# Enhancing Network Security Through Deep Learning: Mitigating Feature Engineering and Zero-Day Attacks

Ibrahim EL Didi, Fouad Trad, Ali Chehab
Department of Electrical and Computer Engineering
American University of Beirut
Beirut 1107 2024, Lebanon

{ike07, fat10}@mail.aub.edu, chehab@aub.edu.lb

***Abstract*—The significant impact of cyber-attacks on their targets, particularly those assisted with ML algorithms, gave rise to a critical need for building a robust detection model based on raw flow traffic without involving feature extraction from the dataset. In recent years, organizations have been working hard to secure their assets from threat agents by implementing IDS and monitoring systems to mitigate these kinds of cyber-attacks, which typically rely on handcrafted input feature engineering. These techniques exhibit a limitation in different scenarios and fail to detect zero-day attacks.**

**To address this limitation, we introduce a deep learning model that inherently learn about features of the raw traffic without any type of preprocessing. We collect streams of raw flows and we train the model to detect malware attacks and to classify their types. The deep learning model extracts the underlying statistics of the network traffic, analyzes the flows without depending on any sort of handcrafted features, and detects the different types of malwares with 100% accuracy, and classify them with an accuracy of 97%. Moreover, the proposed model is able to detect zero-day attacks that it has not seen before with a high accuracy of 99.6%.**

***Keywords— Machine Learning, Raw flow, IDS, Malicious attacks, Zero-day attacks***

## I. INTRODUCTION

The challenge at hand revolves around the effectiveness of network security systems in the face of evolving cyber threats. The prevalent reliance on traditional models and handcrafted input features presents a significant limitation in protecting networks. The techniques based on handcrafted features extracted from network traffic may not be able to adapt to varying network conditions and to detect emerging types of cyberattacks, which is a major limitation for network security. Therefore, there is a pressing need for innovative solutions that can improve the adaptability and responsiveness of network security systems to protect against a broad spectrum of cyber threats [1]. This entails a focus on developing machine learning models that can uncover hidden network traffic patterns and identify malicious code, reducing the dependence on expert-crafted features. Deep learning (DL) has demonstrated powerful capabilities in taking raw data as input, without relying on any sort of feature engineering to solve a variety of problems. In this work, we follow a similar approach by using a DL model to detect zero-day attacks. The concept of representation learning [2] focuses on discovering features from raw input data. DL leverages this principle to capture features, which are crucial for tasks like detection and classification. It is worth mentioning that traditional network analysis methods for traffic classification rely heavily on handcrafted features, which can be time-consuming and requires expert knowledge, with the risk of missing important patterns not obvious to human analysts. Also, shallow models cannot handle large-scale data effectively, limiting their ability to process and analyze extensive datasets typical in network traffic. Traditional systems also face challenges in adapting to new and evolving types of cyber threats, such as zero-day attacks, due to their dependence on predefined rules and features. Moreover, the detection accuracy of these methods can be compromised, as handcrafted features may not capture the full complexity of the data, especially in diverse and dynamic network environments. These systems also tend to generalize poorly to new, unseen data, as they are usually designed for specific scenarios and may not perform well in different contexts. Finally, traditional methods may not be efficient enough for real-time analysis and detection, given their computational limitations and the extensive feature extraction processes required. As a result, DL has been used to analyze traffic data at the packet level, one by packet at a time, however, yielding relatively low accuracies in detection and classification. To overcome these limitations, we introduce a DL model for network traffic analysis that takes as input raw traffic flows to detect zero-day attacks and to classify up to ten types of botnet attacks directly from the byte streams. The model achieved 100% accuracy in detecting malicious traffic, and 97% accuracy when classifying 10 types of botnet attacks. Moreover, we trained the model on benign traffic along with all malware types except one, then tested it on detecting this specific malware as a zero-day emulation, resulting in 100% accuracy. Then, we trained the model on all 10 types of malware and we performed the testing on different malware types from a separate dataset. The model achieved 99.6% accuracy in detecting 10 types of botnet attacks.

In summary, the main contributions of this work are:

- Introducing a deep learning model for network traffic analysis that eliminates the need for handcrafted feature extraction, achieving 100% accuracy for malicious traffic detection and 97% for traffic classification.
- Demonstrating the model's effectiveness in zero-day attack detection by training on benign traffic and all malware types except one, achieving 100% accuracy in detecting the excluded malware.
- Validating the model's generalization capability in detecting zero-day attacks by achieving 99.6% accuracy

2024 4th Intelligent Cybersecurity Conference (ICSC) | 979-8-3503-5477-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICSC63108.2024.10895964

when tested on different malware types from a separate dataset.

