Efficient Early Anomaly Detection of Network Security Attacks Using Deep Learning

Tanwir Ahmad and Dragos Truscan
Åbo Akademi University, Turku, Finland
Email: firstname.lastname@abo.fi

***Abstract*—We present a deep-learning (DL) anomaly-based Intrusion Detection System (IDS) for networked systems, which is able to detect in real-time anomalous network traffic corresponding to security attacks while they are ongoing. Compared to similar approaches, our IDS does not require a fixed number of network packets to analyze in order to make a decision on the type of traffic and it utilizes a more compact neural network which improves its real-time performance. As shown in the experiments using the CICIDS2017 and USTC-TFC-2016 datasets, the approach is able to detect anomalous traffic with high precision and recall. In addition, the approach is able to classify the network traffic by using only a very small portion of the network flows.**

I. Introduction

With the increasing number of attacks against network systems such as web applications, network *intrusion detection systems* (IDSs) have become an important tool for identifying unauthorized and malicious network traffic and for triggering countermeasures [1]. IDSs can be seen as largely falling under two categories: *signature-based* and *anomaly-based* detection systems [2]. The former is able to detect attacks by comparing the network traffic against the signatures of known attacks. Although they are widely used [3] and proven to be efficient [2], [4] in detecting known attack types, they are not able to detect new types of attacks or attacks that they were not trained for. Instead, anomaly-based IDSs are used to distinguish abnormal network traffic from normal one, which allows them to detect both known and new types of attacks. Typically the classification of the traffic is based on a predefined anomaly threshold which dictates if the given network traffic data is anomalous (intrusive). However, one challenge stands in defining the anomaly threshold such that it provides an acceptable level of accuracy without excessive false alerts.

Furthermore, the usefulness of an IDS increases if it is able to detect the intrusion as soon as possible before it is complete and before it creates irreversible damage. Early detection would also allow system administrators or automated tools to deploy mitigation actions and countermeasures in a timely manner. However, only very few works consider early real-time detection of intrusions.

In this paper, we present *Early-A*, a deep-learning (DL) anomaly-based IDS for networked systems, which is able to detect in real-time anomalous network traffic corresponding to security attacks while they are ongoing. Compared to similar approaches, our IDS does not require a fixed number of network packets to analyze in order to make a decision on the type of traffic and it requires a more compact neural network. In addition, the approach is able to classify the network traffic by using only a very small portion of the network flows.

In the remainder of the paper, we will introduce the approach in Section II, then we evaluate it via a set of experiments on a case study in Section III. We discuss related work in Section IV and we conclude in Section V.

II. Approach

In this section, we present our early anomaly-based IDS, called Early-A, for identifying network attacks. Unlike our previous approach, Early-A classifies network flows as either normal or anomalous. It learns the properties of normal (or benign) network traffic and considers the given network traffic anomalous (or malicious) whenever the traffic deviates beyond a certain threshold from the known normality distribution. In principle, Early-A does not require any prior knowledge about the anomalies, and it is capable of discovering new anomalies.

The Early-A approach (Figure 1) is composed of four main modules:

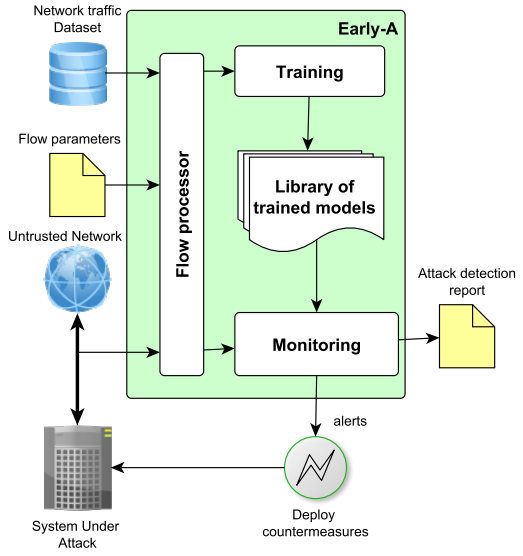

Fig. 1. Overview of Early-A approach

- a **flow processing module** is used to extract flows from network traffic based on given flow parameters, and

provides the flows to other modules, either for training or for monitoring;
- a **training module** is used to train neural network models using various datasets for different application domains (e.g, web, IoT, etc.);
- a **library of attack models** contains trained models for different attack types and application domains;
- a **monitoring module** which is used to monitor network traffic using the corresponding trained model from the library. Whenever attacks are detected, this module will trigger alerts based on predefined triggers in order to deploy automatic countermeasures.

