) is used to calculate SHAP values.

4.2. Dataset

NSL-KDD data set was used in our experiment. A total of 148,517 network access records made up the data set, which was split into 125,973 samples for KDDTrain+ and 22,544 samples for KDDTest+. These samples were categorized into five groups: Benign, DoS (Denial of Service attacks), Probe (Probing attacks), R2L (Root to Local attacks), and U2R (User to Root attacks). The data set comprised of 41 features that allowed us to study various aspects of network traffic. First, duplicate and NA values were eliminated from the train and test sets. Second, we used one hot encoding to convert "protocol_type", "service", and "flag" feature to binary features. Next, the data set was scaled by the Min-Max Scaler:

$$x_{scale} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Where x represents the original value of a feature, x_{min} is the minimum value of the feature and x_{max} is the maximum value of the feature. Finally, SMOTE method is used (Chawla, 2002) to increase the number of 'L2R' and 'U2R' instances in the preprocessed KDDTrain+, which each type being increased to 5000 instances.

Table 2. DNN hyper-parameters

Layer 1	Dense (122, ReLU activation)
Layer 2	Dense (100, ReLU activation)
Layer 3	Dense (100, ReLU activation)
Layer 4	Dense (100, ReLU activation)
Layer 5	Dense (100, ReLU activation)
Output	Dense (5, Softmax activation)
Loss Function	Categorical Cross Entropy
Optimizer	Adam

Table 3. CNN hyper-parameters

Layer 1	Convolution1D (2048, ReLU activation)
Layer 2	MaxPooling1D
Layer 3	Convolution1D (512, ReLU activation)
Layer 4	MaxPooling1D
Layer 5	Flatten
Layer 6	Dense (512, ReLU activation)
Output	Dense (2, Softmax activation)
Loss Function	Categorical Cross Entropy
Optimizer	Adam

4.3. Deep Neural Network

A DNN was constructed to classify inputs into the five groups mentioned in 4.2, at the same time, it is also a target of adversarial attacks. Table 2 displays the hyper-parameters of the DNN. Additionally, the number of epochs and batch sizes were 5 and 1,024, respectively.

4.4. Adversarial Examples Detector

CNN was constructed to discriminate between clean and AE samples. Table 3 displays hyper-parameters of the CNN. Additionally, the number of epochs and batch size were 2 and 16, respectively.

4.5. Experiment Steps

This session presents each individual step to conduct our experiment:

Step 1: Preprocessing the data set as described in Section 4.2 and we obtained the preprocessed KDDTrain+ and the preprocessed KDDTest+. We then trained the DNN using the preprocessed KDDTrain+.

Step 2: 500 instances were taken from the preprocessed KDDTest+ and then sent them to DeepExplainer to retrieve SHAP values. The number of background data for the DeepExplainer was 512 samples from the preprocessed KDDTrain+. A list of predefined features was produced after the operation of the Feature Rank and Selector processes.

Step 3: 10,000 samples were collected from the preprocessed KDDTrain+ and generate 4 adversarial example versions of them. Each version was generated by one of AE attack methods, such as FGSM (Goodfellow, 2014), BIM (Wang J. , 2021), Jacobian-based saliency map attack (JSMA) (Papernot, 2016) and DeepFool (Moosavi-Dezfooli, 2016). After performing the attacks, all versions were necessary to retain features that are not in the list of predefined features. Then, we selected instances in all versions that transform the DNN model's predictions from any attack to normal and refer to them as evasion samples. We merged all 'normal' labeled instances from the preprocessed KDDTrain+ data set and evasion samples into a collection and then referred to it as a Defense Set. The evasion samples in the Defense Set were labeled ‘attack', while the remains were labeled ‘normal'. We then over-sampled the 'attack' labeled samples in the Defense Set by SMOTE (Chawla, 2002) . Here we used SMOTE because, in the context of multi-class classifications, each instance targeted by any AE attack can transform from an attack prediction to another attack prediction, so there are not many instances labeled ‘normal' by DNN-IDS, which leads to the imbalance between safe and evasion samples in Defense Set. The Defense Set was split into a 7:3 ratio for training and testing the AE detector, where 70%, referred to as the Defense Train Set, was used for training and 30%, referred to as the Defense Test Set, for testing. Finally, we extracted information from the Defense Train Set based on the list of predefined features to train the AE detector and test the detector by Defense Test Set as an experiment.

