Payload–Based Network Traffic Analysis for Application Classification and Intrusion Detection

Süleyman Özdel*, Çağatay Ateş*, Pelin Damla Ateş*, Mutlu Koca and Emin Anarım
Department of Electrical and Electronics Engineering, Boğaziçi University, Istanbul, Turkey
{suleyman.ozdel, cagatay.ates, pelin.ates, mutlu.koca, anarim} @boun.edu.tr
*These authors contributed equally to this work.

***Abstract*—Network traffic characterization has become an important topic with the development of evasion techniques. Preventing malicious activities is vital in terms of network performance. On the other hand, defining which applications are run through the network traffic has become a significant issue with the diversification of applications. In this work, a payload-based flow analysis tool that provides both application classification and intrusion detection is proposed. Payload features that characterize network flows efficiently are used to classify network traffic and detect malicious attacks. Application classification performance analysis is performed on a publicly available up-to-date dataset containing traces from most popular applications such as Spotify, WhatsApp, etc. Attack detection performance is evaluated on IDS 2012 and IDS 2017 datasets containing different kinds of attack traces.**

***Index Terms*—network traffic, classification, intrusion detection, payload**

I. Introduction

One of the main topics of computer science is traffic classification in these days. Traffic classification allows internet service providers and network operators to evaluate overall network performance. In this aspect, network traffic classification is defined as an automated process and can be characterized into various traffic classes by looking at certain parameters. Some of those parameters are port numbers and protocol information, which are prime examples for categorizing network traffic. These parameters are frequently encountered in network-related topics. In any system in network operations, anomaly detection is crucial in order to determine the states of the process that do not go as it should. Thus, it is ensured that the aforementioned system remains stable and works with the expected efficiency in this direction. One of the most common areas of implementation of anomaly detection methods is network intrusion detection. In a widespread manner, attacks in network security are DDoS attacks which are also known as Distributed Denial of Service attacks.

The leading studies in this field focus on distinguishing compressed and encrypted network traffic by using traffic classification methods are given in [1]. For instance, social media applications consist of encrypted traffic networks that are detected by using machine learning algorithms given in [2]. This research, which identifies social media subtypes, concentrates on the challenges that may arise as well as machine learning-based alternate solutions to potential bottlenecks. In [3], a systematic approach based on an example of traffic classification that is the optimized feature selections in encrypted traffic classification is offered. The other research which is related to traffic classification relies on the collection, expansion, and selection of flow parameters given in [4].

On the other hand, the detections of DDoS and DoS (Denial of Service) attacks, which are frequently encountered in the field of the Internet of Things (IoT), have been achieved with high performance by using the Residual Network (ResNet) algorithm with the support of the Convolutional Neural Network algorithm (CNN) given in [5]. In addition to this, the Decision Tree algorithm and Support Vector Machine (SVM) algorithm are prevailing examples of machine learning methods used in intrusion detection systems given in [6].

In this paper, the network traffic classification and intrusion detection methods are proposed. This method relies on features extracted from the payload of packets. Although the payload is encrypted, the proposed features model the network efficiently. These same features classify different applications used in intrusion detection processes. To verify the robustness of the proposed method, different publicly available datasets are used.

This paper is organized as follows. The details of the proposed attack detection algorithm are described in Section II. Then, the test results and analysis of the algorithm are given in Section III. Lastly, Section IV evaluates the results, and future studies are mentioned.

II. Network Traffic Characterization

The structure of the proposed system is explained as follows. In the first stage, packets belonging to the same flow on incoming network traffic are gathered. In this process, it is taken into account that five-tuple packet header information named source IP address, destination IP address, source port, destination port, and protocol information is the same. In this way, flow-based analysis is provided instead of packet-based analysis. Flow-based features are constructed when the number of packets in the flow reaches the predetermined minimum number of packets parameter. While creating these features, the payload portion of packets is used.

