The Devil is in the Details: Confident & Explainable Anomaly Detector for Software-Defined Networks

Tapadhir Das*, Raj Mani Shukla † and Shamik Sengupta*
*Department of Computer Science and Engineering, University of Nevada, Reno, USA
†Department of Computer Science, University of Bristol, UK
Email: tapadhird@nevada.unr.edu, raj.shukla@bristol.ac.uk, ssengupta@unr.edu

2021 IEEE 20th International Symposium on Network Computing and Applications (NCA) | 978-1-6654-9550-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/NCA53618.2021.9685157

***Abstract*—Deployment of SDN control plane in high-end servers allow many network applications to be automated and easily managed. In this paper, we propose an SDN anomaly detection application, Confident and Explainable Anomaly Detector (CEAD), that automatically detects malicious network flows in SDN-based network architectures. The proposed application employs a set of Machine Learning (ML) classifiers to improve the confidence score of a prediction, thereby creating improved trust upon the prediction, while providing interpretability to the anomaly detector. The method utilizes the Explainable Artificial Intelligence (XAI) framework to provide interpretation to predictions to unearth network features that establish the most influence between predicted anomaly types. Results show that the proposed framework can achieve efficient anomaly detection performance, with near perfect confidence scores. Analysis with XAI highlights that byte and packet transmissions, and their robust statistics, can be significant indicators for prevalence of any attacks. Results also indicate that a subset of influential features can generally be used to decipher between normal and anomalous flow, while certain dataset features can be specifically influential in detecting specific attack types. This can lead to more efficient network resource utilization.**

***Index Terms*—Software-defined networking, machine learning, anomaly detection, explainable Artificial Intelligence**

I. Introduction

Network anomalies refer to scenarios where network operations and metrics deviate from normal system behavior. These can arise due to internal factors such as device malfunction, environmental issues, and network overload. Network anomalies can also rise from cyber-attacks including, but not limited to, Denial of Service (DoS), malicious traffic insertion, and network intrusion. Anomalies tend to resemble normal network behavior and are problematic as they can leave communication vulnerable, leading to loss in revenue, reputation, and intellectual property.

Software-defined networks (SDN) decouple the control and data planes, where a logically centralized SDN controller administers the wide-scale network of switches. The SDN controller is also responsible for injecting flow tables in network switches which control network operations and policies. This makes it easier to enact network policy alterations, upgradations, and monitoring [1]. It also makes SDN-based network data mining and analysis simpler. This further assists in solving security problems like anomaly detection, as a large-scale network can be easily monitored.

With improved computational capabilities, the use of machine learning (ML) for anomaly detection in networks has also increased [2]. ML is a useful method for anomaly detection due to its capability of finding correlations in traffic flow patterns [3]. However, a common problem associated with ML-based predictions is low confidence scores, which make it difficult to ascertain the certainty of predictions. A high confidence level, along with high accuracy, can allow an ML system to be used in a live environment as the predictions are forecasted correctly with high probability. The high confidence in an ML prediction assists to automate the changes in SDN flow tables that govern network policies.

Another limitation with ML-based systems is the lack of explainability. Many studies in literature have conducted ML analysis in network anomaly detection [3] [4]. However, the research was conducted in a "black box" manner, where the ML approach lacked transparency and explainability. This can be problematic in network anomaly detection as users will not be able to explain the cause of the anomaly. An interpretation of the ML-based anomaly detector makes it easier for the system to understand which of the network features are more influential at detecting anomaly types. Explainability can aid to improve trust on the anomaly detector's predictions; specifically, the relevance of the dataset features, confidence of the predictions, and justification of the results [5]. Therefore, the major contributions of this work are as follows:

- We present an ML pipeline, Confident and Explainable Anomaly Detector (CEAD), that uses different models to improve the confidence score of an SDN-based anomaly detector.
- Additionally, we incorporate an Explainable Artificial Intelligence (XAI) framework to interpret the type of predictions made by the proposed ML pipeline.
- We test the proposed pipeline on diverse SDN network datasets to uncover dataset features that maximally influence predictions.

The rest of the paper is structured as follows: Section II provides the literature study and related works. Our proposed methodology is provided in Section III. Section IV demonstrates our simulation, results, and analysis. Finally, conclusions are drawn in Section V.

