Computer Science and Information Systems 21(1):37–56 https://doi.org/10.2298/CSIS211116064J

PI-BODE: Programmable Intraflow-based IoT Botnet Detection system

Đorđe D. Jovanović and Pavle V. Vuletić

School of Electrical Engineering, Bulevar kralja Aleksandra 73,
11000 Belgrade, Serbia
jd185001p@student.etf.bg.ac.rs
pavle.vuletic@ etf.bg.ac.rs

Abstract. In this paper, we propose a Programmable Intraflow-based IoT Botnet Detection (PI-BODE) system. PI-BODE is based on the detection of the Command and Control (C&C) communication between infected devices and the botmaster. This approach allows detecting malicious communication before any attacks occur. Unlike the majority of existing work, this detection method is based on the analysis of the traffic intraflow statistical parameters. Such an analysis makes the method more scalable and less hardware demanding in operation, while having a higher or equal level of detection accuracy compared to the packet capture based tools and methods. PI-BODE system leverages programmable network elements and Software Defined Networks (SDN) to extract intraflow features from flow time series in real time, while the flows are active. This procedure was verified on two datasets, whose data were gathered during the time span of more than two years: one captured by the authors of the paper and the other, IoT23.

Keywords: Botnet detection, Machine learning, IoT malware, programmable networks.

1. Introduction

Botnet is a network of computers (bots) that are under the control of a malicious hacker - botmaster. Botmasters use the devices under their control for various types of malicious activity, such as: performing Distributed Denial of Service (DDoS) attacks, spreading ransomware, stealing personal information, unwanted digital currency mining, and other [1]. Botnets came into the spotlight with the devastating attacks of the Mirai botnet which at one moment consisted of more than 600.000 devices. Although the peak of the first Mirai infection was in 2016, malware based on the Mirai code still exists and is active in creating new botnet infrastructures. For example, there has been an emergence of new botnets built using a variant of Mirai with the addition of recent exploits in the networking equipment [2]. Furthermore, new botnet generating malware appears every day, often reusing the code of the previous malware, making small changes (e.g., changing IP addresses or some strings in the messages exchanged) in order to avoid the detection or even mixing the features of multiple malware families. The recently discovered Dark Nexus botnet malware which is built on top of Mirai and QBot code [3] is one such case. For many recent botnet malware samples it is difficult

to tell which family they belong to, and multiple tags in relevant malware databases like URLhaus [32] are assigned to them.

In the botnet infrastructure, the botmaster communicates with the infected computers via the command and control (C&C) channel. One use of the C&C channel by the botmaster is to maintain the list of active bots. Botmaster either periodically polls the bots or requires periodic messages from bots using so-called C&C heartbeat packets. Another key use of C&C is to initiate the attack or to send other commands to the bots. By exploring C&C dynamic behavior patterns and creating a system that can efficiently discover botnet communication, it is possible to stop the bots before they become activated by the botmaster and involved in an attack. Since the C&C channel is a single point of failure for the botnet and its detection and mitigation fully disables the control exerted upon the bots, C&C channels evolved in time, and their detection avoidance techniques (e.g. Domain Name System (DNS) fluxing or Domain Generation Algorithms (DGA)[4]) became more sophisticated. However, our preliminary investigation [5] of the recorded samples of IoT malware did not show sophisticated detection avoidance techniques among the IoT malware samples. It revealed similar C&C behavior among multiple malware families due to the previously mentioned code reuse. C&C heartbeat communication differed in some aspects (e.g., strings in packets, the number of packets exchanged in bursts, IP flags, heartbeat initiators), but on the other side kept some features with relatively stable and similar values (e.g., low and constant bit rates, flow symmetry, periodicity and so on) which can be used for botnet detection.

Botnet or botnet-based attack detection are nowadays based on traffic statistics analysis. In this type of analysis there is a trade-off between the richness of data on one hand and network capacity, storage and processing capabilities on the other. Flow-based detection methods (that use e.g. IPFIX or similar flow gathering mechanisms) [6] store only the digest information about each network flow upon its completion (duration, number of packets and bytes). However, these methods have several drawbacks. First, full information about the flow is recorded after each flow ends, meaning that it can provide information about the attack or malicious behavior post factum. For long network flows, such as C&C heartbeats, communications last for hours or even days before the bot is activated for the attack. Such flows are either cut into multiple separate flows at the active flow timeout, which destroys the original flow properties, or stored too late to be detected during the C&C operation. Second, the global flow statistics are too coarse-grained and do not provide enough information about the intraflow dynamics, preventing the detection of some malicious activity. The alternative to this method is the analysis based on full packet capture, which provides an insight into all the packets on a specific link. The drawback to this is large overhead in processing of such data. One day of traffic recorded on a 100Gbps link amounts to approximately 1 petabyte of data that needs to be stored and processed. Furthermore, with most of the internet traffic being encrypted nowadays, stored packet data cannot provide any information apart from the size of the payload, rendering this approach highly inefficient. Therefore, in this paper we propose a PI-BODE system that lies in between the two aforementioned approaches. The system, based on Software Defined Networking principles, gathers flow statistics, enriched with the set of intraflow statistical parameters, while the flows are still active. Botnet detection is based on detecting C&C communication patterns through the analysis of intraflow statistical features, which to the best of our knowledge were not used before.