The payload represents the actual data carried on the packet. Characterization of network flows cannot be performed efficiently by using traditional intrusion detection systems (IDS) in some cases because, in general, these systems focus on

using packet header information. In order to characterize each flow, payload or content analysis is required at the application level. Payload–based feature extraction is performed in this algorithm. Even though the payload portion of packets is encrypted, the proposed algorithm highlights the features of different types of traffic content. It does not look for keywords in packets. Therefore, it does not have to decrypt the messages.

A. N–Gram Payload Feature Extraction

N–gram analysis was first introduced by Damashek in 1995 to analyze writing independent of language [7]. Its common use is for analyzing language characteristics in natural language processing (NLP). Here, N represents the number of array elements to be analyzed. These elements can be characters or words involved in language processing. N can take any value other than zero in the set of natural numbers. Arrays are created using the sliding window principle over the data to be analyzed. The most common use of this method is examining the characteristics of the payload portion of packet payloads in network security.

For payload–based analysis, in the general use of N–gram method, the payload is processed with the help of a sliding window of size N. This determines the number of occurrences of each N–gram sequence. Let D be a dictionary of length m containing all possible sequences of N–grams. D can be expressed as,

$$D = \{d_1, d_2, ...d_i, ...d_m\} \tag{1}$$

where d_i, $i \in \{1, 2, ...m\}$ represents a unique sequence of N–gram. If the frequency of use of this array is expressed by O_i, then comparable frequency of it can be expressed as,

$$f_i = \frac{O_i}{\sum_{k=1}^{m} O_k} \tag{2}$$

where f_i is the comparable frequency of i^{th} array. Comparable frequency values correspond to probability values. In this way, the probability distribution of the payload of a packet is constructed. The probability distribution of a payload can be expressed as

$$P_k = \{f_1, f_2, ...f_i, ...f_m\} \tag{3}$$

where P_k represents the probability distribution of payload of the k^{th} packet.

The values in the payload of a packet vary between 0 and 255. For example, if N is taken as one (unigram), if each value is observed in the payload, the length of the probability distribution of the payload becomes 256. Therefore, in the unigram case, the maximum length of the probability distribution is found to be 256. Similarly, if n is taken as two (bigram), the maximum number of unique sequences is calculated as 256^2. Therefore, if n increases, the complexity of the procedure increases as the size of probability distributions increases.

1) Entropy – Based Features: Probability distributions of packet payloads generated by the N–gram method are used to derive different features. The metric that measures the randomness of a probability distribution is called entropy. Probability distributions with low randomness have a low entropy, while probability distributions with high randomness have high entropy. This property is used to distinguish between attack and non–attack traffic. The entropy of a probability distribution of packet payload can be calculated as

$$H(P_k) = -\sum_{i=1}^{m} f_i \log(f_i) \tag{4}$$

where $H(.)$ represents the entropy operator. By finding the probability distributions of the payloads of packets in the flow, their entropy values are calculated. If a flow contains p number of packets, p number of entropy values are obtained. These values can be called the entropy vector. The minimum, maximum, average and standard deviation values of this entropy vector are used as features in attack detection procedure. These features are called H_{min}, H_{max}, H_{mean} and H_{std}.

2) Ratio of Printable Characters in Payload: The printable character ratio attribute is frequently encountered in encrypted packet detection algorithms. Thanks to this method, network packets are effectively characterized. DoS and DDoS attack utilities generate their own packets to evade detection mechanisms. While various packet payloads are created by the attacker, it is aimed to capture the bandwidth and resources of the target. This attribute, which is used to categorize any of the patterns caused by the attack traffic, plays a very critical role. ASCII characters, usually 0 to 255, are used to encrypt the payload properly. Only bytes between 32 and 127 correspond to printable characters and are used to compose text messages. For randomly generated payloads, the printable character ratio corresponds to approximately 37.5%. Higher rates are used to retain written data based on transferring characters.