The approach works at the network packet level. It analyzes the network traffic and extracts and extracts *network flows* for analysis. A *network flow* is a bidirectional sequence of packets exchanged between two endpoints (e.g., a web server and a client) during a certain time interval with some common flow properties [5] such as source and destination IP addresses, source and destination port numbers, and the protocol type. In our work, we define a network flow as a sequence of T ordered packets, where T represents the length of a complete flow. A flow is denoted as:

$$\mathbf{f} = \{p_1, p_2, ..., p_T\},\ \forall\ p_i \in \mathbb{R}^D \wedge 1 \leq i \leq T \qquad (1)$$

where D is the dimension (or length) of a packet.

A. Flow processing

In order to extract the flows from raw network traffic, a flow processing pipeline performs the following operations (see Figure 2):

- **packet filtering** - selects the network packets used for analysis based on a set of criteria such as protocol, port, etc;
- **flow identification** - create and maintain network flows for packets based on their source and destination IP addresses. There are two types of flows: active and passive. A flow is considered *active* if a packet was added to it in a predefined period of time (ie., flow expiration timeout), otherwise, it becomes *inactive*. If an active flow already exists the packet is added to it otherwise a new active flow is created;
- **packet pre-processing**
  - ***truncation*** - Media Access Control (MAC) address (i.e., used for transferring the frames between different nodes in the network) and the Internet Protocol (IP) header (containing information such as the total length of the packet, protocol version, source, and destination IP addresses) are removed from the packets. The truncated information is necessary for routing packets in the network. However, we consider this information irrelevant and counter-productive for our classifier since there is a chance that the classifier will start relying on the IP information (e.g., IP addresses) for detecting attack flows. Therefore, we remove it from the packets.

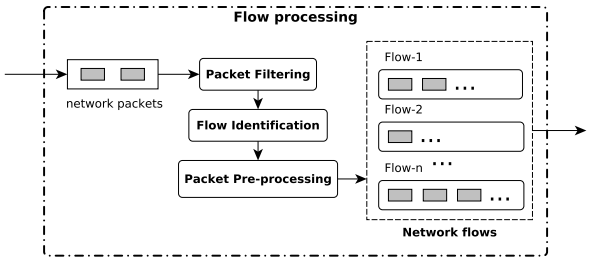

Fig. 2. Flow processing module

  - ***transformation*** - pad with zeros or crop the packet to a fixed length. We would like to highlight that, even though we fix the length of the packets; we do not restrict the length of a flow (i.e., number of packets) unlike many other state-of-the-art approaches though it is implicitly bounded by time out.

This flow processing pipeline is used in the approach both for training and for monitoring intrusion detection as follows.

B. Training

For training, we require a labeled flow dataset for supervised training that mainly contains normal flows and few attack ones. The dataset should also have raw network data corresponding to the flows. This phase is composed of two steps.

1) Data set augmentation: In order to train the classifier capable of reliably detecting the attack flow after observing the first few packets out of a given flow, we extend the dataset by cumulatively creating short segments (prefixes) of a flow at different lengths. This approach will ensure that the classifier will be able to recognize also prefixes of anomalous flows, and consequently increase its early detection capability. The details of our data augmentation process can be found in our previous work [6].

We denote a flow dataset as

$$S = \{(\mathbf{f}_1, y_1), (\mathbf{f}_2, y_2), ..., (\mathbf{f}_N, y_N)\} \qquad (2)$$

where N represents the total number of flows $\mathbf{f}$ and their corresponding labels $y \in \{0, 1\}$. The label y is 0 for normal and 1 for attack flows.

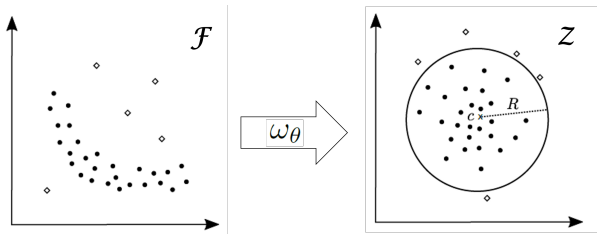

Fig. 3. Generic deep one class classification example adapted from [7]. Full circles represent nominal flows, whereas empty diamonds represent anomalous flows.