The solution was designed and evaluated using the analysis of real malware captures which were gathered in the Laboratory for information security at the School of Electrical Engineering in the period between June 2019 and October 2020 (ETF-IoTB dataset), and the IoT23 dataset captured in the period 2018-2020. The experimental results show that the PI-BODE is capable of achieving the same or higher level of detection accuracy as in the similar systems that use full traffic analysis, while it requires up to two orders of magnitude less storage space.

The structure of the paper is as follows. Section 2 provides an overview of the existing research literature. Section 3 discusses the datasets that are often used in botnet detection research, their properties and presents in detail the datasets that were used in this research. Section 4 describes the methodology of gathering and choosing the intraflow statistical features analysis. It also describes the proposed PI-BODE, a Software Defined Networking (SDN)-based intraflow statistics capturing system. Section 5 gives an overview of the data science pipeline used in this research. It describes each classifier used, feature selection and estimator parameter tuning steps. In Section 6, we presented the scores of the PI-BODE machine learning for both datasets. Obtained results are compared to the results from the related research that uses C&C communication detection. Finally, Section 7 summarizes the paper.

2. Related Work

Devastating botnet attacks on the computing and communication infrastructure provoked an increased research interest in their detection and mitigation in the last decade. Two most common botnet-related research topics are: 1. detecting the attacks and attack patterns (most often DDoS), which presents the larger part of the research work, and 2. detecting the botnet infrastructure and bot behavior. Since DDoS attacks are most often characterized by a high number of connections from various IP addresses or a high total volume of traffic, the former is done by finding anomalies in the numbers of connections and total traffic volume from the packet or flow captures [6]. However, such an approach provides information about the attack after the damage was made. For our research on C&C communication detection, more relevant is the second group of related research where we focus on the most recent papers. A thorough overview of these, mainly machine-learning based approaches is given in [7]. The key differences among the previous studies were the set of traffic features that were used to detect botnet behavior, the sources of information about the traffic (packet or flow trace), the type of botnet analyzed and the dataset that was used and machine learning methods.

One of the first C&C-related studies [8] focused on Internet Relay Chat (IRC) based C&C communication by using 16 features extracted from the full packet header traces. These features included regular statistics that can be obtained through network flow analysis, as well as packet size histogram and variances of the bytes and packet inter-arrival times for each flow. On the other side, a more recent study [9] provided an enriched traffic model and a dataset with 29 recorded and 14 generated traffic features for specific groups of flows and 3 category fields. Meanwhile, some authors explored traffic properties which cannot be found in the previously mentioned datasets, such as the periodicity in botnet communication for HTTP-based C&C communication [10, 11]. Wang et al. [12] created a BotMark system, which combines graph-based and statistical

features of traffic and hybrid analysis using similarity, stability and anomaly scores. This method uses the aforementioned scores as input for the ensemble method to classify network traffic. One of the particularly interesting features of the BotMark system is the detection method, which relies on the observed flow similarity (multiple bots have similar communication patterns with the botmaster) as the basis for the botnet communication detection.