The rest of the paper is organized as follow: Section II presents a review of the relevant literature. Section III describes the methodology used to deal with the data, and the design of the predictive models. Section IV presents the experiments conducted and the results obtained with a comparison against previous works; Section V summarizes the finding and presents future research directions.

## II. Related work

Several researchers have discussed data representation and its performance with machine learning when building classifiers using a Convolutional Neural Network (CNN) model for malware. For instance, the authors in [3] take the traffic data as an image and train the model to classify the malware type; Authors in [4] proposed a framework to build malware classifiers using CNNs by converting the input binaries into grayscale image and training the CNN model on two datasets including Malimg and Microsoft malware, achieving accuracies of 98.52% and 99.97% respectively. In [5], the authors proposed a CNN model for real-time IDS based on deep learning model that collects logs in real time and reduces the dimension using a proposed self-encoding AE-AlexNet model using the KDD 99 dataset, achieving 94.32% accuracy. In [6], the authors proposed an IDS for flow based classification using SDN OpenFlow switch that analyzes packets by extracting and aggregating features to detect malicious traffic, achieving 98% accuracy. The authors of DeepMal [7] highlighted the ability of CNN and RNN models to detect and classify raw packets of network traffic directly from the byte stream. They proposed using raw packet representation to detect benign and malicious traffic, achieving a detection accuracy of 77.6%. With the same representation, they classified the type of malicious botnet traffic, achieving an average accuracy of 72.4%, with individual accuracies for different types of malicious attacks: botnet 99%, Virut 54.7%, and Neris 63.5%. Moreover, they suggested using raw flows only for malicious traffic detection and achieved an accuracy of 98.6%. However, they did not propose any flow-based mechanism for traffic classification. As such, in this work, we use the same flow-architecture of DeepMal, but we adapt it for the classification of flows, rather than classifying packet by packet. Then, we test this approach in identifying zero-day attacks, which up to our knowledge, no previous work has addressed when working on raw traffic data. In summary, the approach we propose will have three main advantages compared to previous works:

- Using raw flow network traffic. Hence, no reliance on feature engineering.
- Better classification performance for a large number of attacks.
- The ability to detect zero-day attacks.

In [8], the authors proposed a framework for Internet of Things (IoT) devices to identify the malicious traffic by extracting the features from network flows to identify the source, and to detect the traffic type. However, this approach involves feature extraction, which we aim to eliminate in this work.

## III. Methodology

The model used in our research relies on one dimensional CNNs, which are utilized for analyzing and uncovering the patterns of byte stream data. Such architectures are usually based on four pillars: convolutional layers, Max pooling layers, a flatten layer, and fully connected layers.

The convolutional layer, upon receiving a raw vector of integers as input, processes the information using a filter of specified window size. This filter convolves over the raw vector to gather weights, which are then forwarded to the next layer, which identifies the features of the input raw vector. The output values of the convolutional layer are scaled properly after passing through a non-linear activation function, which produces the output channels.

The max-pooling layer is used to take the maximum byte values from the selected filter while convolving over the raw vector. This is similar to a filter that convolves over an image to select the highest value; this step is crucial for two reasons 1) to reduce the size of the raw vector obtained from each flow, and 2) to compress the extracted features. However, in our case, to avoid dataset limitations and to preserve the values of the extracted bytes for studying the underlying statistics of the flow, we chose to omit the max pooling layer, which has proven to be an effective strategy.

The flatten layer is used to flatten the values coming from the convolutional layer into a one-dimensional array, which is then connected to dense layers.

Finally, the dense layers take the flattened vector as input, and process the values until reaching the output layer. The number of neurons in the output layer depends on the task at hand. For detection tasks, the output layer consists of a single neuron with a sigmoid activation function. For classification tasks, the output layer has neurons equal to the number of classes, each with a softmax activation function.

### A. Proposed CNN Model and Flow Representation

The proposed CNN model consists of a 1D-CNN layer with 16 filters, each having a kernel size of 5, and two fully connected layers with 50 and 100 neurons, respectively, as shown in Figure 1. A flow represents a collection of packets. In our case, we have two datasets: one containing benign and malicious flows in PCAP files [9], and the other [10] consisting of a single file that combines benign and malicious traffic. For both datasets, we applied some processing steps (highlighted in the next section) to aggregate the packets into flows. Then, for each flow, we select the first two packets as suggested in a previous study [7].

