Unsupervised protocol-based intrusion detection for real-world networks

Maxime Labonne, Alexis Olivereau, Baptiste Polve, Djamal Zeghlache

Unsupervised Protocol-based Intrusion Detection for Real-world Networks

Maxime Labonne
Institut LIST, CEA
F-91120, Palaiseau, France
maxime.labonne@cea.fr

Alexis Olivereau
Institut LIST, CEA
F-91120, Palaiseau, France
alexis.olivereau@cea.fr

Baptise Polve´
Institut LIST, CEA
F-91120, Palaiseau, France
baptiste.polve@cea.fr

Djamal Zeghlache
Institut Te´le´com
Te´le´com SudParis
E´vry, France
djamal.zeghlache@it-sudparis.eu

***Abstract*—Anomaly-based Intrusion Detection Systems (IDSs) are rarely deployed in real networks, because of their high false positive rate. Their ability to detect unknown attacks is, however, very valuable in a context where new threats are emerging almost daily. This paper presents an unsupervised anomaly-based intrusion detection solution focused on protocol headers analysis. This approach is tested on a recent and realistic dataset (CICIDS2017) over a 4-day period. Each protocol is converted to a set of normalized numeric features, which are processed by 5 neural network architectures: deep autoencoders, deep MLPs, LSTMs, BiLSTMs, and GANs. The output of these algorithms is an anomaly score, which is normalized and combined with the anomaly scores of other protocols. We argue that this classification problem is very different from the actual problem of intrusion detection and requires new metrics. In particular, packet anomaly scores must be refined in a post-processing step to aggregate anomalies into continuous attacks. This approach successfully detects 7 out of 11 attacks not seen during the training phase, without any false positives. It is thus possible to consider deployments in real-world networks of such IDSs, capable of reliably detecting zero-day attacks.**

***Index Terms*—intrusion detection, unsupervised learning, CICIDS2017, neural networks**

I. INTRODUCTION

In recent years, hacking has become an industry unto itself, increasing the number and diversity of cyber attacks. Threats on computer networks range from malware to denial of service attacks, phishing and social engineering. An effective cyber security plan can no longer rely solely on antiviruses and firewalls to counter these threats: it must include several layers of defence. Network-based Intrusion Detection Systems (IDSs) are a complementary means of enhancing security, with the ability to monitor packets from OSI layer 2 (Data link) to layer 7 (Application).

Traditional intrusion detection techniques compare the monitored traffic to a database of known attack signatures to determine whether an intrusion is under way (e.g., Snort [1], Suricata [2]). This method effectively detects attacks that match signatures in the database, with high accuracy and low computational overhead. However, these systems can only detect known attacks. This is why anomaly-based IDSs are valuable. They create a model of the normal behavior of the network and detect any variation from this model as an anomaly. This approach can thus detect unknown attacks, but often generates an overwhelming number of false positives (i.e., flows or packets marked as attacks though they are benign).

Anomaly detection is most often based on machine learning algorithms with supervised learning. This is a problem as anomaly detection specifically learns the behavior of a given network: it is therefore not directly transferable to another computer network. The IDS must be re-trained on the network where it is deployed with a dataset containing labelled attack data. Unfortunately, it is difficult in practice to set up such a dataset, which requires a dedicated traffic generator [3]. Furthermore, such an IDS is mainly trained to detect attacks that are already known, which limits its value compared to a signature-based IDS.

On the other hand, unsupervised learning does not require any attacks in its training data. It is therefore much easier to deploy in reality and not biased by a selection of known attacks [4].

In this paper, we test 5 different architectures of neural networks trained in a unsupervised way on protocol headers. The best models are then combined to obtain the most accurate detection possible. Finally, a post-processing step refines the results and eliminates most false alarms. We argue that the metrics usually used to evaluate the quality of an IDS (TPR, FPR, F1-score...) are not appropriate to effectively measure its performance in a real situation. A new metric is proposed, focusing on the detection speed of attacks, and not on a correct classification rate. This metric is defined as the time between the beginning of the attack and its actual detection.

Network intrusion detection has been extensively studied in the past decades. Clustering is one of the most popular unsupervised method to find anomalies in a dataset. Leung and Leckie [5] show an application of this technique with a clustering algorithm on KDD Cup 99, specifically designed for intrusion detection.