Cusack et al., created a ransomware detection system based on machine learning and SDN-based feature extraction [13]. The network feature set in [13] includes features such as the interarrival times, the ratio between the inflow and outflow packet counts and burst lengths, which have specific values in the analyzed ransomware communication. Another paper that proposed the usage of SDN for detecting threats on the Internet of Medical Things is that of Liaqat et al. [14]. Authors used SDN to collect packet data and make forwarding decisions, which can be leveraged to stop botnet spreading. The detection mechanism uses Convolutional Neural Networks and Cuda Deep Neural Network Long Short-Term Memory (LSTM). Paper by Bilge et al. [15] focuses on the detection of botnet C&C communication using NetFlow statistics as parameters for the machine learning algorithm. Authors noticed some characteristic patterns of the C&C communication in the set of malware samples they analyzed (e.g., periodic generation of new flows, flow sizes, client access patterns, and inter-flow temporal behavior) and used inter-flow statistics to get a richer set of features that is used for the C&C channel detection. The innovative idea presented in this paper is observing flow sizes between two endpoints over a significant period of time, with the supposition that the sizes of the flows in a botnet communication will not change significantly as time passes. The main difference between [15] and our paper is that in [15] authors calculated statistics based on completed flows between two endpoints at a given time, while our paper adds fine-grain statistics of each individual flow. Blaise et al. in [16] propose an anomaly detection technique that spots the changes in the usage of a single port to identify botnets. The method is oriented towards the initial phases of the infection - TCP scans, and is adapted to detect even slow and stealthy changes. Koroniotis et al. worked on the detection of various attacks towards the IoT infrastructure including probing attacks (scanning and fingerprinting), DoS and information theft [9]. The paper by De la Torre Parra et al. [17] provides an overview of IoT based botnets and their detection methods. Authors used LSTM neural networks for the attack detection. The attack set in that paper included various types of traffic floods, but also scanning as a preparatory attack phase and sending spam data. Kurniabudi et al. in [18] analyzed feature selection using information gain as a criterion and created several classification machine learning models. It was shown that the model prediction depends on the number of features selected, and that the optimal number of those features differs from model to model. One significant branch of work was directed towards DNS-based botnet evasion techniques detection, such as DGA and DNS-fluxing [4,19-22]. Since the analyzed IoT malware samples did not use such mechanisms in this paper, DNS-based evasion techniques were not considered.

In contrast to previously mentioned approaches, all based on full packet capture analysis, PI-BODE uses flow information enriched with an original set of intraflow features, which creates a new type of dataset for the C&C channel detection, different from the datasets mentioned in this section. The produced dataset is significantly smaller, allowing cheaper storage and processing capabilities while providing at least the same level of detection accuracy. Flow statistics are extracted leveraging the

capabilities of the programmable network elements while the flows are still active, which enables online botnet detection. Similarly to the other research, PI-BODE uses machine learning techniques for detecting botnet C&C heartbeat communication channels.

3. Malware Samples and Datasets in Recent Botnet Research

One of the critical decisions in malware and intrusion detection research is the choice of the malware samples and dataset which are used for algorithm training and evaluation. A recent study [23] emphasized some issues with the datasets that are often used for the research. Some of those issues are dataset age or conditions under which the traffic was recorded, which become irrelevant due to the changes in the attack patterns over the years. Such an example is the NSL-KDD dataset, which was recorded more than twenty years ago, yet it is still being used in some recent research studies [24-26]. The use of this dataset can be justified for research on the volumetric type of (D)DoS attacks, which do not change a lot in nature over time. However, other botnet features, like other information security threats, tend to constantly change their pattern of behavior to avoid detection.

Some features, such as scanning, malware proliferation mechanisms or C&C communication continuously change in time [27]. It is a well-known fact that Mirai malware had 24 versions only in the first two months since it appeared [28], and its derivatives still appear daily. For such changing features, the use of outdated datasets gives results of limited usability. Besides, the age of the datasets, Al-Hadrami et al. [23] highlighted the problems with labeling these datasets. For instance, the information about the conditions under which they have been recorded. However, we would like to emphasize another issue with recent datasets that are commonly being used for this type of research. Datasets are recorded over a limited time presenting a snapshot of the threat landscape at that brief moment. Table 1 summarizes some of the recently used datasets for botnet detection.

Table 1. Dates when some of the commonly used datasets were recorded

Dataset	Used in	Year of recording	Recording duration
AWID	[26]	2015	9 days
ISOT	[8]	2017	7 days
CIC-IDS	[18][24][26]	2017	5 days
BotMark	[13]	2016	16 days
Bot-IoT	[8][10]	2018	6 days in a one-month period
NIMS	[29]	2014	1 day

To avoid models overfitted to one specific dataset and threat, we argue that the dataset should be created over a longer period and continuously refreshed. Also, the

research should focus on finding the features which rarely change, while the verification of machine learning models must be done on different, most recent datasets, captured over longer periods. Therefore, we decided to create our own dataset (ETF-IoTB), based on recorded malware samples over a period of more than one year, while performing verification with the IoT23 dataset [30]. The dataset details are given in the remainder of this section.