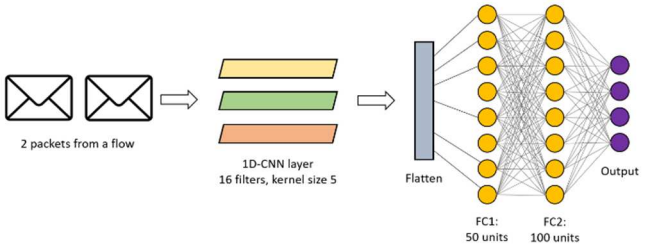


Figure 1: CNN architecture for Raw Flow Representation

### B. DataSet Processing

We use two main datasets in this study. Both of them are in the form of PCAP files. The goal of processing them is to transform them into flows. The first dataset [9] consists of large PCAP files containing 10 types of malware traffic and 10 types of benign traffic. The second dataset [10] consists of one PCAP file that includes the benign and malicious traffic at the same time. However, the dataset documentation reveals that malicious traffic was targeted

toward specific IP addresses. Accordingly, we split the data between malicious and benign using the command:

*SplitCap -r "input file path" -o "output file path" – ip,*

To avoid applying the same command several times, each time for one IP, we create a list of the malicious IP addresses in a file, and add it to the command to gather the whole malicious traffic in one PCAP file. Afterwards, we remove the malicious IPs from the PCAP file to isolate benign traffic. Since we aim to conduct a statistical analysis of the data to uncover patterns that differentiate between benign and malicious traffic, we decided to treat the raw flows as "sessions." Hence, we split every PCAP file into sessions using the "SplitCap" tool with the following command:

*SplitCap -r "input file path" -o "output file path" -s session*

This approach allows us to analyze each session individually, providing a more detailed and accurate understanding of the traffic patterns. By splitting a large PCAP file into smaller ones, each containing the packets of a single flow [11], we ensure that each session holds bi-directional flows. This preserves the sequential interactions between hosts, which are obtained from the upper layers of the OSI model. Analyzing the behaviors of higher-level protocols maintains a complete view of stateful interactions between hosts. It is crucial to retain the values of the flow contents to train the model with sufficient information to detect and classify different types of cyber-attacks. This method enables a comprehensive analysis and enhances the model's ability to identify and categorize malicious activities accurately.

## IV. Experiments and Results

The evaluation of the proposed model was based on two publicly available datasets [9], [10]. The architecture we used was based on 1D CNN, similar to DeepMAL [7], however, to achieve a higher accuracy in detection and in classification, the data was processed differently. Then, to validate further the model, we performed testing on a different dataset. Finally, we checked the extent to which the proposed approach can detect zero-day attacks.

### *A. Proposed approach vs. DeepMAL*

We started with the first dataset [9] to evaluate the model performance for malware traffic detection, that is for binary classification. The results were compared with those achieved by DeepMAL [7]. The dataset consists of two labeled groups (PCAP files) from benign and malware traffic, where the malware traffic includes 10 types, and the benign traffic is also for 10 types. We take every flow after spliting the PCAP files into sessions, then normalize the decimal representation for every byte. Next, we select the first two packets from each flow, which results in 15,000 samples. We used this data to train and validate the model to detect malicious traffic. We split the dataset as 80% for training, 10% for validation and 10% for testing. The training of the proposed architecture took place for 10 epochs, using the ADAM optimizer that dynamically adapts the learning rate. The model achieved 100% accuracy in malware detection; while DeepMAL achieved 98.6% accuracy with row flows. This shows that the proposed approach has a better potential in detecting attacks.

### *B. Proposed CNN for detecting Zero-day attacks*

In this experiment, we test the efficiency of the proposed approach in detecting zero-day attacks, which are attacks that exploit unknown vulnerabilities. To that end, we used the same dataset of the previous experiment containing 15,000 samples, equally divided between malicious and benign traffic. To emulate a zero-day attack scenario, we excluded one class of malicious traffic during the training phase, and then tested the model against this excluded class. This method allows us to simulate the conditions of a zero-day attack, where the system encounters a previously unseen type of malware. We repeated this process iteratively for each malware type in the dataset. The dataset included four main classes of malicious traffic. For each iteration, we train the model on three of the classes and use the fourth class solely for testing.

Finally, we train the model on the 4 types of malwares of the first dataset [9], and we tested on the traffic of the second dataset [10]. The results are summarized in Table 1. As we can see, when we take the malicious traffic for testing from the same distribution of the training dataset, the model achieved perfect performance with 100% accuracy.