More recently, autoencoders and deep autoencoders have been used for their ability to reconstruct their input. However, like other methods, they are prone to generate many false positives. Kotani and Sekiya [6] developed a flow-based method with robust autoencoders to reduce the false positive rate on a real-world traffic dataset. Mirza and Cosan [7]

This work was supported by the SCENE project (http://scene-project.eu/), which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement number 831138.

tested different RNN units for autoencoders to find the best architecture for packet payload reconstruction. This technique is complementary to ours since we do not analyze protocol payload data in this study.

The remainder of the paper is organized as follows. Section 2 introduces the CICIDS2017 dataset and the preprocessing stage. Section 3 describes 5 neural network architectures for unsupervised learning and a protocol-based ensemble learning process. Section 4 discusses the problem of machine learning metrics in intrusion detection. Section 5 presents our experimental results. Finally, section 6 concludes this paper and draws the avenues of future work.

II. DATASET

This work requires data that is representative of a real network, including both malicious and normal traffic. This dataset needs to be labelled in order to measure the performance of the IDS. We chose the CICIDS2017 dataset for its realistic network and credible attacks [8].

A. CICIDS2017

CICIDS2017 is a dataset designed for Intrusion Detection Systems (IDSs) and Intrusion Prevention Systems (IPSs) by the Canadian Institute for Cybersecurity. The goal of the authors is to propose a reliable, publicly available IDS evaluation dataset on a realistic network, with a diverse set of modern attack scenarios. It was designed to solve the problem of lack of a up-to-date and credible dataset for intrusion detection.

CICIDS2017 provides 5 days of traffic, from Monday, July 3, 2017 to Friday July 7, 2017. The first day contains only normal traffic, while the 4 next days include normal traffic and 14 types of attacks: brute-force (FTP-Patator and SSH-Patator), Denial of Service (slowloris, SlowHTTPTest, Hulk, GoldenEye), Heartbleed, web attacks (brute-force, SQL injection, XSS), infiltration, botnet, Distributed Denial of Service (ARES), and port scanning.

CICIDS2017 consists of 3 119 345 labeled network flows (83 features) and 56 329 679 network packets. This dataset is also highly imbalanced, with 83.34% normal flows against 0.00039% flows labelled as "Heartbleed" [9]. In a supervised learning task, data augmentation would be a good solution to rebalance the 15 classes of CICIDS2017. However, in our unsupervised learning task, data augmentation cannot be used: Heartbleed attacks will simply be more difficult to detect compared to more prevalent classes.

To the best of our knowledge, we found an undocumented shortcoming of CICIDS2017. The attacker's private IP address (172.16.0.1) remains the same for most attacks (Tuesday, Wednesday, Thursday morning and Friday afternoon), contrary to the authors' indications. Furthermore, this IP address is used almost exclusively for intrusions, which makes it very easy to detect. We could not identify other attacking IP addresses from the network packets. We ensured that the neural networks used in this study did not simply learn to detect the IP address 172.16.0.1. However, this feature was considered too important to be simply removed from the dataset.

B. Preprocessing

Network traffic analysis is mainly divided into flow and packet analysis. RFC 6437 defines a flow as "a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that a node desires to label as a flow" [10]. The transition from packets to flows leads to a loss of the information contained in the packets. The lost information is not necessarily useful for intrusion detection, and this compression speeds up processing compared to packet analysis.

However, network protocols are a form of artificial language, while flows are only an aggregation of them. Recent work in Natural Language Processing (NLP) has shown excellent results in language modeling with machine learning algorithms [11]. Although network protocols have many differences with natural languages, their low variance makes them easier to predict than flows. For these reasons, we chose to train machine learning algorithms to model the behavior of protocol headers.