This research is supported by the National Science Foundation (NSF) Award #2019164.

II. Related Works

Anomaly detection methods in SDN are continually being proposed to provide improved performance against cyber threats. The researchers in [6] addressed anomaly detection in SDNs using a Hidden Markov Model (HMM). HMMs are prone to inefficient anomaly detection if the anomaly is uncommon [7]. Other anomaly detection studies have been investigated using statistical approaches [8] [9]. In [8], the authors used Principal Component Analysis (PCA) to analyze SDN network traffic and detect potential Distributed DoS (DDoS) attacks. In [9], researchers employed Discrete Wavelet Transform to identify anomalies in an SDN. The limitation with statistical approaches is that selected anomaly thresholds do not properly represent real world threats due to their limited and static nature [7].

Unsupervised ML approaches like clustering [10] [11] and supervised techniques like deep learning [3] [4] have also been investigated for SDN anomaly detection. The authors in [10] used a distributed SDN and k-means clustering to achieve optimal anomaly detection performance in smart grid communications. [11] utilized clustering to classify SDN network traffic into safe and unsafe clusters. The limitation of clustering algorithms is that they could get stuck in local minima and provide incorrect predictions [7]. The authors in [3] developed a deep neural network for anomaly detection for SDN networks. In [4], the researchers proposed a Gated Recurrent Unit Recurrent Neural Network for anomaly detection in SDN. Like mentioned previously, these studies were conducted in a "black box" approach and provided no interpretability. This lack of interpretability is detrimental as it complicates the process for knowledge discovery for the network administrators and detracts them from diagnosing, debugging, and understanding the reasons behind the predictions. This can lead to increased mistrust on the anomaly detector. Also, in the cases of data driven anomaly detection, if the ML model is presented with new attack types that have no previous instance, the anomaly detector can misclassify the attack as normal and provide no clear explanation for their decision, leading to a contravention [12].

The concept of XAI in SDN networks has received limited attention [13], [14], [15]. In [13], the authors used a variational autoencoder for anomaly detection along with a gradient based fingerprinting technique for explainability. The researchers in [15] developed an SDN anomaly detection approach using a Random Forest classifier which incorporated explainability. In [14], the researchers investigated multiple classifiers with the SHapley Additive exPlanations (SHAP) framework to explain detected SDN anomalies in a single dataset. The above studies focused on anomaly detection performance on singular algorithms and their associated explanations, on isolated datasets.

In contrast, our proposed ML pipeline not only focuses on model accuracy but also on providing better confidence scores. In addition, our pipeline provides the interpretation of the predicted cyber-attacks and anomalies. We also conduct analysis to see which features maximally influence cyber-attack predictions. Thus, while better confidence scores and model interpretation help in automation of anomaly detector while it is running online, the feature analysis will help network administrators better design policies offline.

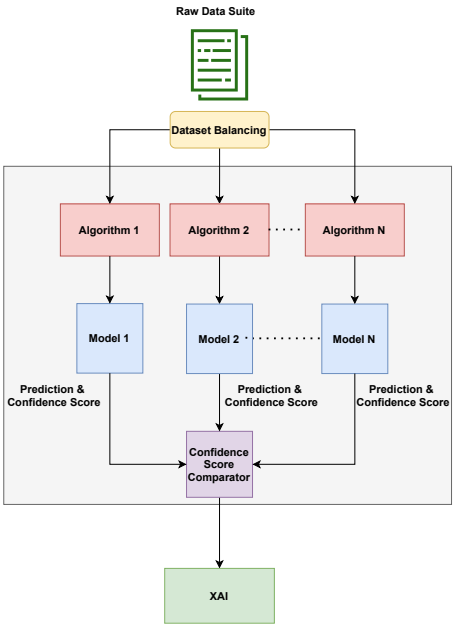

Fig. 1: Proposed ML pipeline for anomaly detection and interpretation

III. Methodology

The proposed approach for CEAD primarily consists of a high confidence anomaly detector whose predictions are then interpreted by an XAI framework, from which the influential features are logged. Figure 1 illustrates our proposed methodology.