3.1. ETF-IoTB Dataset

The ETF-IoTB dataset was obtained by recording the traffic of the devices infected with the malware samples [31]. Malware samples were downloaded from the links in the URLHaus [32] database of the recent malware distribution points. Malware was executed on RaspberryPi 3 devices with Raspbian OS. RaspberryPi devices were connected to the internet without any protocol filtering, apart from protecting the local network infrastructure from malware lateral movement. The created dataset fulfills several recommendations of what constitutes a correct botnet dataset [33]: includes real botnet communication and not simulation, has unknown/regular traffic, has ground-truth labels for training and evaluating the methods and includes different types of botnet malware. Benign traffic was traffic from a regular workstation during the day.

The infected devices were constantly monitored in order to detect the situations when there is a sharp increase of the traffic volume as an indication that the infected machine is a part of the DDoS attack. The devices typically worked in the pre-attack phase, when their initial and C&C heartbeat communication were recorded, and in few cases were stopped as soon as the DoS attacks from the infected device were noticed, as described in the provided dataset. Malware dynamic behavior samples were downloaded in the period between June 15th 2019 and October 15th 2020. Two IoT malware families which are the most common today were analyzed: Mirai and Gafgyt. The dataset consists of sixteen Mirai, eighteen Gafgyt and one NanoCore sample. More details about the dataset and noticed patterns of C&C communication, as well as initiating attack commands are given in our previous work [5].

3.2. IoT23 Dataset

The IoT23 dataset is the latest dataset provided by the Technical University in Prague. Dataset consists of labeled benign and botnet traffic collected on IoT devices. The IoT23 dataset contains 23 scenarios, 20 of which contain malware traffic, while the remaining 3 contain benign traffic. Traffic samples in this dataset were captured in the period between 2018 and 2020. Malware was also recorded on RaspberryPi devices, while the benign traffic scenarios were recorded on various IoT devices: a Philips HUE smart LED lamp, an Amazon Echo home intelligent personal assistant and a Somfy smart door lock. The IoT23 dataset consists of 7 Mirai, 1 Muhstik, 1 Kenjiro, 2 Torii, 1 IRCBot, and 1 Gafgyt sample. The datasets are labeled and contain 8 labels related to the C&C communication.

4. Intraflow Statistical Feature Analysis

In this section we present PI-BODE principles of operation and gathering network intraflow statistical features.

4.1. Capturing Intraflow Information

Intraflow statistical parameters can be obtained from the packet capture (pcap) files using a software tool similar to Joy [34], which is capable of extracting some intraflow features (e.g., per-flow packet size probability distribution, entropy or Walsh-Hadamard transform).

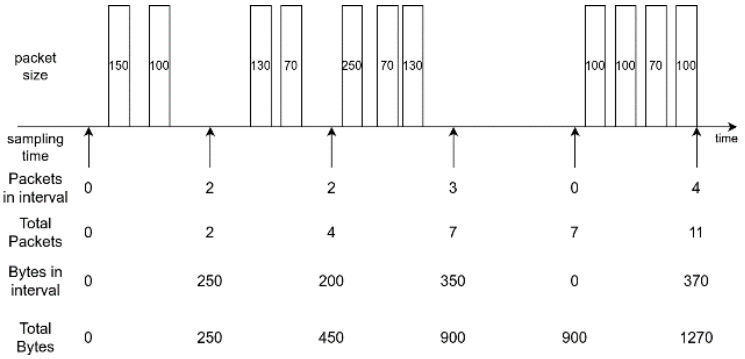

Fig. 1. Sampling flow statistics - A visualization on how packet size in bytes and number of packets is collected from a time series

Unlike this approach, PI-BODE leverages the SDN OpenFlow features for real-time statistics capturing. Instead of capturing each packet, our system samples the flow statistics periodically and gathers the flow metrics at each sample time. Figure 1 shows how flow data is being captured from the SDN switch. By sending the OFPFlowStatsRequest message periodically, the controller requests from the switch the total number of packets and the total amount of data for each flow. By sampling these counters and calculating the difference between the results of the two consecutive samples, per-flow time series for packet and byte counts are created. They are then used as a basis for various statistical features calculations, described in Section 4.3.