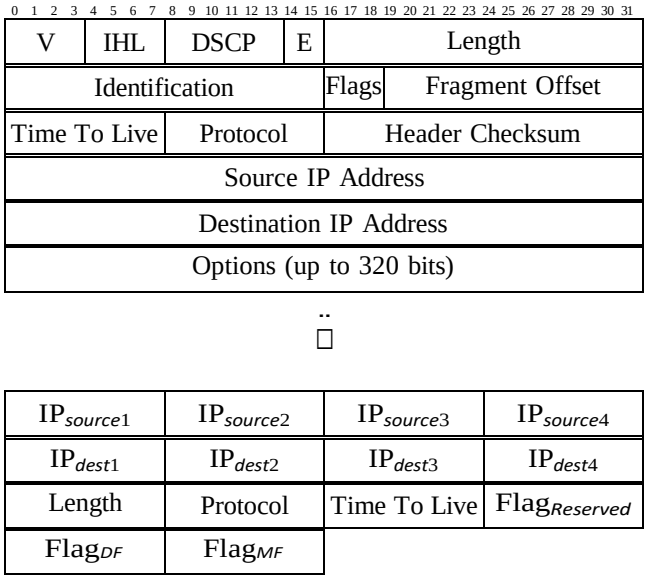

Fig. 1. IPv4 Header Feature Extraction.

In this work, we focus on the 8 protocols most represented in CICIDS2017: Frame, Ethernet, ARP, IP, TCP, UDP, DNS, and HTTP. First, these protocols headers need to be extracted from the packets contained in the dataset. Then, a set of features is selected from the variables contained in these headers. Fig. 1 shows the example of feature extraction for IPv4. Categorical features are converted into numerical features by one-hot encoding or label encoding. All these features are then normalized between 0 and 1 (min-max scaler).

Finally, each packet must be labelled as an attack or a benign packet. Note that only flows are labelled by the authors of CICIDS2017. Fortunately, they detailed their labelling process in the original paper [8] with IP addresses and timestamps. Although these details are probably not exhaustive enough to properly label each packet during an intrusion, they

are sufficient to detect continous attacks. This preprocessing framework creates 8 datasets for each day that are stored in SQL databases.

This process has been applied on Monday (11 701 690 packets – 11GB), Tuesday (11 543 109 packets – 11GB), Wednesday (13 781 563 packets – 13GB), and Thursday (9 314 199 packets – 7.8GB). Friday (9 989 118 packets – 8.3GB) has been excluded to reduce computing time and because of the difficulty of labelling numerous very short attacks.

III. Anomaly Detection Process

A. Neural network architectures

The objective of the neural networks is to learn the behavior of each protocol header. Different neural network architectures can be used to solve this problem. In this paper, 5 architectures are studied: deep autoencoders, deep MLPs, LSTMs, Bidirectional LSTMs (BiLSTMs), and Generative Adversarial Networks (GANs). Each architecture and its specific use is briefly described in the following.

Autoencoder is a feedforward neural network with a bottleneck to force the network to learn a compressed representation of the original input. In this paper, we use a wide topology with a *len*(*input*)-256-128-64-32-64-128-256-*len*(*input*) structure and ReLU as the activation function. Autoencoders learn to minimize the distance between the output $\bar{x}$ and the input x. The Mean Squared Error (MSE) is employed to evaluate performance during training, and as an anomaly score during test. This approach assumes that an attack will be more difficult to reconstruct for the network than a normal protocol header.

MLP is another feedforward neural network, with a *len*(*input*)-512-256-128-64-*len*(*input*) structure and ReLU as the activation function. This MLP receives the 50 previous protocol headers as input and tries to predict the next protocol header. This input window provides context to the network compared to the previous approach. The distances between each prediction and the actual next protocol header are used to evaluate performance during training, and as an anomaly score during test.

LSTM is a recurrent neural network that can learn long-term dependencies. Its input is similar to the previous architecture, with a window of 40 protocol headers. Its purpose is not to rebuild part of the input, but to predict the next protocol header. On the other hand, **BiLSTMs** run input data once from beginning to the end, and once from end to beginning to understand context better. Their input is comprised of the 20 previous protocol headers and the 20 next protocol headers. BiLSTMs try to predict the protocol header in the middle of this input window. Both architectures have *len*(*input*) input units, 400 LSTM or BiLSTM units, $2 \times$ *len*(*input*) ReLU units, and *len*(*input*) sigmoid units.

GAN is a deep neural network architecture composed of two networks: a generator (1000-128-256-512-*len*(*input*) with LeakyReLU and dropout) and a discriminator (*len*(*input*)-256-128-64-1 with LeakyReLU and dropout). The first net generates new data, while the other classifies each example as real (i.e., from the training set) or fake (i.e., generated). In this paper, GANs are trained on Monday with benign traffic. Only discriminators are then used as classifiers for the next days. This approach assumes that an attack will not be recognized by the discriminator as part of the training set.