A. Raw dataset for ML pipeline development

The proposed approach has wide network applications and is agnostic to the dataset features. Let the network dataset be denoted using the variable X, consisting of E total samples. x_i represents a single data sample and $x_i \in X, i = \{0, 1...E\}$. The full data suite X is partitioned into the training X_{train} and test X_{test} sets. We assume the total number of training samples is U, while total number of testing samples as V. After preparation of training and test sets for the ML pipeline, we perform data balancing. It should be noted that X_{test} is used in this paper, primarily, to showcase testing purpose and performance evaluation. However, in a live system for which this pipeline is intended, the input to the model would be network features in real-time. Based on the network features as input, the attack type and its interpretation are computed.

B. Data balancing

We balance the data in training set to avoid overfitting due to majority classes. Drastic class imbalance can be known to degrade anomaly detection performance, which is why data balancing is required. Data balancing is performed using the random oversampling method. Random oversampling involves

augmenting the dataset with multiple copies of minority classes to provide more representative data samples for minority classes in X_{train}. This ensures that both minority and majority classes are proportionate.

C. ML pipeline anomaly detection and classification

Following dataset balancing, we develop the ML pipeline for prediction, interpretation, and log information. The proposed pipeline combines many ML classifiers. Let an ML classifier be represented by μ and the total number of classifiers be denoted as N. Then, each classifier is trained on X_{train}, to obtain a model M_j that is saved and stored:

$$M_j = \mu_j(X_{train}), j = \{1, 2.....N\} \tag{1}$$

Once the ML classifiers are trained, the values in X_{test} is parsed through every trained ML classifier to obtain the predicted type of the anomaly γ_j and the confidence level ϵ_j of the associated prediction:

$$\gamma_j, \epsilon_j = M_j(X_{test}) \tag{2}$$

After parsing, the next step is comparing the confidence scores.

D. Confidence Score Comparator

In this module, the aim is to determine the trained ML model μ_j that provides the best confidence score for each sample $x_i \in X_{test}$. This can be conducted using the following equation:

$$\eta_i = \arg \max_{x_i \in X_{test}} (\epsilon_j), i = \{0, 1.....V\} \tag{3}$$

Here, η_i corresponds to the trained ML classifier μ_j that provides the best confidence score and corresponding prediction γ_j, for the testing data sample x_i.

E. Prediction interpretation using XAI

The detected attack type is interpreted using an XAI framework. For our technique, we use Local Interpretable Model-agnostic Explanations (LIME) [16]. We perform interpretability analysis for X_{test}, using each testing sample x_i's optimally trained ML classifier η_i. We introduce β that represents the fitted and simple, interpretable model around the data sample $x_i \in X_{test}$ and supplies the local explanations of the classifier η_i and operates along the same interpretable representation as x_i. The interpretable representation of x_i is denoted as x_i'. Therefore, the explanation τ is provided using:

$$\tau(x_i) = \arg \min_{\beta \in B} \Lambda(\eta_i, \beta, \pi_{x_i}) + \Omega(\beta) \tag{4}$$

Here, B represents the family of explainable linear models. The loss function of the framework is symbolized by Λ, while π_{x_i} denotes the locality around the value of x_i. Finally, Ω represents the complexity penalty of the local model β. The locally weighted square loss function Λ is represented by:

$$\Lambda(\eta_i, \beta, \pi_{x_i}) = \sum_j \pi_{x_i}(z_j)(\eta_i(z_j) - \beta(z_j'))^2 \tag{5}$$

which denotes the weighted euclidean distance where the sum iterates over a set of sample perturbed points z around x_i, $\{(z_k, z_k'), k = 1, 2....M\}$. Here, $z_k \in z$ is a perturbed original data sample while z_k' is the corresponding interpretable representation. The samples are weighed using $\pi_{x_i}(z_k)$, based on their similarity to x_i. Here, $\pi_{x_i}(z_k)$ is represented as:

$$\pi_{x_i}(z) = exp(-D(x_i, z)^2/\sigma^2) \tag{6}$$

where D represents a distance function with width σ. Many open-sourced XAI frameworks use a Lasso regression [17], as the distance function D, to prompt any sparsity in explanations and depends on a hyper parameter to restrict their explorations and complexity. In this case, the hyper parameter is K that represents the number of most influential features for the local interpretable explanation. After performing Equations 4, 5, 6, the resultant value for each $\tau(x_i) = \{F_0, F_1, F_2, ...K\}$, where F represents the influential dataset feature for sample x_i. Lastly, we must compute the complete list of influential features within the whole testing dataset. This is computed using a frequency counter that is represented by:

$$\phi = \sum_{i=0}^{V} \tau(x_i) \tag{7}$$

where ϕ represents the list of the most influential dataset features or features that were most frequently encountered by the XAI algorithm when providing local interpretable explanations throughout X_{test}.