Two configuration parameters are: sampling period and flow idle timeout value. For the first parameter we used 1 second period as a rule of thumb, which showed reliable detection results and low network overhead. The choice of the second parameter is important for those flows, which have either long idle periods or do not have explicit end of flow signalization (e.g., UDP flows). Longer idle timeout means more overhead because expired flows will remain longer in the switch tables, but also simpler processing at the analysis station. Processing is simpler because there is no need to concatenate the parts of the flows with long idle periods if they are broken in multiple flows. We used 2 minutes for the idle timeout, as the longest periods in the C&C heartbeat communication we observed were 60s, thus ensuring non-interruptible C&C

flow recording. These parameters also define the key limitations of the PI-BODE detection system. Worse detection results can be expected if the C&C communication patterns change in a way which would make them hardly noticeable with the chosen set of configuration parameters (sampling period and idle timeout). Botnet heartbeats with a period shorter than sampling period or longer than the idle timeout would be difficult to detect. However, in case of such changes, the system can be easily reconfigured to adapt to the changes.

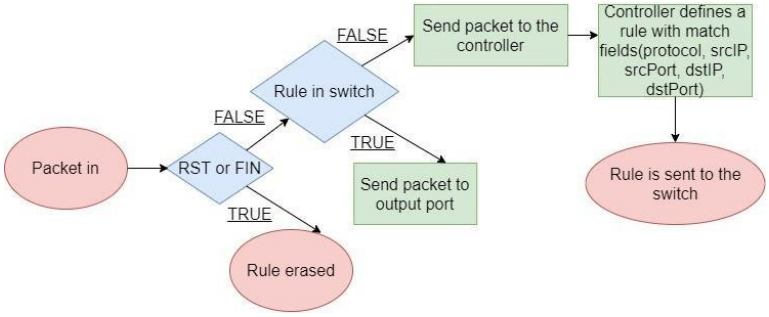

Fig. 2. Flow statistics gathering algorithm - Algorithm used by the SDN controller to perform flow switching

The simple algorithm for the TCP flow statistics gathering is given in Figure 2. For each TCP flow, the key events (first packet and terminating handshake) cause flow rules to be added or deleted in the switch. The SDN switch works as a Layer-2 transparent bridge where packets are being forwarded per flow. Each new flow, defined as a tuple (Source IP, Destination IP, Source Port, Destination Port, Transport protocol) creates a new entry in the controller's flow statistics time series database. Each flow has an OFPFF_SEND_FLOW_REM flag set to force the switch to send the summary information to the controller upon the flow removal. UDP or TCP flows with idle periods longer than the idle timeout are removed from the flow time series database upon the expiration of the idle timeout values.

Intraflow data series provide a more compact dataset and a much smaller network overhead. The exact gain in the amount of transferred traffic from the network element needed to do the attack detection is difficult to estimate, as it depends on various parameters in a given network traffic sample like: flow length distribution, number of packets per flow, flow ending methodology and others. We will illustrate the difference using one example which is taken from our packet capture [31]. One of our packet capture files with the non-malware traffic contained a total of 1,474,536 packets, which created 5,171 network flows of a total size of 2.496 gigabytes. The duration of the packet capture was 12,641s with 116 packets or 196 kilobytes per second on average transferred from the network element towards the analysis station. That same traffic when we used flow parameters sampling rate of 1s created the average traffic from the network element towards the controller in the range between 685 and 6,979 bytes per second for varying idle flow timeout values between 5s and 600s respectively that is a reduction in storage and network overhead between 28 and 280 times.

4.2. SDN Based Botnet Detection System based on intraflow statistics

Since most of the IoT devices do not possess sufficient hardware capacities, software on them is often simple and prone to attacks. Adding additional security measures on the IoT devices is not always possible, meaning security measures must be implemented on the network level. The optimal solution is to place the attack detecting device in the line of the packets that flow between the protected devices and the internet and act as both intrusion detection and prevention systems (IDS and IPS) (Figure 3).

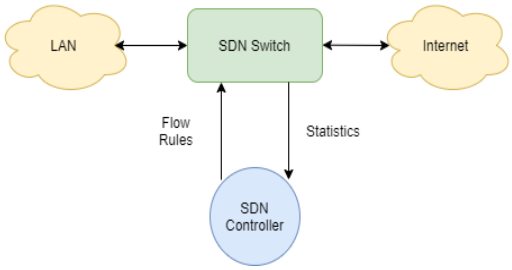

Fig. 3. IDS system using SDN - A schematic diagram of a system that protects the LAN from botnet attacks

The PI-BODE SDN controller software consists of three components (depicted in Figure 4.): Flow engine, Data collection thread and Analysis thread. The flow engine enables packet forwarding through the network. Each network flow has its own entry in the switch's flow table, but in terms of packet rewriting the switch does the basic Layer-2 forwarding. The data collection thread is used to collect packet and byte counts for all flows each second, and the collected data are then put into their appropriate time series as described in Section 4.1.