Step 4: We sampled 5,000 examples from the preprocessed KDDTest+, filter instances that DNN-IDS's predictions are attacking, and then generated an adversarial version of them in which each sample was targeted by one of the four attack methods, such as FGSM, BIM, JSMA, and DeepFool, randomly. Here, it is necessary to retain features of the adversarial version that are not in the predefined feature list. Next, we selected samples that make the predictions of DNN-IDS as safe samples from the adversarial version, which we refer to as evasion samples. Sequentially, we merged the benign instances in preprocessed KDDTest+ and evasion samples into a set and refer to them as the Random Test Set. Finally, the AE detector was tested using the Random Test Set as a practice experiment.

4.6. Evaluation Metrics

Figure 2 displays the confusion matrix of DNN model prediction on the KDDTest+. In the Normal, DoS, and Probe classes, the DNN performed well; however, in the U2R and U2R classes, it performed a poorly.

Figure 3 displays accuracy values of DNN-IDS when the AE attack methods increased the number of targeted features. The y-axis represents the model's accuracy, while the x-axis represents the number of the most or least important features. Note that the most or least important features are in the list of predefined features. For example, if the value on the x-axis is 10, it means we consider the top 10 of the most or least important features. The solid line represents the variation in accuracy when increasing the number of the most important features targeted by adversarial attack methods. On the other hand, the dotted line represents a similar variation, but it considers the number of least significant features. Based on this Figure, we may conclude two things: 1) The accuracy of DNN-IDS decreases as more features are targeted by AE attacks. 2) Attacking the most important features typically leads to a more significant decrease in the model’s accuracy compared to attacking the least one, considering the same number. The Deep Fool, on the other hand, displayed the opposite trend by selecting fewer than 15 features. Table 4 illustrates the top 8 influential features.