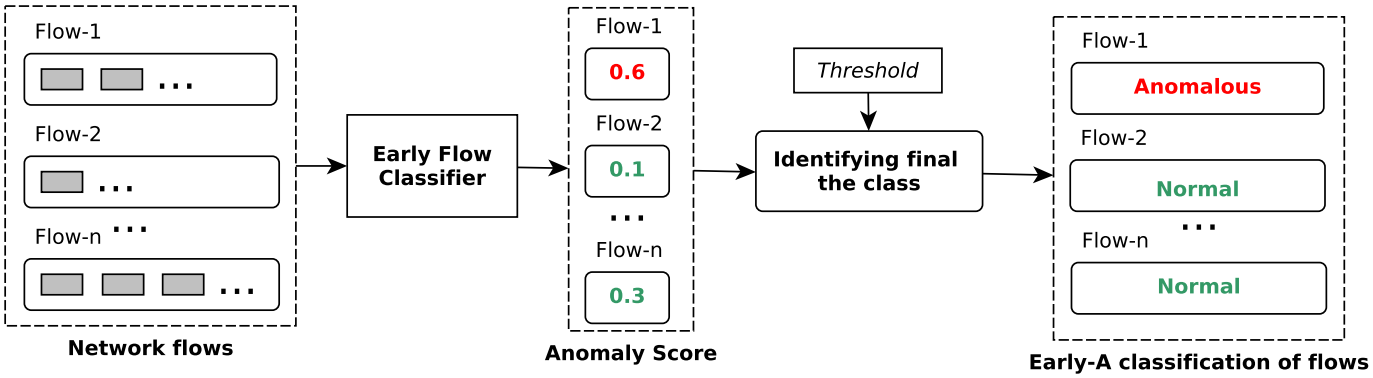

Fig. 4. Early-A approach for classification of network flows

2) Network training: We train a neural network model ω_θ with parameters θ following the Deep One-Class Classification [8] method. The model learns a transformation $\omega_\theta : \mathbb{R}^{D\times T} \rightarrow \mathbb{R}^k$ where $\mathcal{Z} \subseteq \mathbb{R}^k$ and k specifies the dimensions of the output space $\mathcal{Z}$.

The goal of the transformation is to bring the normal flows from an input space $\mathcal{F}$ close to a center $\mathbf{c}$ in the output space $\mathcal{Z}$ and anomalous flows away from the center, as shown in Figure 3. In our case, we set the center $\mathbf{c}$ to the origin (i.e., $\mathbf{c} = 0$) of the hyper-sphere enclosing the training samples in the output space. We can formulate the objective as the following loss function $\ell(\omega_\theta, \mathbf{y})$ [8]:

$$\frac{1}{N}\sum_{i=1}^{N}(1-y_i)h(\omega_\theta(\mathbf{f}_i)) - y_i \log(1-\exp(-h(\omega_\theta(\mathbf{f}_i)))) \quad (3)$$

$$\text{where } h(z) = \sqrt{||z||^2+1} - 1 \quad (4)$$

C. Intrusion Detection

After training the network, we calculate the anomaly score for a given flow $\mathbf{f}$ using the $score(\mathbf{f}) \in [0,1]$ function:

$$score(\mathbf{f}) = 1 - \exp(-h(\omega_\theta(\mathbf{f}))) \quad (5)$$

The anomaly score of a given flow represents its distance from the center. We expect the anomaly score of normal flows to be lower than the scores of anomalous flows. In this sense, a flow is considered to be anomalous or malicious if the anomaly score exceeds a certain threshold.

A *threshold* value, corresponding to the radius R of the hyper-sphere in Figure 3, can be determined using different methods. In an ideal scenario where the training dataset does not contain outliers, the anomaly scores of the normal and anomalous flows will not overlap. Thus, we can set the highest anomaly score obtained by a normal flow as a threshold in order to achieve 1 recall (i.e., the percentage of actual anomalous flows that were correctly classified) at 0 false positive rate (i.e., the proportion of normal flows wrongly predicted as anomalous over the total number of normal flows). However, in practice, some outliers are expected to be present in the training dataset. Therefore, we use the 99th percentile of the anomaly score distribution of the normal flows in the training dataset. The threshold value can be provided or fine-tuned by system administrators who observe the network traffic to get desired results. We would like to point out that as we increase the threshold value, the false positive rate decreases, but the precision of the classifier degrades as well.