The printable character ratio attribute is based on a simple basis. This basis is counting only sequences that are fully printable characters. If n equals 1, the 1–gram sequence contains only 1 byte and fully printable characters. Each string has more than 1 byte when n equals to any value other than or greater than one. Any n–gram sequence can be expressed as

$$S_i = \{b_1, b_2, ..., b_n\} \tag{5}$$

where $b_k, \quad \in \{1, ..., n\}$ denotes the ASCII equivalent of byte value of k^{th} value of the payload. In a similar manner, by defining O_i as the number of occurrences of unique sequence S_i in the n–gram analysis, and defining the probability value as,

$$\rho_i = \begin{cases} 1, & \text{if} \quad 32 < b_k < 127 \quad \text{for all} \quad b_k \in S_i \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

the total number of occurrences of printable characters called printable character sequences is calculated as

$$O_{pcs_i} = \sum_{i=1}^{m} p_i \times O_i \tag{7}$$

where all values in the dictionary are added by multiplying with the corresponding probability value. After defining these values, the ratio of the total number of occurrences of printable characters in the n–gram sequence to the number of occurrences in the original sequences in the whole n–gram analysis is given as

$$R_{pcs_i} = \frac{O_{bko_i}}{\sum_{i=1}^{m} O_i} \tag{8}$$

where denominator is the summation of all values in the dictionary. This value is calculated for each packet in the flow. In this way, if a flow contains N number of packets, then there will be N number of printable character ratio values. Then, the features are created by using the minimum, maximum, mean and standard deviation values of them. These features are called r_{min}, r_{max}, r_{mean} ve r_{std}.

B. Greedy–Algorithm Based Features

It is provided to derive new features with different metrics other than entropy using the probability distributions constructed for the payload in the packets. This section discusses the features generated using the Greedy algorithm. This algorithm assigns an upper value to the distance between probability distributions that differ in size. If the probability distributions were the same size, the distance could be calculated using the Kullback–Leibler (KL) metric. However, since the amount of bytes used in the payload portion of each packet is not constant, probability distributions are not expected to always be equal in length. For this reason, the Greedy algorithm is used to calculate the distance between probability distributions with different sizes. Details of the Greedy algorithm are given in [8].

Basically, the Greedy algorithm takes two probability distributions of payloads such that

$$G_{k,l} = d_{greedy}(P_k, P_l) \tag{9}$$

and calculates the upper bound for the distance between them. In (9), the distance between the k^{th} and l^{th} probability distributions is calculated and named as $G_{k,l}$.

The distance calculation process between the probability distributions of packet payloads is applied to all packet pairs in the flow. For example, if there are p packets in the flow, then $\binom{p}{2}$ distance values will be obtained. For the i^{th} flow, the Greedy distance values vector can be constructed as

$$\mathbf{G}_i = [G_{1,2}, G_{1,3}, ..., G_{1,p}, G_{2,3}, ..., G_{p-1,p}]. \tag{10}$$

By taking the minimum, maximum, mean and standard deviation of this vector, new features are created. These features are called dg_{min}, dg_{max}, dg_{mean} ve dg_{std}.

C. Classification Method

The bag of decision trees method is selected as a classifier for this proposed system. In this method, more than one decision tree is trained for various subsets of data, and the final decision is made by averaging the results of these decision trees. Compared to other popular machine learning algorithms such as Support Vector Machines (SVM), k–Nearest Neighbors, Naive Bayes etc., the bag of decision trees gives a better and more robust performance. In the proposed system, 50 different decision trees are constructed, and their decisions are used for the final decision.

III. Analysis and Discussion

The proposed network traffic classification algorithm has been tested on three different datasets. One of them is the application classification dataset which contains up-to-date traffic samples from popular applications such as Whatsapp, Zoom, Facebook etc. It is a publicly available dataset, and details of it can be found in [9]. The other two datasets are intrusion detection datasets containing different types of attacks such as DDoS, DoS, Port Scan etc. These datasets, IDS 2012 [10], and IDS 2017 [11], are widely used in the network anomaly detection research community.

A. Application Classification

The algorithm has been tested on a publicly available dataset mentioned above. The dataset contains the flows of 22 applications collected at different times. Applications and the number of flows that belong to each application are given in Table I. There are many popular applications seen in the dataset. Also, some similar applications such as Telegram and Whatsapp make classification harder.