B. Ensemble learning

Fig. 2 illustrates the ensemble learning process that is used to infer a packet anomaly from the analysis of multiple protocols. First, features are extracted from protocol headers as described in the previous section. These features are analyzed by each of the 5 neural network architectures to output an anomaly score. Finally, these anomaly scores are averaged to obtain an anomaly score for the entire packet.

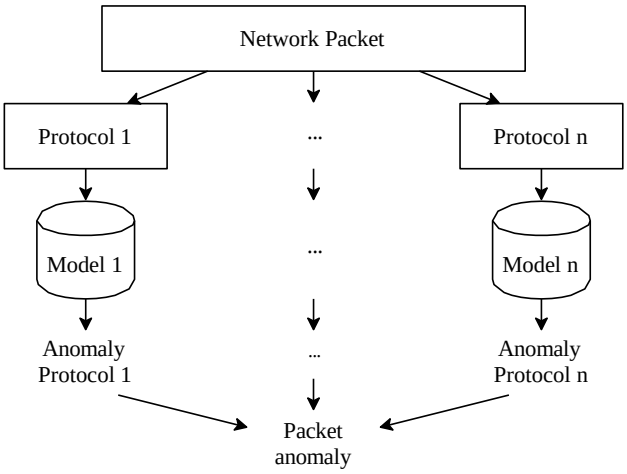

Fig. 2. Protocol-based Ensemble Learning.

The purpose of protocol-based ensemble learning is to detect attacks related to one protocol (i.e., producing a very high anomaly), and to increase confidence in the predictions when attacks are detected on multiple protocols. Moreover, it also allows to dynamically add or remove protocols according to the monitored network.

IV. The problem with metrics

Anomaly-based intrusion detection usually relies on machine learning. Different metrics are used to measure the performance of these machine learning algorithms: True Positive Rate (TPR), False Positive Rate (FPR), F1-score, the Area Under the Receiver Operating Characteristics (AUROC), etc. [12] The very choice of these metrics is problematic, as none of them can perfectly represent the quality of the algorithm's detection alone. For instance, F1-score completely ignores true negatives. However, the main problem is that these metrics do not reflect the problem that the algorithm is trying to solve.

Correctly classifying flows or packets and detecting attacks are two distinct problems. Nevertheless, the metrics inherited from machine learning are only relevant to the first one. Indeed, intrusions are most often composed of several flows or packets [13]. However, detecting all the flows or packets of an unknown attacks is unrealistic without producing a lot of

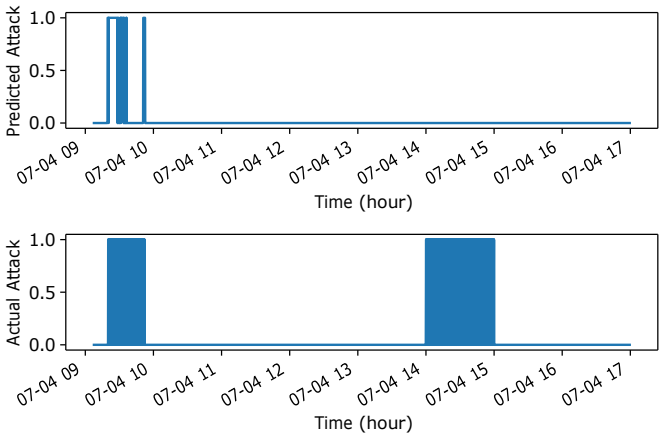

Fig. 3. Autoencoders Predictions of Attacks on Tuesday.

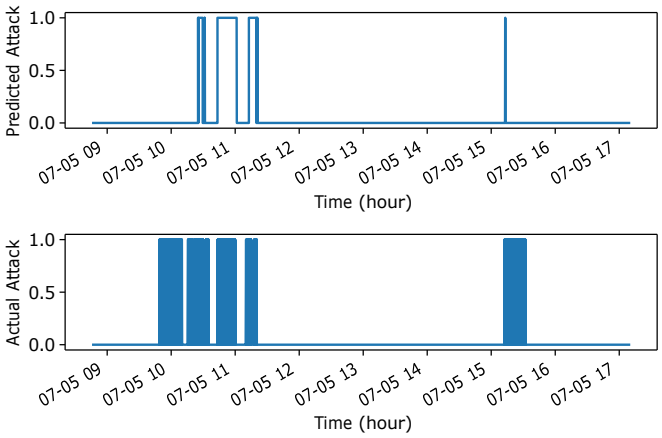

Fig. 4. BiLSTMs Predictions of Attacks on Wednesday.

false positives. Fortunately, anomaly scores can be correlated to better estimate the probability of a given sequence being an attack. Therefore, detecting a small portion of the flows or packets of an attack is enough to deduce that an intrusion is ongoing. This is why metrics in intrusion detection should measure the performance of the *attack* classification.