However, when we test on malware traffic from a different dataset, the performance drops slightly, achieving 99.93%. This is expected, because the dataset comes from a different distribution. But this high performance reflects the capability of the model to detect zero-day attacks.

The training process when training the model to detect malicious traffic on dataset 1 and evaluating it on the malicious traffic from dataset 2 is shown in Figure 2. If we carefully monitor the curves, it is clear that with more training, the model is able to better detect zero-day attacks, since the test accuracy is increasing after every epoch.

Table 1. Results of detecting zero-day attacks

| Training | Testing | Accuracy |
|---|---|---|
| Neris_Rbot_Virut | Zeus | 100% |
| Neris_Rbot_Zeus | Virut | 100% |
| Neris_Virut_Zeus | Rbot | 100% |
| Rbot_Virut_Zeus | Neris | 100% |
| Neris_Rbot_Virut_Zeus | Malware from the second dataset | 99.93% |

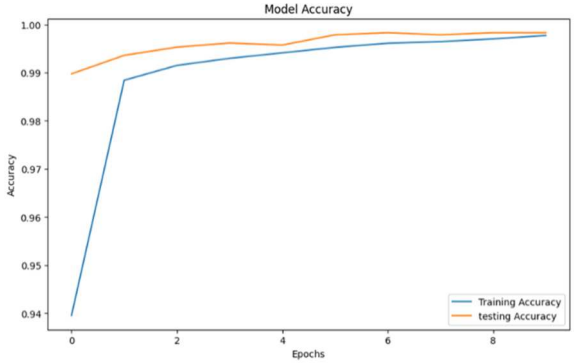


Figure 2: Zero-day attack detection

### *C. Proposed CNN Model for Malware Classification*

Using the same approach presented earlier, we test the model ability to classify the botnet types of malicious attacks, using different malware classes. For the first dataset, we apply the proposed CNN model to classify four types of malware

classes corresponding to different types of botnet attacks, namely, Virut, Neris, Zeus, and Rbot, as reported in Table 2.

For this dataset, malware traffic was captured according to different protocols (P2P, HTTP, IRC) and the malicious activities included spam, phishing, DDOS, and port scan. The processed dataset was built by selecting the flows derived from the large PCAP files as explained in the methodology section. We used the same proposed CNN model for malware detection, however, we include 5 neurons in the final layer to classify among normal traffic and each of the four attacks, with a softmax activation function instead of a sigmoid for multi-class classification. Additional metrics were measured when evaluating the performance of the model: accuracy, precision, recall, and F1 score for each class, as shown in Table 3, along with the confusion matrix shown in Figure 3.

Table 2. Different kinds of botnet attacks and their protocols

| label encoded | Botnet name | Protocol Type | Malicious Activity |
|---|---|---|---|
| 0 | Neris | IRC | spam |
| 1 | Rbot | IRC | DDOS |
| 2 | Virut | HTTP | port scan |
| 3 | Zeus | P2P (C&C) | phishing |
| 4 | Normal | | |

We can see that the proposed approach achieves excellent performance in classifying all malware classes, with an average accuracy of 97% and an average F1-score of 97.8%. To further demonstrate the effectiveness of the proposed approach, we compared its performance against the outcomes achieved by DeepMal, as shown in Figures 4, 5, 6, and 7. While DeepMal slightly outperforms our approach for the Rbot class, its performance for the other classes is much lower, suggesting that their model might be biased towards that specific class. In contrast, the proposed approach shows higher performance for all other classes and higher overall performance.

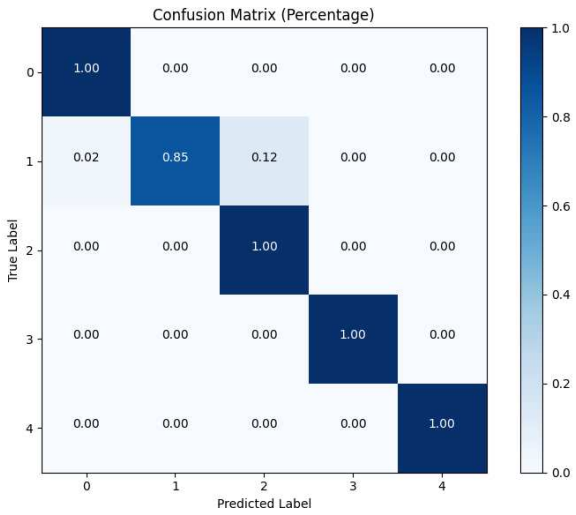


Figure 3. Confusion matrix for the classification

Table 3. Classification metrics for each class

| Class label | Precision | Recall | F1 Score |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 1 | 0.85 | 0.92 |
| 2 | 1 | 1 | 1 |
| 3 | 0.94 | 1 | 0.97 |
| 4 | 1 | 1 | 1 |