TABLE I
Application List

Amazon Prime	2001	**CyberGhost**	474
Dropbox	510	**Deezer**	760
Discord	646	**Epic Games**	1650
Facebook	477	**Hotspot**	395
ITunes	787	**Microsoft Teams**	559
Proton VPN	557	**Skype**	996
Slack	1364	**Soulseekqt**	2249
Spotify	2129	**Steam**	547
Telegram	374	**Tunnelbear**	15076
Tunneln	3110	**Ultrasurf**	12279
Whatsapp	442	**Zoom**	703

Before the simulation, two parameters must be determined. The first of these is N, that is, the minimum number of packets in the flow. When a network flow reaches N packets, its features are calculated and entered into the classification algorithm. While higher N values increase the algorithm's complexity and time for detection, it improves the correctness of the prediction. Small N values are required for the early detection and real-time processes. It can be considered as a trade between the detection time and accuracy of detection. The second parameter is m, that is, how many bytes will be taken from the payload portion of each packet. If this value

is small, the flow characteristic may not be expressed clearly. On the other hand, a high byte value increases the algorithm's complexity.

The algorithm has been tested for different values of N and m. The performance analysis were performed by dividing the dataset into 70% training and 30% tests. F1-score is used as the performance metric in the evaluation of proposed method. The F1-score is the harmonic mean of the precision and sensitivity values. It is generally used in datasets having unbalanced class distribution. Besides that, in the case in which a balance between precision and sensitivity is required in the evaluation of the performance of the classification algorithm, F1-Score becomes a better option F1-score values of test results are given in Table II. It is seen that the performance of the algorithm increases when the number of packets and the payload byte of each packet increases. Best performance is obtained in the 10–packet and 2000–byte case. Also, 10–packet and 500–byte case has relatively good performance compared to other cases. This shows that increasing the number of packets within a flow has more impact than increasing the byte number of payloads.

TABLE II
SIMULATION RESULTS - F1 SCORE

	3 Packets	**5 Packets**	**10 Packets**
100 Byte	0.7094	0.8170	0.8629
200 Byte	0.7829	0.8895	0.9279
300 Byte	0.7894	0.9177	0.9394
400 Byte	0.7910	0.9186	0.9478
500 Byte	0.8116	0.9343	0.9481
1000 Byte	0.8829	0.9554	0.9573
2000 Byte	0.9367	0.9604	0.9720

In order to get a more detailed analysis, the confusion matrix of the 5-packet 500-byte case is given in Figure 2. Diagonal values on this matrix represent correctly detected samples. The vertical axis represents the actual class, and the horizontal axis represents the prediction class resulting from the algorithm. Therefore, based on a row, values in non-application indices are incorrect estimates. Also, the values on the matrix represent the number of samples in the data. In the confusion matrix, the highest number of errors occurs in Spotify samples predicted as Slack. Flows belonging to Tunneln, a radio app, are correctly predicted. Besides that, Hotspot, Telegram, and Facebook flows are predicted with a very low number of errors.

B. Anomaly Detection

Simulation of anomaly detection part has been carried out using the datasets aforementioned before. 70% of the samples are taken as training, and 30% of the samples are taken as test data. Results are shown with the different number of packets within flows. Also, the other parameter, which is the number of bytes in a packet payload, is determined using multiple simulations and searching for the best accuracy value.

F1-Score was used as a performance criterion in evaluating the simulation results.