From the perspective of network administrators, we identified 3 expected outputs:

1) **Correct alerts**. An IDS must be able to detect attacks with a low false positive rate. TPR, FPR, F1-score, AUROC, etc. are relevant metrics to measure the performance of attack classification.
2) **Low-latency alerts**. Intrusion detection is often followed by an intrusion reaction stage. This reaction can be manual or automatic with an IPS. Reactions are all the more effective if the attack is detected early enough. This is why we propose a new metric: the time between the beginning of the attack and its detection (called Time Before Detection in the rest of this paper)
3) **Information gathering**. Collecting as much information as possible about the attack is also a very important feature for an IDS. This information can then be used to stop or to block the ongoing attack. However, this study does not cover intrusion reaction, so we will not collect information on detected attacks to illustrate this point.

V. Experiments

The proposed framework was implemented on two GTX 1080 Ti GPUs and an Intel i5-7500 CPU with 64GB of RAM. 8 models of each neural network architecture were trained on Monday's traffic on each protocol: Frame (11 701 690 instances), Ethernet (11 701 690 instances), ARP (46 971 instances), IP (11 618 823 instances), TCP (10 710 204 instances), UDP (934 997 instances), DNS (788 288 instances), and HTTP (107 681 instances).

Training time on the frame protocol depends on the architecture, but takes on average between 9 and 15 hours (with early stopping and cyclical learning rate). Autoencoders are the fastest architecture to train with 10 minutes 13 seconds per epoch on the frame protocol. BiLSTMs are the slowest architecture to train with 16 minutes 53 seconds per epoch on the same protocol.

input : Network packet
output: Packet anomaly (number between 0 and 1)

```
1  extract protocol headers from network packet;
2  foreach protocol header do
3      select list of features;
4      convert categorical features into numerical
         features;
5      normalize features between 0 and 1;
6      if protocol header has a trained neural network
         then
7          predict anomaly score;
8      else
9          train neural network;
10     end
11 end
12 average anomaly scores;
```

Algorithm 1: Packet Anomaly Prediction

Models are tested on Tuesday, Wednesday, and Thursday. Attacks can be deduced from packet anomalies, but these final scores are very noisy. A moving average with a window of 10 000 packets is applied to erase isolated anomalies and to aggregate groups of anomalies into continuous attacks. Since this detection framework is intended for real world application, the moving average is only applied on past packets. Anomalies are normalized between 0 and 1, and a threshold of 0.8 is chosen to categorize attack packets.

Results show that GANs produce too many false positives to be used. Autoencoder is the only architecture capable of detecting the FTP-Patator attack on Tuesday morning (see Fig. 3). BiLSTM outperforms other architectures for detecting Wednesday and Thursday attacks (see Fig. 4 and Fig. 5). These

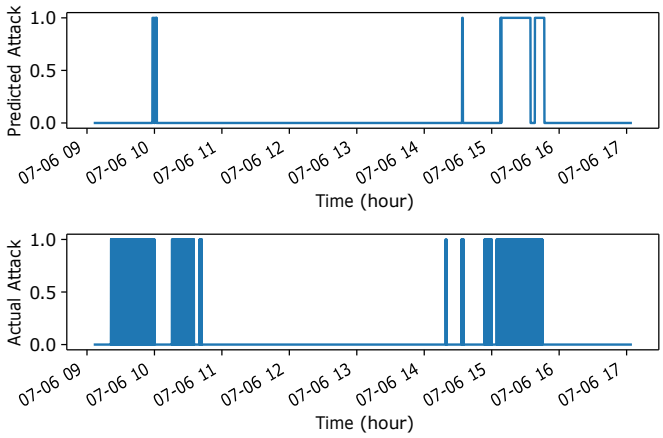

Fig. 5. BiLSTMs Predictions of Attacks on Thursday.

results are sufficiently uncorrelated to motivate a new step of ensemble learning, by averaging autoencoders and BiLSTMs packet anomalies. This final model successfully detects 7 out of 11 attacks not seen during the training phase, without any false positives. Table I summarizes the Times Before Detection for each attack.