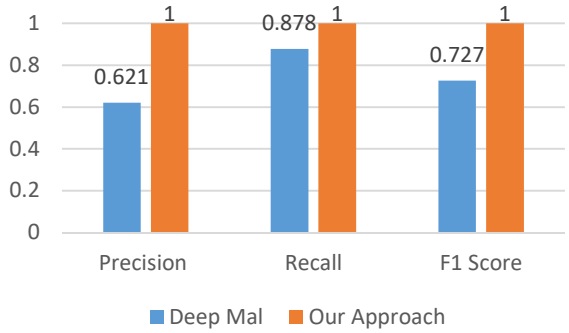


Figure 4 Comparison for the normal class

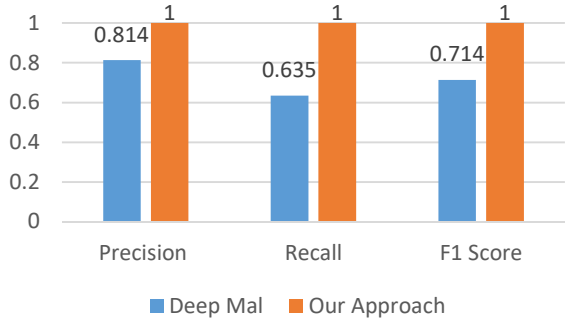


Figure 5 Comparison for the Neris class

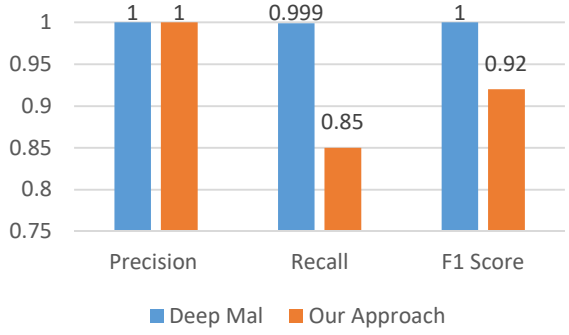


Figure 6 Comparison for the RBot class

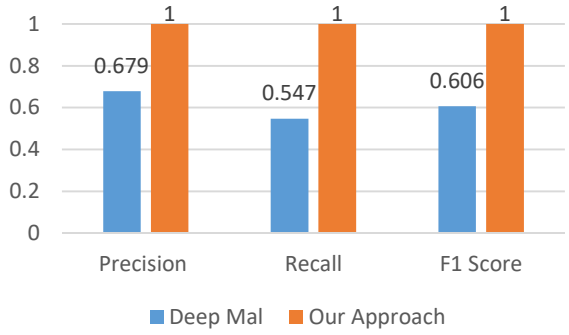


Figure 7 Comparison for the Virut class

### D. Classification for ten types of Botnet attack

In this experiment, we repeat the process on a larger dataset containing 10 botnet attacks. We follow the same steps to derive raw flows, and the objective goal is to classify 11 classes, one class representing normal traffic, and 10 malware classes, as shown in Table 3.
The flow dataset consists of 10,000 samples, balanced with 1,000 instances per class.

Table 4. The different classes in the new dataset

| label | Botnet name | Protocol Type | Malicious Activity |
|---|---|---|---|
| 0 | Neris | IRC | spam |
| 1 | Virut | IRC | DDOS |
| 2 | Normal | HTTP | port scan |
| 3 | Cridex | P2P (C&C) | phishing |
| 4 | Geodo | HTTP | ransomware |
| 5 | Htbot | TLS, TLS v1.2 | click fraud |
| 6 | Miuref | HTTP | drive by downloads |
| 7 | Nsis-ay | HTTP | phishing emails |
| 8 | Shifu | DNS | exploit kit |
| 9 | Tinba | DNS | phishing emails |
| 10 | Zeus | TCP | drive by downloads |

Applying the proposed model on this dataset, it achieved an accuracy of 99.96% for the malware classification of the 11 categories. The confusion matrix (Figure 8) shows that the model is able to accurately identify all attacks and distinguish them from normal traffic. These results prove that the proposed approach works well for multiple datasets, highlighting its effectiveness to accurately detect and classify traffic attacks.