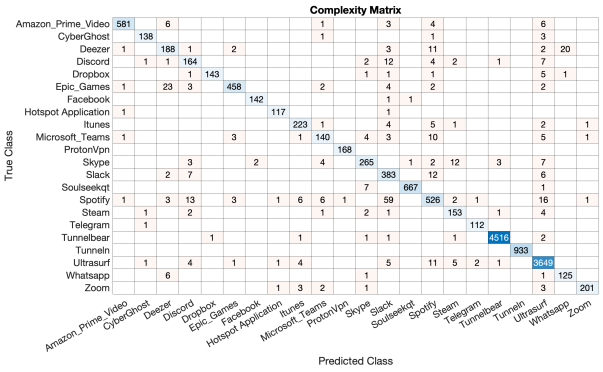

Fig. 1. Application Classification Complexity Matrix

1) IDS 2012 Dataset: To use this dataset, malicious traces from Tuesday and Thursday are taken with the normal activities. 3–class classification is performed as anomaly detection. Simulation results based on F1–score are given in Table III. The number of packets within a flow is taken as a parameter in the simulations. The number of bytes in a packet is taken as 400. It is observed that detection performance is slightly better when the number of packets within a flow increases. On the other hand, detection is performed with relatively good accuracy with a low number of packets. This shows that the proposed algorithm detects malicious activities with high accuracy even with a small number of packets. This is a favorable property in intrusion detection systems.

TABLE III
SIMULATION RESULTS - IDS 2012

	3 Packets	**5 Packets**	**10 Packets**
Normal	0.9995	0.9996	0.9995
DDoS	0.9569	0.9605	0.9789
Brute Force - SSH	0.9884	0.9936	0.9998

2) IDS 2017 Dataset: In this dataset, different attack traces like DoS Hulk, Port Scan, DDoS etc. are evaluated. In total, 9–class classification model, including normal samples, is constructed. In this model, there are different types of DoS attacks like GoldenEye, Hulk, Slowhttptest, and Slowloris. Simulation results are shown in Table IV. In these simulations, 400–byte payload is taken, and the number of packets within a flow is taken as a parameter. As in the previous case, it is observed that a small number of packets can reach a high-performance rate. This is very important in terms of minimizing the damage of attacks.

Among different types of attacks, it is seen that best performance is obtained with the DoS Slowloris case. Apart from DoS and DDoS attacks, the proposed system detects Port Scan and Brute Force attacks too. This shows that the proposed algorithm is suitable for different kinds of attacks. Among these attacks, the worst performance is obtained in the DDoS case with about 87% F1–score. This shows that payload–based features extracted in the proposed system can model the DoS and Brute Force attack better than DDoS and Port Scan attack cases.

TABLE IV
SIMULATION RESULTS - IDS 2017

	3 Packets	5 Packets	10 Packets
Normal	0.9831	0.9833	0.9835
DDoS	0.8703	0.8702	0.8707
DoS GoldenEye	0.9368	0.9468	0.9509
DoS Hulk	0.9897	0.9903	0.9907
DoS Slowhttptest	0.9478	0.9636	0.9816
DoS Slowloris	0.9958	0.9941	0.9925
PortScan	0.9234	0.9230	0.9235
Brute Force - SSH - Patator	0.9927	0.9833	0.9905
Brute Force - FTP - Patator	0.9858	0.9855	0.9858

The complexity matrix of the model trained for the IDS 2017 dataset is given in Figure 2. On the matrix, the vertical axis represents the real classes, and the horizontal axis represents the classes predicted as a result of the proposed system. Numbers on the diagonal of the matrix correspond to the correctly detected number of samples, and other elements are the number of incorrect estimates. As an example, 112 DoS Hulk instances are labeled as Normal. This number occupies very little space among all of the DoS Hulk samples. In this scenario, whose complexity matrix is shown below, the first 5 packets of the flows that carry the payload are used to obtain the features. Then, using the features calculated for different n–gram values, a classification model containing 50 different decision trees were constructed with the bag of trees method. In this classification model, there are a total of 9 classes, 8 of which are attack classes, and the other one is Normal. In the complexity matrix, it is seen that most errors occur whether an attack sample is labeled as normal or vice versa. Mislabeling between different attack types is observed very little such as 3 DoS Hulk samples are labeled as DoS GoldenEye. This shows that different types of attacks are separated among themselves with high accuracy. The main obstacle is deciding whether a sample is anomalous or normal. In the proposed algorithm, from the simulation results, it can be said that it has good performance in differentiating attack traffic and normal traffic.