TABLE I
TIME BEFORE DETECTION FOR CICIDS2017 ATTACKS

	Time Before Detection (s)
FTP-Patator	28.72
SSH-Patator	Not detected
DoS slowloris	Not detected
DoS Slowhttptest	550.10
DoS Hulk	23.60
DoS GoldenEye	170.07
Heartbleed	175.78
Web Brute-force	2293.03
XSS	Not detected
SQL injection	Not detected
Infiltration	874.92

The infiltration attack (Thursday afternoon) is the only multi-step attack, comprised of 3 stages: Meta exploit Win Vista (14:19, 14:20-14:21, and 14:33-14:35), Infiltration – Cool disk – MAC (14:53-15:00) and Infiltration – Dropbox download Win Vista (15:04-15:45). The intrusion is first detected between 14:33 and 14:35, and then between 15:10 and 15:31.

VI. CONCLUSIONS

Our proposed detection framework offers a high detection rate while solving the main problem of anomaly detection (high FPR). Its deployment in a real network does not require a traffic generator or transfer learning like supervised learning methods would. The protocol-based detection is highly customizable, with the ability to add or remove protocols depending on the monitored network. The main drawback of this method is a high training time (up to 14 hours for a single protocol) compared to other unsupervised techniques. Feature selection and a reduced input window for BiLSTMs could help speed up the training.

Attack recognition is also an important information for intrusion reaction. Indeed, for example, mitigation methods for DDoS attacks and targeted infiltrations are very different [14]. A multi-class classifier could analyze protocol anomalies to identify patterns specific to certain attacks. This classifier would be independent of the network and could be reused in all implementations.

ACKNOWLEDGMENT

This work was supported by the SCENE project (http://scene-project.eu/), which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement number 831138.

REFERENCES

[1] B. Caswell, J. C. Foster, R. Russell, J. Beale, and J. Posluns, *Snort 2.0 Intrusion Detection*. Syngress Publishing, 2003.

[2] W. Park and S. Ahn, "Performance comparison and detection analysis in snort and suricata environment," *Wirel. Pers. Commun.*, vol. 94, no. 2, pp. 241–252, May 2017. [Online]. Available: https://doi.org/10.1007/s11277-016-3209-9

[3] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *CoRR*, vol. abs/1903.02460, 2019. [Online]. Available: http://arxiv.org/abs/1903.02460

[4] S. Zanero and S. M. Savaresi, "Unsupervised learning techniques for an intrusion detection system," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, ser. SAC '04. New York, NY, USA: ACM, 2004, pp. 412–419. [Online]. Available: http://doi.acm.org/10.1145/967900.967988

[5] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters." 01 2005, pp. 333–342.

[6] G. Kotani and Y. Sekiya, "Unsupervised scanning behavior detection based on distribution of network traffic features using robust autoencoders," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov 2018, pp. 35–38.

[7] A. H. Mirza and S. Cosan, "Computer network intrusion detection using sequential lstm neural networks autoencoders," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, May 2018, pp. 1–4.

[8] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," 01 2018, pp. 108–116.

[9] R. Panigrahi and S. Borah, "A detailed analysis of cicids2017 dataset for designing intrusion detection systems," vol. 7, pp. 479–482, 01 2018.

[10] S. Amante, J. Rajahalme, B. E. Carpenter, and S. Jiang, "IPv6 Flow Label Specification," RFC 6437, Tech. Rep. 6437, Nov. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6437.txt

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[12] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skoric´, "Measuring intrusion detection capability: An information-theoretic approach," in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '06. New York, NY, USA: ACM, 2006, pp. 90–101. [Online]. Available: http://doi.acm.org/10.1145/1128817.1128834

[13] 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*, July 2009, pp. 1–6.

[14] N. Z. Bawany, J. A. Shamsi, and K. Salah, "Ddos attack detection and mitigation using sdn: Methods, practices, and solutions," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425–441, Feb 2017. [Online]. Available: https://doi.org/10.1007/s13369-017-2414-5