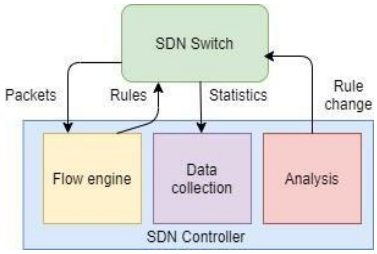

Fig. 4. SDN controller's internal structure - Principal components of the SDN controller, as well as its interactions with the switch

For all flows we also captured flow duration, total number of packets and number of entries in the empirical distribution. These parameters are not deemed flow features that are used in the detection process but are used to calculate the other features. Furthermore, flow duration is also the criterion that qualifies the flow to be used in the detection process. All flows shorter than 60s are not used for the botnet C&C detection. Some bots within the ETF-IoTB dataset (for example, xs.arm7 and nuclear.arm7) that

use periodic C&C pings and each ping represents a different flow (uses different port numbers for each ping). In these cases, defining a flow as double (src_IP, dst_IP) can capture the C&C communication and make flow tables more compact. However, in the remainder of the document we worked with tuples as defined in Section 4.1.

4.3. Intraflow Features

As described in Section 4.1, for each flow, packet and byte count samples are put into two time series, with a 1s time bin. From those time series we extracted 22 statistical features. Table 2 lists 14 features that will be analyzed in the remainder of this paper, their brief descriptions and Area Under Curve (AUC) values. The detailed discussion about the use of AUC values is given in Section 5.2.

Features 1-11 are extracted from the two time series. Features 12-14 are extracted from the empirical distribution of the flow quiet times. The flow quiet time is the number of contiguous time bins whose byte count value is 0. Empirical distribution of the flow quiet times consists of the number of occurrences of different flow quiet times in a bytes flow time series. The reason for creating such a distribution is that the observed botnet C&C behavior consists of periodic exchanges between the C&C and bot for most of the time. Therefore, by capturing intervals of flow inactivity and analyzing their distribution, it is possible to detect this periodic behavior. The explanation of the flow collection process was given in Section 4.2.

5. C&C Detection Methodology

To detect IoT based C&C communication, a regular machine learning method is applied and further optimized using a data science pipeline which comprises the following steps: data extraction and handling, feature selection, hyperparameter tuning and classification. One step that is often used as a part of the data science pipeline is data transformation, namely, normalization and standardization [36]. These procedures transform range or data distribution for each feature, which allows certain classifiers to show better performance. However, there is a reason for not using data transformation in machine learning models that are applied to security systems. Normalization and standardization are based on transforming feature values in a certain dataset, using statistical properties of each feature. The goal of the detection system is to detect malware unknown during the training phase. With transformed data, new malware can have features whose values are out of bounds of the normalized or standardized dataset that was used for training.

Table 2. Intraflow features used in the C&C flow detection

No	Feature	Feature description	AUC
1	Average flow throughput (bytes/s)	Number of bytes per second of the time series which is re-calculated after each new sample is added to the time series.	0.96
2	Average number of packets in a flow (packets/s)	Number of packets per second of the time series which is re-calculated after each new sample is added to the time series.	0.93
3	Median of the throughput time series (median)	Median value of all time bins in throughput time series	0.553
4	Average throughput difference	Difference between the average flow throughputs for bidirectional sessions	0.738
5	Average throughput ratio (Bidirectional bytes ratio)	Ratio between the average flow throughputs for bidirectional sessions.	0.805
6	Standard deviation (Std)	Throughput time series standard deviation	0.863
7	Correlation (Corr)	Pearson correlation coefficient of the throughput time series	0.551
8	ADF Stat value	Augmented Dickey–Fuller test (ADF test): a statistical test which determines whether a time series is stationary or not.	0.511
9	ADF Test p-value	p-value for the ADF test	0.607
10	Autocorrelation (Autocorr)	Autocorrelation of the bytes count time series calculated at the lag equal to the highest probability in the empirical distribution of the flow inactive periods (described below). This feature aims to capture the periodic behavior of the flow.	0.985
11	Entropy	Throughput time series entropy calculated using the formula $H(X)=-iP(xi)\ln(P(xi))$, where P(xi)is the probability of a packet having given size.	0.767
12	Best/Other ratio	The ratio of the probabilities of the value that appears the most often in the empirical probability distribution and the most probable highest value in the periodicity dictionary to the sum of other values	0.893
13	Best/Second ratio	The ratio of two most probable values in the empirical distribution	0.85
14	Top Two	Sum of top two values that appear the most often in the empirical distribution	0.849

5.1. Classifier Description

The classifiers used in this paper are: K-nearest neighbor (KNN), logistic regression, and random forest from the Python sklearn library. The K-nearest neighbor algorithm is a non-parametric method used in pattern recognition [35]. Each data point is represented

by a n-dimensional vector in feature space. The training phase of the algorithm consists of storing the data points with labels. The classification phase begins by considering each new data point's position in feature space. Then, by using a distance metric, k-nearest neighbors are determined, and a data point is classified by majority voting.