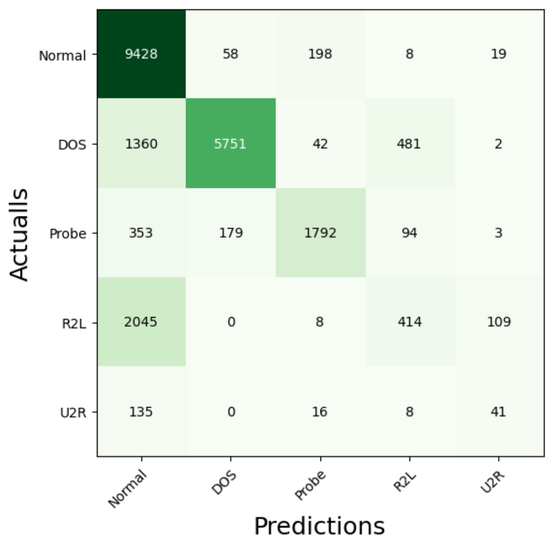

Figure 2. Confusion Matrix on KDDTest+ of DNN model

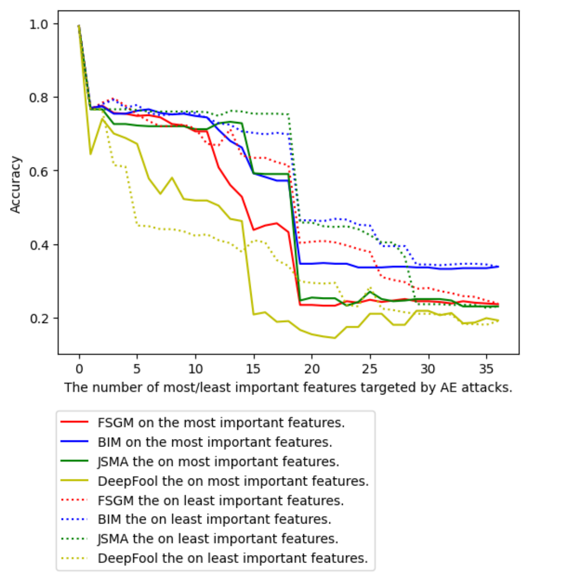

Figure 3. The change in accuracy of the model encountering AE attacks

Table 5 displays the number of safe and evasion samples in the Defense Set (before applying SMOTE (Chawla, 2002)) and the Random Test Set. We can see that the Defense Set shows a serious imbalance between safe and evasion samples, therefore, SMOTE method (Chawla, 2002) is used to balance the attack class in the Defense Set. The imbalance of Random Test Set is lower than the Defense Set, so we decide not to balance the Random Test Set.

Table 4. The top 8 features with the highest SHAP sum

Features	SHAP sum value of the feature
service_auth	242.6854267
dst_host_srv_count	212.4594738
dst_host_serror_rate	191.2416891
srv_count	113.7827973
srv_serror_rate	70.11557833
count	69.99419681
same_srv_rate	63.1915511
logged_in	44.15858485

Table 5. Defense Set and Random Test Set information

	Normal	AE samples	Total
Defense Test Set	67583	9659	77242
Random Test Set	2895	1348	4243

Table 6. Metrics on Defense Test and Random Test Set

	Accuracy	Precision	Recall	F1-Score
Defense Test Set	99.46%	99.76%	99.16%	99.46%
Random Test Set	65.68%	18.60%	2.37%	4.21%

Table 6 represents the evaluation metrics of the AE detector on the Defense Test Set and Random Test Set. Although the AE detector shows high performance on the Defense Test Set, it has poor effectiveness on the Random Test Set. This issue can be DUE TO by several reasons. Firstly, the DNN-IDS's performance in testing is not high, which affects the extraction of the most important features and results in an ineffective list of predefined features. Secondly, selecting only 500 instances from the preprocessed KDDTest+ dataset in Step 2.