IV. Simulation and Results

A. Setup

Our proposed methodology was implemented in python and utilizes the tensorflow, sklearn, imblearn, and LIME frameworks. For our data, we used the NSL-KDD [18] and CICIDS-2017 [19] datasets. The features in these datasets contain network metrics that can be gathered from any SDN setup, like inter-arrival time, source and destination bytes etc. Any customary classification algorithm can be chosen for the ML classifiers. We selected random forest (RF), multi-layer perceptron (MLP), and support vector machine (SVM) classifiers.

B. Experimentation

In our experiments, we set up the following initial values: $N = 3, K = 5, L = 1000, M = 1000$. All the datasets were split using a 70%-30% ratio between training and testing data, respectively. For performance metrics, we are using accuracy and confidence scores.

*1) **Pipeline Accuracy and Performance**:* First, we look at the performance achieved by our proposed method. Table I provides the metrics and performance achieved between RF, MLP, SVM, and CEAD. From Table I, we can see that SVM gives the least accuracy on both datasets, followed by the MLP classifier. The proposed CEAD classifier gives better accuracy than the SVM and MLP. CEAD also provides comparable accuracy against the RF, which gives the best accuracy across

Dataset	Classifier	Accuracy	Mean Confidence Score
NSL-KDD	RF	99.82 %	0.979
	MLP	96.09 %	0.997
	SVM	93.06 %	0.595
	CEAD	97.25%	0.998
CICIDS-2017	RF	99.86 %	0.994
	MLP	84.44 %	0.824
	SVM	84.16 %	0.451
	CEAD	98.94 %	0.995

TABLE I: Performance of ML classifiers using NSL-KDD and CICIDS-2017 datasets

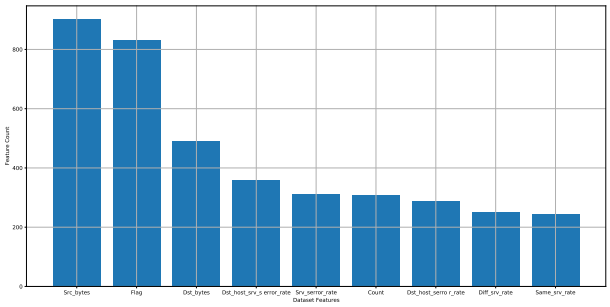

Fig. 2: Most influential dataset features in NSL-KDD

the SDN datasets. However, the CEAD achieves the best mean confidence score for their predictions out of all the ML classifiers. This can be attributed to the fact that the CEAD prioritizes decisions made with the highest confidence score and associates a dataset sample with the classifier that provided the best confidence score, in contrast to the RF, MLP, and SVM.

*2) **XAI Results**:* Next, we observe the raw data sample and the XAI explainability results obtained from the CEAD framework. This is achieved by running individual and random testing samples through the approach to get their explainability and most influential features. Due to resource constraints, we adapted our testing dataset X_{test} for this XAI analysis. Instead of performing experiments on the entirety of the testing data, we performed them on a subset of the testing data. For adapting to the new format, we introduced two new variables. L represents the number of new samples in the testing dataset and M represents the iterations for sample gathering. To generate the new testing data, we start by randomly selecting L samples from X_{test}, per iteration. In total, we run this random selection over M iterations to generate a total of $L * M$ testing samples. We, then, check the frequency of all the randomly selected X_{test} samples. From here, we select the top L most frequently occurring random data samples as our testing data. The remainder of the analysis has been conducted using this adapted testing dataset. We begin by analyzing the CEAD framework's ability to discover dataset features that are influential for predictions, for all labels, in the whole testing dataset, and the frequency with which these features are encountered. This analysis is illustrated in Figures 2 and 3.