We keep track of active flows and their corresponding predictions made by our early flow classifier (as depicted in Figure 4). Whenever a new packet is added to a network flow, the early flow classifier employs the model to determine the anomaly score. The final class of the flow is selected based on the score and the predetermined threshold.

III. Evaluation

The following section outlines the evaluation of our approach by addressing two research questions:

- RQ1: How well does our method detect complete flows (i.e., flows that contain all packets)?
- RQ2: What is the efficiency of our approach in recognizing abnormal flows in real-time by analyzing only the first few packets of the flow?

RQ1 evaluates the effectiveness of our approach in identifying anomalous flows, while RQ2 focuses on assessing its performance in a real-time environment. The datasets and model architecture employed in the evaluation are further explained in this section. The results are discussed in the context of each research question.

A. Datasets

To assess the effectiveness of our approach, we utilize two datasets: CICIDS2017 [9] and USTC-TFC-2016 [10]. In CICIDS2017, we focus on a specific segment obtained on Thursday, July 6, 2017. This segment of the dataset comprises network flows related to normal traffic and web attacks such as SQL Injection, Cross-Site Scripting (XSS), and Brute Force. All attack flows are considered anomalous. We should note that this data set is considered highly unbalanced among different attack types. However, this does not affect our approach since we do not make a distinction between different types of attack classes, but we consider all of them as anomalies.

TABLE I
FLOW DATASETS

Class	Number of Flows	Average Flow length
CICIDS2017		
Normal	27 129	124.39
Anomalous	2 180	16.23
USTC-TFC-2016		
Normal	323 556	8.91
Anomalous	300 632	8.11

The USTC-TFC-2016 dataset has ten classes of normal flows: Bittorrent, Facetime, FTP, Gmail, MySQL, Outlook, Skype, SMB, Weibo, and World of Warcraft; and ten classes of malicious flows: Cridex, Geodo, Htbot, Miuref, Neris, Nsis-ay, Shifu, Tinba, Virut, and Zeus. The dataset was collected from a real network environment. All types of malicious flows are considered anomalous and the rest are considered normal.

Table I presents the number of flows and the average flow length (i.e., number of packets) in the datasets. The total number of flows and packets related to those flows in the USTC-TFC-2016 dataset are 624 188 and around 5 million, respectively. Due to the limited amount of computing resources and time, it was not feasible for us to use the entire dataset for training and evaluation. Therefore, to speed up the training and evaluation process, we uniformly sample 10 000 flows from each class.

To address packets with varying header and payload lengths, we either crop or pad them with zeros at the end, resulting in headers and payloads extended to 48 and 400 bytes for the CICIDS2017 dataset and to 36 and 500 bytes for the USTC-TFC-2016 dataset, respectively. Lastly, we normalize all packet bytes between 0 and 1 by dividing them by 255. This practice is commonly used to aid machine learning algorithms in converging faster [11].

B. Network architecture

In our approach, we use a one-dimensional Convolutional Neural Network (1D-CNN) [12] to extract relevant features from the network traffic (see Figure 5). The first layer of the model is the 1D-CNN layer using 32 kernels with size 1, valid padding, *LeakyReLU* activation, and bias. We perform global average pooling to flatten the output of the 1D-CNN layer to a fixed-length vector, which is then provided as input to a fully connected layer with 64 units to get the feature vector. We use the same neural network architecture for both datasets. The total number of trainable parameters of the model for CICIDS2017 and USTC-TFC-2016 are 16 480 and 19 296, respectively.

C. RQ1: Intrusion Detection performance

The objective of this research question is to investigate the classification performance of our approach. To answer this question, our classifier is trained and evaluated against the independent test set that is extracted from the datasets. We split each dataset into two subsets using the ratio 0.7:0.3: training and test set.