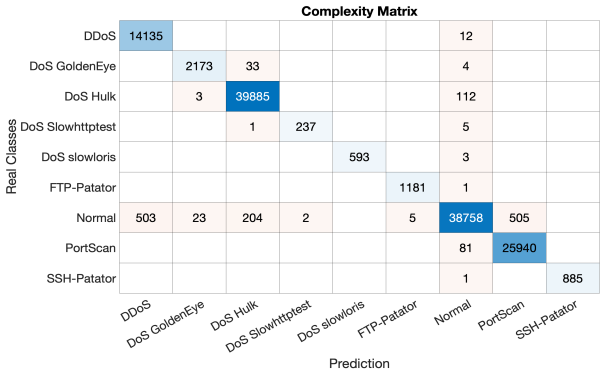

Fig. 2. IDS 2017 Complexity Matrix

IV. CONCLUSION AND FUTURE WORK

In this article, traffic classification and intrusion detection algorithm are proposed. The proposed system can be seen as accomplishing two tasks which are network traffic classification and intrusion detection. In the first one, network flows are classified according to the applications, while in the second one, these flows are classified according to whether they have malicious traffic or not. The network traffic classification algorithm is simulated with up–to–date publicly available traffic traces containing samples from popular applications such as Facebook, Whatsapp, Spotify etc. The results show that the proposed algorithm achieves a satisfying classification rate. On the other hand, the intrusion detection algorithm is simulated with well-used malicious traffic datasets, which are IDS 2012 and IDS 2017. The results show that the proposed system detects anomalous traffic even with the first few packets within flows.

In order to reduce the false alarm rates of the proposed system, the cluster of payload–based features will be enriched by developing new type of features. Also, the proposed system will be tested on new applications and malicious traffic scenarios.

ACKNOWLEDGEMENT

This work is supported by Boğaziçi University Research Fund Grant Number 18281.

REFERENCES

[1] F. Casino, K.-K. R. Choo, and C. Patsakis, "Hedge: efficient traffic classification of encrypted and compressed packets," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 11, pp. 2916–2926, 2019.

[2] F. Al-Obaidy, S. Momtahen, M. F. Hossain, and F. Mohammadi, "Encrypted traffic classification based ml for identifying different social media applications," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pp. 1–5, IEEE, 2019.

[3] M. Shen, Y. Liu, L. Zhu, K. Xu, X. Du, and N. Guizani, "Optimizing feature selection for efficient encrypted traffic classification: A systematic approach," *IEEE Network*, vol. 34, no. 4, pp. 20–27, 2020.

[4] A. S. Da Silva, C. C. Machado, R. V. Bisol, L. Z. Granville, and A. Schaeffer-Filho, "Identification and selection of flow features for accurate traffic classification in sdn," in *2015 IEEE 14th International Symposium on Network Computing and Applications*, pp. 134–141, IEEE, 2015.

[5] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "Iot dos and ddos attack detection using resnet," in *2020 IEEE 23rd International Multitopic Conference (INMIC)*, pp. 1–6, IEEE, 2020.

[6] V. Deepa, K. M. Sudar, and P. Deepalakshmi, "Detection of ddos attack on sdn control plane using hybrid machine learning techniques," in *2018 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 299–303, IEEE, 2018.

[7] M. Damashek, "Gauging similarity with n-grams: Language-independent categorization of text," *Science*, vol. 267, no. 5199, pp. 843–848, 1995.

[8] Ç. Ateş, S. Özdel, and E. Anarım, "A new network anomaly detection method based on header information using greedy algorithm," in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 657–662, IEEE, 2019.

[9] M. Karayaka, A. Bayer, S. Balkı, M. Koca, and E. Anarım, "Application Based Network Traffic Dataset." "https://www.kaggle.com/applicationdataset/applicationbasednetworktrafficdataset/", 2022. [Available: 20-February-2022].

[10] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[11] 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.