In Figure 2, we can see that across all label types, *Src_bytes* and *Flag* were the two most influential dataset features, followed by *Dst_bytes*. *Src_bytes* measures the number of

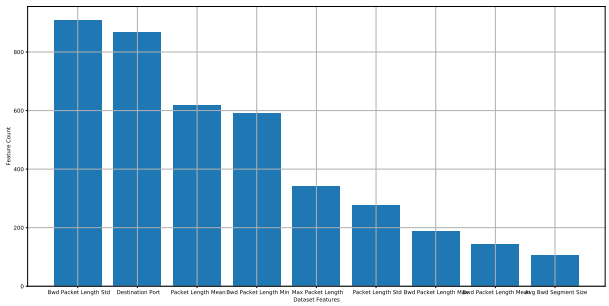

Fig. 3: Most influential dataset features in CICIDS-2017

data bytes transferred from source to destination in a single connection. *Flag* indicates the status of the connection, normal or error. *Dst_bytes* represents the number of data bytes transferred from destination to source in a single connection. From this, we can note that the number of data bytes or packets transmitted in a single connection within this SDN, along with the status of the connection, are the highest indicators regarding if this SDN is under attack.

In Figure 3, we can see that, across all label types, *Bwd Packet Length Std*, *Destination Port*, and *Packet Length Mean* were the most influential dataset features. *Bwd Packet Length Std* measures the standard deviation size of a packet from server to client. *Packet Length Mean* represents the mean length of a packet in a flow. From this, we can note that the average length and statistics of packets in a network flow along with the destination port are the influential features to indicate if this SDN is under attack.

Next, we analyze the CEAD framework's ability to discover dataset featues that are influential for predictions, for each label, and the frequency with which these features are encountered. This analysis is illustrated in Figures 4 and 5. In Figure 4, we can see that, across all label types, certain dataset features are specifically influential. The feature counts for labels *neptune* and *normal* are higher than the rest of the labels, due to an increased number of samples. Also, we note that all the labels exhibit similar behavior patterns across the feature indexes, i.e., certain features are counted more often than others across all label types. Also, certain features are more influential than others in detecting different attack types. For example, feature indexes 33 and 35 are more influential when detecting *neptune* or TCP SYN attacks. Feature indexes 9 and 12 are more influential when detecting *ipsweep* or ICMP sweep attacks on the network.

Similarly, in Figure 5, we can see that, across all label types, certain dataset features are specifically influential. The feature counts for label *BENIGN* is higher than the rest of the labels, due to an increased number of samples. Also, we note that all the labels exhibit similar behavior patterns across the feature indexes, i.e., certain features are counted more often across all label types. Also, certain features are more influential than others in detecting different attack types. For example, in the CICIDS-2017 dataset, feature indexes 11, 13, and 40 are more influential when detecting *PortScan* attacks and *DoS-Hulk* or Hulk flood attacks.