We used 10-fold cross-validation on the training dataset to fine-tune the hyper-parameter values and model selection. For statistical reasons, the evaluation procedure is repeated 30 times, and every time, we randomly shuffle the dataset to remove any ordering bias before splitting it into training and test set using stratified sampling [13]. We have augmented the training dataset using the segmentation rate $s_r = 0.1$.

Table II shows the achieved performance of our early flow classifier on the test set. Our approach achieves 0.908 detection rate or recall at 0.032 FPR for the anomalous flows in the CICIDS2017 dataset. In other words, our approach correctly identifies 90.8% of the anomalous flows in the test dataset and wrongly identifies around 3% of normal flows as anomalous flows. Similarly, for the USTC-TFC-2016 dataset, our approach obtains 0.954 detection rate or recall at 0.068 FPR for the anomalous flows. Figure 6 shows the anomaly score distribution with respect to the normal and anomalous flows in each test dataset using the kernel density estimation plot. The vertical black line represents the threshold we calculated to identify anomalous flows. One can notice the overlap between the anomaly scores of the normal and anomalous flows. Therefore, it is often difficult (if not infeasible) to choose the best threshold value which maximizes the detection rate without sacrificing the precision. Overall, our approach performed well and attained 0.92 and 0.943 balanced accuracies for the CICIDS2017 and the USTC-TFC-2016 datasets, respectively.

TABLE II
INTRUSION DETECTION PERFORMANCE

Class	Precision	Recall	FPR	BM
CICIDS2017				
Normal	0.992	0.968	0.092	0.877
Anomalous	0.703	0.908	0.032	0.877
USTC-TFC-2016				
Normal	0.960	0.932	0.046	0.886
Anomalous	0.931	0.954	0.068	0.886

D. RQ2: Earliness Performance

The purpose of this research question is to investigate the effectiveness of our approach in detecting attacks at an early stage. To answer our research question, we conducted a replay session where we replicated the network traffic captured in the dataset and tested it against our approach to simulate a real-time environment. Our approach and the traffic replay software were run on separate machines, both featuring an Intel Core i9-10900X CPU, 64 GB of memory, RTX 3090 graphics card, and Ubuntu 20.04 Operating System. The machines were connected via a 1Gb Ethernet connection in an isolated environment to reduce network latency.

Table III shows the earliness, the minimum number of packets required (MNP) to predict a flow class accurately, and the average flow length per class per model. The results show that our approach can detect anomalous flows by inspecting roughly the first packet.