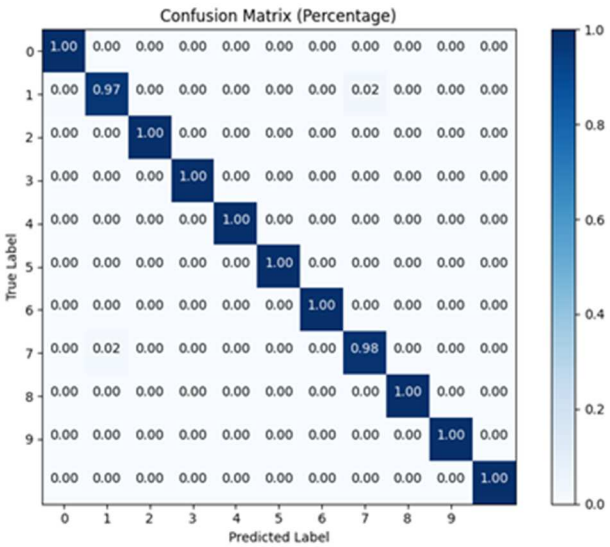


Figure 8. Confusion matrix for the new dataset of 11 classes

## V. Conclusion

In summary, this paper proposed a mechanism to analyze network traffic without relying on traditional feature engineering. The proposed approach proved to be effective in detecting and classifying multiple types of botnet attacks, using two well-known datasets. The results of these tests confirmed that the model is effective, achieving an accuracy rate of 99.96% when it comes to identifying malicious attacks directly from raw byte streams.
Moreover, the proposed framework showed a significant potential in detecting zero-day attacks, even when taking such attacks from a different dataset than the one used for training. These outcomes prove that the proposed strategy helps in achieving better classification performance while mitigating traditional feature engineering limitations and zero-day attacks.

Future work could expand in several promising directions. First, the scalability of the model could be tested under more varied network conditions and across larger datasets to further validate its robustness and efficiency. Second, integrating this model with real-time traffic analysis systems could enhance network security frameworks by providing immediate detection and response capabilities. Further research might also explore the adaptation of this model to other types of cybersecurity challenges, such as ransomware or advanced persistent threats (APTs), thereby broadening its applicability and impact in the field of cyber defense.

## References

[1] Negoita, O., & Carabas, M. (2020). Enhanced security using Elasticsearch and machine learning. *Advances in Intelligent Systems and Computing*, 244–254. https://doi.org/10.1007/978-3-030-52243-8_19.

[2] Bengio, Y., et al. Representation Learning: A Review and New Perspectives. IEEE Trans. P.A.M.I., 2013.

[3] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye and Yiqiang Sheng, "Malware traffic classification using convolutional neural network for representation learning," 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 2017, pp. 712-717.

[4] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang and F. Iqbal, "Malware Classification with Deep Convolutional Neural Networks," 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 2018, pp. 1-5, doi: 10.1109/NTMS.2018.8328749.

[5] Özgür A, Erdem H. 2016. A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. PeerJ Preprints4: e1954v1.

[6] G. A. Ajaeiya, N. Adalian, I. H. Elhajj, A. Kayssi and A. Chehab, "Flow-based Intrusion Detection System for SDN," *2017 IEEE Symposium on Computers and Communications (ISCC)*, Heraklion, Greece, 2017, pp. 787-793, doi: 10.1109/ISCC.2017.8024623.

[7] Marín, G., Caasas, P., & Capdehourat, G. (2021). Deepmal - deep learning models for malware traffic detection and classification. *Data Science – Analytics and Applications*, 105–112. https://doi.org/10.1007/978-3-658-32182-6_16

[8] Salman, O., Elhajj, I. H., Chehab, A., & Kayssi, A. (2019). A machine learning based framework for IOT device identification and abnormal traffic detection. *Transactions on Emerging Telecommunications Technologies*, *33*(3). https://doi.org/10.1002/ett.3743

[9] Yungshenglu. (n.d.). *Yungshenglu/USTC-TK2016: Toolkit for processing PCAP file and transform into image of Mnist Dataset*. GitHub. https://github.com/yungshenglu/USTC-TK2016

[10] *ISCX botnet dataset 2014*. University of New Brunswick est.1785. (n.d.). https://www.unb.ca/cic/datasets/botnet.html

[11] Hjelmvik, E. (2022, October 27). *What is a PCAP file?* Netresec. https://www.netresec.com/?page=Blog&month=2022-10&post=What-is-a-PCAP-file