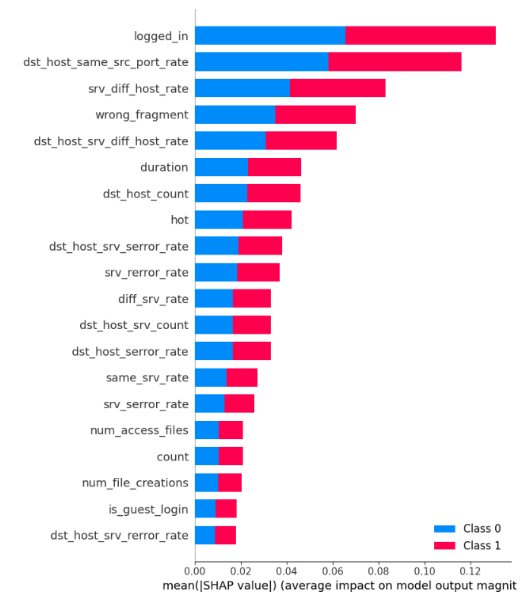

Figure 4. Global Explanation on a subset of Defense Test Set

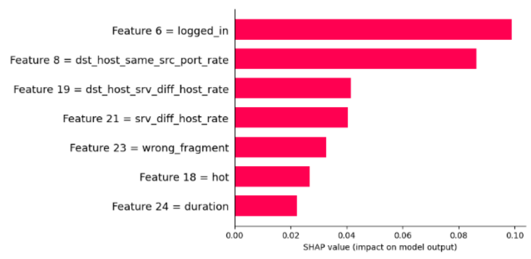

Figure 5. Local Explanation runs on a normal instance in Defense Test Set

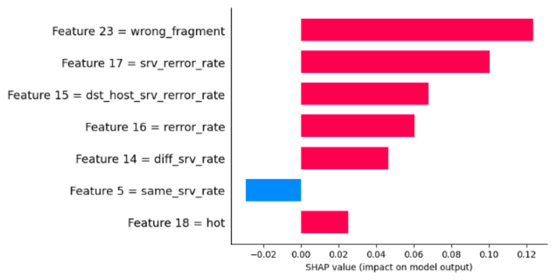

Figure 6. Local Explanation runs on an AE attack instance in Defense Test Set

is quite small, leading to a poor list of predefined features. Lastly, sampling 10,000 samples from the preprocessed KDDTrain+ dataset in Step 3, instead of selecting attack instances, resulted in an unqualified Defense Set.

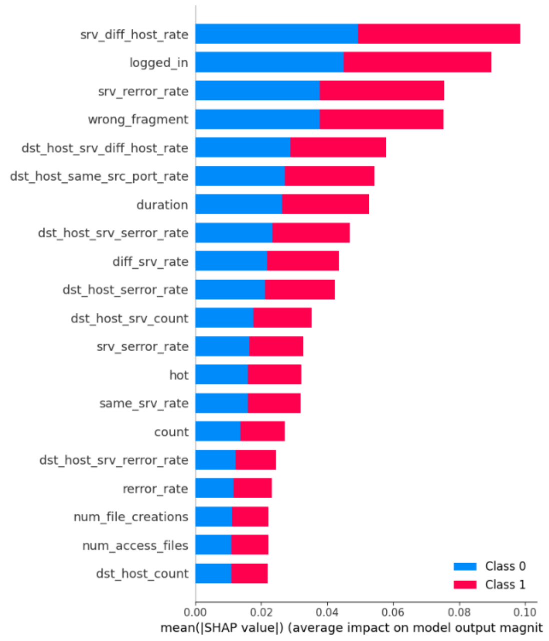

Figure 7. Global Explanation runs on a subset of Random Test Set

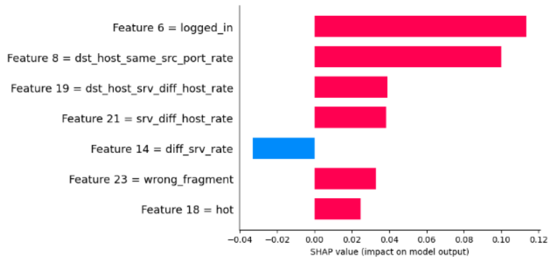

Figure 8. Local Explanation runs on a normal instance of Random Test Set

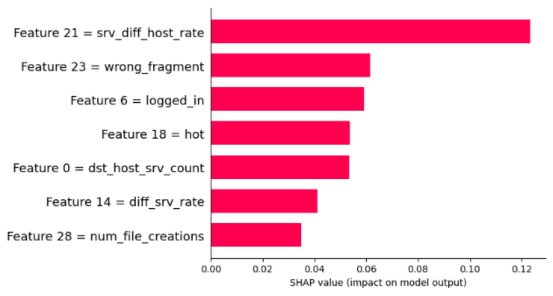

Figure 9. Local Explanation runs on an AE instance of Random Test Set

Figure 4 and Figure 7 show the global explanation of the CNN's predictions on the Defense Test Set and Random Test Set, respectively. The y-axis represents features with the highest impact on the CNN, while the x-axis represents the mean SHAP values of these features. The main purpose of the global explanation charts is to illustrate which features have the most significant impact on the model's predictions. Based on Figure 4, we can see that the most influential features in order are 'logged_in', 'dst_host_same_src_port_rate', 'srv_diff_host_rate', etc. Whereas Figure 7 shows that 'srv_diff_host_rate', 'logged_in', 'srv_rerror_rate', etc., in order, are the most important features that impact the AE detector.