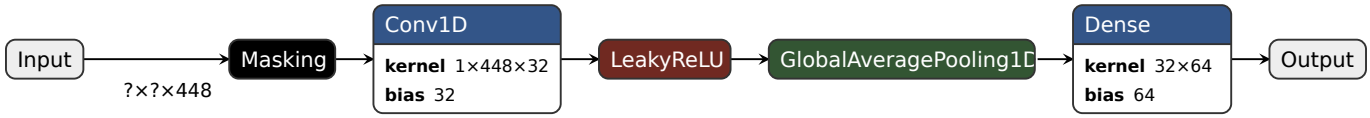

Fig. 5. Early-A model architecture

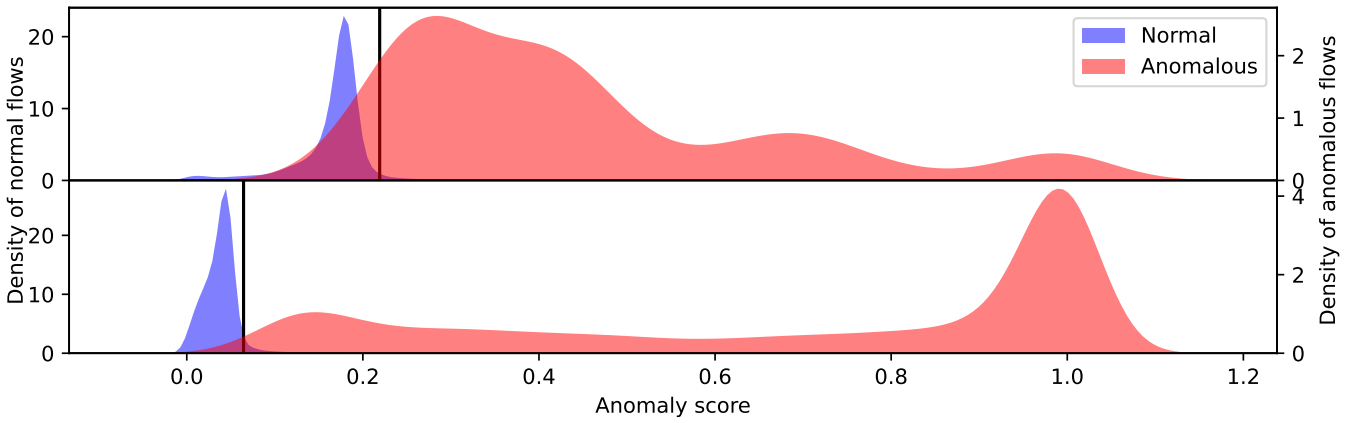

Fig. 6. Kernel density estimation plots of the anomaly scores for the CICIDS2017 dataset at the top and the USTC-TFC-2016 dataset at the bottom. The vertical black lines represent the thresholds.

TABLE III
EARLINESS METRIC AND THE AVERAGE MINIMUM NUMBER OF PACKETS REQUIRED (MNP) TO PREDICT THE FLOW CLASS

Class	Earliness	MNP	Average Flow length
CICIDS2017			
Normal	0.946	7.66	124.39
Anomalous	0.993	1.11	16.23
USTC-TFC-2016			
Normal	0.928	1.57	8.91
Anomalous	0.997	1.02	8.11

IV. Related Work

Several researchers have investigated the use of neural networks for anomaly-based intrusion detection. Out of these, only a few suggested solutions for early detection of attacks. In the following, we briefly enumerate some of the works more relevant to our approach.

In [14], the authors combine an attention mechanism with an autoencoder for detecting intrusions in the in-vehicle network. Similarly to our approach, they perform the detection at the packet level. Differently from us, they do not group the traffic into flows, instead, they convert hexadecimal traffic to binary one.

Hwang et al. [15] propose a method for detecting anomalous traffic called D-PACK. It utilizes a CNN and an unsupervised deep learning model Autoencoder to learn normal traffic patterns and filter out abnormal traffic. D-PACK examines only the initial 80 bytes of the first two packets in each flow to achieve early detection. In contrast, we do not restrict the length of flows to a fixed value; moreover, we define a metric to properly evaluate the earliness of our approach.

Lunardi et al. [16] propose an unsupervised anomaly-based IDS called ARCADE (Adversarially Regularized Convolutional Autoencoder for unsupervised network anomaly DEtection). It uses 1D-CNN based Autoencoder and Generative Adversarial Networks (GAN) to identify anomalous flows by analyzing a few initial packets of network flows. They evaluate the earliness performance of their by fixing the maximum length of flows to different values. They do not investigate how different types of flows with different lengths affect the performance of their approach. In contrast, we do not restrict the length of flows to a fixed value; moreover, we define a metric to properly evaluate the earliness of our approach. Even though our approach had around 15 times fewer parameters than ARCADE, we were able to achieve the same results.

The approach in [17] presents a combination of deep and shallow learning, targeted at efficient training and real-time detection of network attacks. The authors suggest the combination of the improved classification performance of a non-symmetric deep autoencoder with the accuracy and speed of random forests. The resulting model, evaluated on the KDD Cup '99 and NSL-KDD datasets, exhibits an average precision of 92.97% and an average reduction in training time of 97.72%. However, they do not consider early detection of attacks.

The authors of [18] propose an approach for intrusion detection that is intended to be able to generalize between related data distributions and to provide explainable results. The approach uses energy-based flow classification inferred from a statistical model. The approach was evaluated on three web-based datasets (CIDDS-001, CICIDS17, and CICDDoS19) performed network flow binary classification with an F1 score

of around 97% and an AUC of 99%. Despite the benefits of the approach, early real-time classification is not considered.

The work in [19] proposes an unsupervised hierarchical detection model in which the first level is used for feature extraction and the second one for flow classification. Similar to our approach, they target web network traffic and they also preprocess the dataset to extract relevant features from packets. The approach achieves a recognition accuracy of 99.4988%. However, the approach does not detect flows and does not attempt to detect the attacks early.

The approach described in [20] uses a Long Short Term Memory (LSTM) model combined with the attention mechanism for flow classification. They evaluate the approach using CSE-CIC-IDS2018 dataset and obtained a classification accuracy of 0.96. However, the approach does not consider the early classification of the network flows.