Logistic regression uses a sigmoid function to perform classification. This machine learning model is the discrete version of linear regression. Using the data point representation as *n*-dimensional vectors designated with x_i, polynomial real coefficients designated with β_i, and error value designated with ε, the input to the sigmoid function has the following form:

$$f(x) = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon. \quad (1)$$

The value of probability is between 0 (data point certainly belongs to class "0") and 1 (data point certainly belongs to class "1"). The classes named "0" and "1" are defined before classification. During the training phase, each data point is fed into the sigmoid function, fitting the coefficients and error value so that the best fit is achieved.

Random forest belongs to a class of data science models called ensemble methods. In essence, an ensemble method combines many simple or weak classifiers, and the classification is performed using majority voting. The core classifier in this method is the decision tree, consisting of nodes and leaves. Each node contains a boundary value for a feature, whilst the leaves contain the information to which class a certain data point belongs. Selecting a subset of features, as well as a ranking function, the features are ranked and the decision tree is constructed. This is done by putting the features as boundary values inside the decision tree, going from the topmost ranking feature down to the lowest. The random forest consists of a certain number of random trees, and the class for the data point is determined by majority voting.

5.2. Feature Selection

Dimensionality reduction is an important step in order to create a classifier suitable for the implementation in the network elements in terms of memory and processing power consumption. For this purpose, the area under the curve (AUC) for receiver operating characteristic (ROC) calculated as if each feature alone was used for the classification is used as a measure of the impact that the specific feature has on the classification.
The ROC is a plot that shows for each value of a feature the ratio of true positive rate (TPR) to true negative rate (TNR). The number of true positives and the number of true negatives is designated as TP and TN, while the number of false positives and the number of false negatives is designated as FP and FN. TPR and TNR two values are calculated as follows:

$$TPR = TP / (TP+FN), \quad TNR = 1 - Specificity = 1 - TN / (TN + FP). \quad (2)$$

The ROC is a probability curve, and the AUC value shows how well a certain feature separates the data correctly into classes. The AUC holds values in the [0,1] interval, where a value closer to one indicates a better feature score. The features with the AUC values higher or equal to 0.85 are autocorrelation, throughput (bytes/s), packets/s, Best/Other ratio, Standard deviation, Best/Second ratio. However, the features that are chosen for the classification process should not be mutually correlated as this means that they describe the same or similar dataset property. Therefore, a correlation matrix

was used to test the correlation between all pairs of features. The correlation matrix is shown in Figure 5. High correlation (greater than 0.8) is observed between the bytes/s and packets/s features. Therefore, the five features that are chosen as input features to the classification process are Autocorrelation, Throughput (bytes/s), Best/Other ratio, Standard deviation of the throughput time series and Best/Second ratio.

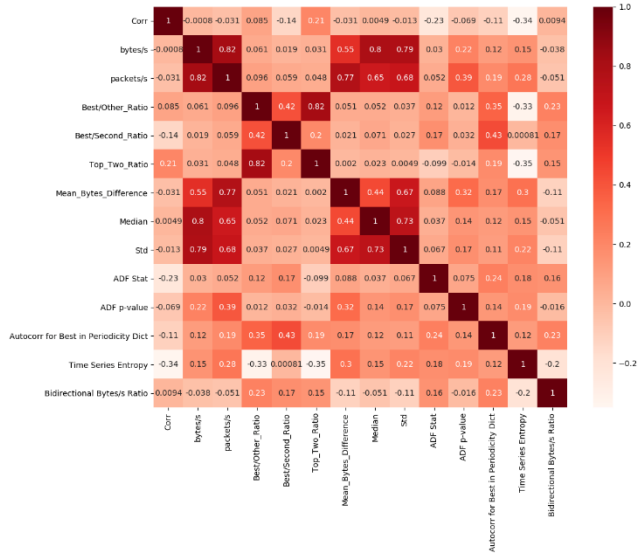

Fig. 5. Correlation matrix for feature selection - A correlation matrix of all the input features, which is used for feature selection.