The local explanations of a benign instance prediction and an attack instance prediction in the Defense Test Set are shown in Figure 5 and Figure 6 display, respectively. The y-axis shows features that have the most impact on the prediction, and the x-axis shows the values of these features. Figure 8 and Figure 9 display the local explanations of a benign instance prediction and a malicious instance prediction in the Random Test Set. These explanations increase our understanding of the AE detector's decision-making process within the context of what features contribute to the output and how much they influence it. In the Defense Test Set, Figure 5 represents 'logged_in' and 'dst_host_same_src_port_rate' that they have an impressive effect on the 'normal' outcome of the AE detector, and Figure 6 shows 'wrong_fragment' and 'srv_rerror_rate' that they have the most impact on the `attack' outcome of the AE detector. In the Random Test Set, Figure 8 represents 'logged_in' and 'dst_host_same_src_port_rate' that are the most impactful features that lead AE detectors predict as normal, and Figure 9 clarifies that a most significant feature to the attack output of the AE detector is 'srv_diff_host_rate'.

5. CONCLUSION

In this paper, we have proposed an interpretable approach for trustworthy intrusion detection systems against evasion samples. We used the DNN model to classify the collected network data and the CNN model as an AE detector to identify evasion samples. SHAP served as the explanation method, generating a predefined feature list and providing explanations for both the DNN IDS and the AE detector. Our experiment employed the NSL-KDD dataset and demonstrated the successful detection of AE samples from common attacks like FGSM, BIM, JSMA, and DeepFool using the Defense Test Set. The outcomes of our practice experiment using a Random Test Set, however, were poor and highlight areas that need to be improved in future studies. SHAP helped us define the most important features for generating a predefined feature list and making the prediction of the DNN-IDS and the AE detector transparent and understandable. Moreover, we discovered how the performance of DNN-IDS will be affected by increasing the number of features targeted by AE attacks. Increasing the most or least significant features both reduces the performance of the IDS, and increasing the number of most important features will decrease the effectiveness of the IDS faster than increasing the number of least important features.

ACKNOWLEDGMENT

This research was supported by the VNUHCM-University of Information Technology's Scientific Research Support Fund.

REFERENCES

ART. (n.d.). Retrieved from Adversarial Robustness Toolbox: https://github.com/Trusted-AI/adversarial-robustness-toolbox

Capuano, N. A. (2022). Explainable artificial intelligence in cybersecurity: A survey. *IEEE Access*, 93575--93600.

Carlini, N. A. (2017). Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (sp)* (pp. 39--57). IEEE.

Chawla, N. V. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of artificial intelligence research*.

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

Ko, G. A. (2021). Unsupervised detection of adversarial examples with model explanations. *arXiv preprint arXiv:2107.10480*.

Le, T.-T.-H. A. (2022). Classification and explanation for intrusion detection system based on ensemble trees and SHAP method. *Sensors*, 1154.

Liang, H. A. (2022). Adversarial attack and defense: A survey. *Electronics*, 1283.

Lundberg, S. M. I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems, 30*.

Moosavi-Dezfooli, S.-M. A. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition.*

Otoum, Y. A. (2022). DL-IDS: a deep learning--based intrusion detection framework for securing IoT. *Transactions on Emerging Telecommunications Technologies, e3803*.

Papernot, N. A. (2016). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)* (pp. 372--387).

Peng, J. A. (2022). An trustworthy intrusion detection framework enabled by ex-post-interpretation-enabled approach. *Journal of Information Security and Applications, 103364*.

SHAP. (n.d.). Retrieved from SHAP: https://github.com/shap/shap

Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R., & Chen, J. (2020). DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system. *Security and communication networks, 2020*, 1-11.

Wang, J. (2021). Adversarial examples in physical world. In *IJCAI* (pp. 4925-4926).

Wang, N. A. (2022). Manda: On adversarial example detection for network intrusion detection system. *IEEE Transactions on Dependable and Secure Computing*, 1139-1153.

Wilson, D. (2020). From explanations to feature selection: assessing SHAP values as feature selection mechanism. In *2020 33rd SIBGRAPI conference on Graphics, Patterns and Images (SIBGRAPI)* (pp. 340-347).