V. Conclusions

In this paper, we proposed an anomaly-based intrusion detection approach which is able to detect attacks in real-time while they are happening. The CNN model (with a comparatively small number of parameters) used for training and monitoring is chosen so that the approach to be fast and usable in real-time settings. The evaluation showed that it can detect reliably attacks with high accuracy, however, additional adjustments can be delegated to the network administrator if needed. Future work will evaluate the approach on additional datasets and from different application domains and will consider improving the explainability of the classifications by employing attention mechanisms.

Acknowledgments

This work was made possible with funding from the European Union's Horizon 2020 research and innovation programme, under grant agreement No. 957212 (VeriDevOps). The opinions expressed and arguments employed herein do not necessarily reflect the official views of the funding body.

References

[1] B. Mukherjee, L.T. Heberlein, and K.N. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, 1994.

[2] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150, 2021.

[3] Giovanni Vigna, William K. Robertson, and Davide Balzarotti. Testing network-based intrusion detection signatures using mutant exploits. In *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*, pages 21–30. ACM, 2004.

[4] Pedro Garcia-Teodoro, Jesús Esteban Díaz Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.*, 28(1-2):18–28, 2009.

[5] Benoit Claise, Brian Trammell, and Paul Aitken. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. *RFC*, 7011:1–76, 2013.

[6] Tanwir Ahmad, Dragos Truscan, Jüri Vain, and Ivan Porres. Early detection of network attacks using deep learning. In *15th IEEE International Conference on Software Testing, Verification and Validation Workshops ICST Workshops 2022, Valencia, Spain, April 4-13, 2022*, pages 30–39. IEEE, 2022.

[7] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 10–15 Jul 2018.

[8] Lukas Ruff, Robert A. Vandermeulen, Billy Joe Franks, Klaus-Robert Müller, and Marius Kloft. Rethinking assumptions in deep anomaly detection. *CoRR*, abs/2006.00339, 2020.

[9] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Paolo Mori, Steven Furnell, and Olivier Camp, editors, *Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Madeira - Portugal, January 22-24, 2018*, pages 108–116. SciTePress, 2018.

[10] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking, ICOIN 2017, Da Nang, Vietnam, January 11-13, 2017*, pages 712–717. IEEE, 2017.

[11] J. Sola and J. Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. 44:1464–1468, 1997.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[13] William G. Cochran. *Sampling Techniques, 3rd Edition*. John Wiley, 1977.

[14] Pengcheng Wei, Bo Wang, Xiaojun Dai, Li Li, and Fangcheng He. A novel intrusion detection model for the can bus packet of in-vehicle network based on attention mechanism and autoencoder. *Digital Communications and Networks*, 2022.

[15] Ren-Hung Hwang, Min-Chun Peng, Chien-Wei Huang, Po-Ching Lin, and Van Linh Nguyen. An unsupervised deep learning model for early network traffic anomaly detection. *IEEE Access*, 8:30387–30399, 2020.

[16] Willian T Lunardi, Martin Andreoni Lopez, and Jean-Pierre Giacalone. Arcade: Adversarially regularized convolutional autoencoder for network anomaly detection. *IEEE Transactions on Network and Service Management*, 2022.

[17] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):41–50, 2018.

[18] Camila F. T. Pontes, Manuela M. C. de Souza, Joao J. C. Gondim, Matt Bishop, and Marcelo Antonio Marotta. A new method for flow-based network intrusion detection using the inverse potts model. *IEEE Transactions on Network and Service Management*, 18(2):1125–1136, jun 2021.

[19] Yujie Zhu, Dezhi Han, and Xinming Yin. A hierarchical network intrusion detection model based on unsupervised clustering. In *Proceedings of the 13th International Conference on Management of Digital EcoSystems*, MEDES '21, page 22–29, New York, NY, USA, 2021. Association for Computing Machinery.

[20] Peng Lin, Kejiang Ye, and Cheng-Zhong Xu. Dynamic network anomaly detection system by using deep learning techniques. In Dilma Da Silva, Qingyang Wang, and Liang-Jie Zhang, editors, *Cloud Computing – CLOUD 2019*, pages 161–176, Cham, 2019. Springer International Publishing.