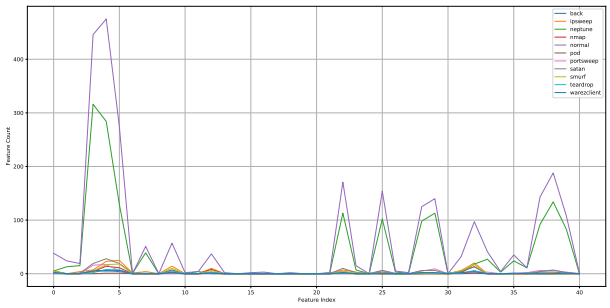

Fig. 4: Most influential dataset features, per label, in NSL-KDD

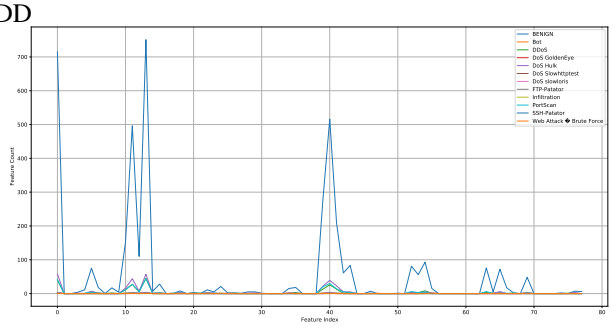

Fig. 5: Most influential dataset features, per label, in CICIDS-2017

From a complete overview, across two separate SDN datasets, the transmission of bytes and packets and their correlated statistics can be significant indicators if an SDN is working normally or under attack. This correlates with contemporary knowledge that if the productivity of a network degrades, something is hindering it. We also note that across both datasets, all labels are generally influenced by similar features. Additionally, in both datasets, we have noted that specific features can be very influential in detecting specific anomalies. These insights can have a positive influence as an SDN based anomaly detection algorithm can primarily focus on these significant features, thereby reducing operational overhead.

V. Conclusions

In this paper, we proposed an ML-based pipeline, CEAD, for SDN network anomaly detection that is accurate, has high confidence, and interpretable. We tested the proposed pipeline on different SDN network datasets. Results show that the proposed CEAD framework achieves better performance than sole SVM or MLP classifiers, while comparable results to an RF. The proposed pipeline also achieves the highest mean confidence score between all algorithms. Further analysis across the datasets using XAI framework has highlighted that network byte and packet transmissions and their robust statistics can be significant indicators for prevalence of any attack. We also observe that differentiation between normal and attack flow can be consolidated to a general subset of influential dataset features. Additionally, we have also noted that specific feature indexes can be influential in detecting specific attack types. This knowledge can help limit the number of observed features, consequently reducing operational overhead. This information can be used by the network administrators to provide better design and decisions for the communication systems.

References

[1] R. M. Shukla, S. Sengupta, and M. Chatterjee, "Software-defined network and cloud-edge collaboration for smart and connected vehicles," in *Proceedings of the Workshop Program of the 19th International Conference on Distributed Computing and Networking*, 2018, pp. 1–6.

[2] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," *arXiv preprint arXiv:1803.10769*, 2018.

[3] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 international conference on wireless networks and mobile communications (WINCOM)*. IEEE, 2016, pp. 258–263.

[4] ——, "Deep recurrent neural network for intrusion detection in sdn-based networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 202–206.

[5] K. Amarasinghe, K. Kenney, and M. Manic, "Toward explainable deep neural network based anomaly detection," in *2018 11th International Conference on Human System Interaction (HSI)*. IEEE, 2018, pp. 311–317.

[6] T. Hurley, J. E. Perdomo, and A. Perez-Pons, "Hmm-based intrusion detection system for software defined networking," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 617–621.

[7] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommunication Systems*, vol. 70, no. 3, pp. 447–489, 2019.

[8] L.-q. Han and Y. Zhang, "Pca-based ddos attack detection of sdn environments," in *International conference on Big Data Analytics for Cyber-Physical-Systems*. Springer, 2020, pp. 1413–1419.

[9] C. B. Zerbini, L. F. Carvalho, T. Abrao, and M. L. Proenca Jr, "Wavelet against random forest for anomaly mitigation in software-defined networking," *Applied Soft Computing*, vol. 80, pp. 138–153, 2019.

[10] A. Starke, J. McNair, R. Trevizan, A. Bretas, J. Peeples, and A. Zare, "Toward resilient smart grid communications using distributed sdn with ml-based anomaly detection," in *International Conference on Wired/Wireless Internet Communication*. Springer, 2018, pp. 83–94.

[11] M. Monshizadeh, V. Khatri, and R. Kantola, "An adaptive detection and prevention architecture for unsafe traffic in sdn enabled mobile networks," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017, pp. 883–884.

[12] D. L. Marino, C. S. Wickramasinghe, and M. Manic, "An adversarial approach for explainable ai in intrusion detection systems," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 3237–3243.

[13] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "Gee: A gradient-based explainable variational autoencoder for network anomaly detection," in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 91–99.

[14] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73 127–73 141, 2020.

[15] A. K. Sarica and P. Angin, "Explainable security in sdn-based iot networks," *Sensors*, vol. 20, no. 24, p. 7326, 2020.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[17] D. Valcarce, J. Parapar, and Á. Barreiro, "Lime: linear methods for pseudo-relevance feedback," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 678–687.

[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009, pp. 1–6.

[19] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.