5.3. Estimator hyperparameters

Each estimator has a set of hyperparameters, which define its behavior. In this phase, we analyzed the combination of estimator parameters that maximizes the scores used in classification using a "fit" and a "score" method implemented by the GridSearchCV method [37] in the sklearn Python library. The method is slightly modified as described below to fit the analyzed datasets which are disbalanced. The set of scores used in this paper are precision (PREC), recall (REC), and f1-score (F1) [35]. These scores are given in the equations below:

$$\mathrm{PREC} = \mathrm{TP} / (\mathrm{TP} + \mathrm{FP}), \tag{3}$$

$$\mathrm{REC} = \mathrm{TP} / (\mathrm{TP} + \mathrm{FN}),$$

$$\mathrm{F1} = 2 \cdot \mathrm{PREC} \cdot \mathrm{REC} / (\mathrm{PREC} + \mathrm{REC}).$$

Precision is a measure which shows how many false negatives are there during classification, while recall is a measure which shows how many false negatives are there

during classification. The F1 score represents a measure whose objective is to analyze the trade-offs between correctness and coverage in classifying positive instances [38], as it includes both precision and recall. Although accuracy is a more intuitive metric, these three measures are chosen because there is a difference in sizes of normal and botnet flows that is larger than 20 percent, which is considered a disbalanced dataset. In case of disbalanced datasets accuracy can be biased towards the majority class. Even high accuracy values can yield poor classification rates on minority classes and classifiers can have low predictive power [39]. Botnet C&C datasets are disbalanced because botnet C&C communication is a relatively rare occurrence in network traffic compared to all the other traffic and produces typically one or few flows at a time.

Balanced sampling was used to split data into training and testing subsets with a 90/10 ratio. The data is split in such a manner, in order to preserve the ratio of classes from the dataset, which is known as stratified sampling. A similar approach was presented in [40], in order to account for class imbalance in the dataset. After splitting the dataset, the model is trained and tested, and the scores are extracted. This procedure is repeated 100 times, and mean values are calculated for all scores (Monte Carlo cross-validation [41]). This approach was taken by the authors in order to have as much variation to the training and testing set scenarios, while keeping the number of scenarios tested to a minimum. Again, because the dataset is disbalanced, this method is more appropriate than the ubiquitous k-fold cross-validation, which splits the whole dataset into k equal parts. Such an approach produces some parts of the dataset to have low presence of botnet data points. After calculating all the scores, the list of parameter combinations is searched for those having the maximum values for all scores. In the following paragraph, an overview of the tested hyperparameters for both datasets is given. After the name of each hyperparameter, the best value is given in parentheses. If there is one value, that means that the value is best for both datasets, while if there are two values, the first one refers to the ETF-IoTB dataset, and the other refers to the IOT23 dataset.

The hyperparameters that were considered for the KNN classifier are the following: number of neighbors (5), weight function (uniform), neighbor computation algorithm (ball tree), and the distance metric (Manhattan). The hyperparameters that were considered for the logistic regression classifier are the following: inverse of the regularization strength (0.25), class weights (none), option of adding a constant to the decision function (false), maximum number of iterations (100), solver (newton-cg, lbfgs), and the tolerance for the stopping criteria (0.001, 10^{-5}). The hyperparameters that were considered for the random forest classifier are the following: class weights (balanced subsample), ranking function (entropy), max number of features (none), and the total number of estimators (200).

6. PI-BODE Classification Evaluation

This section presents and discusses the results of machine learning classification for three estimators: KNN, Logistic regression and Random Forest, using the hyperparameter values determined in the previous section. Precision, Recall and F1 scores for the PI-BODE detection method are given in Tables 3 and 4 for the ETF-IoTB and IoT23 datasets, respectively.

Table 3. Scores for all the estimators for the ETF-IoTB dataset

Model	botnet precision	botnet recall	botnet f1 score	normal precision	normal recall	normal f1 score
KNN	0.9363	0.9141	0.9218	0.9967	0.9973	0.9970
Logistic regression	0.9131	0.9247	0.9150	0.9971	0.9962	0.9967
Random forest	0.5954	0.6240	0.5994	0.9859	0.9948	0.9902

Table 4. Scores for all the estimators for the IOT23 dataset

Model	botnet precision	botnet recall	botnet f1 score	normal precision	normal recall	normal f1 score
KNN	0.8683	0.8759	0.8690	0.8558	0.8382	0.8423
Logistic regression	0.8347	0.8778	0.8510	0.8522	0.7858	0.8093
Random forest	0.5564	0.6071	0.5780	0.7566	0.8733	0.7724

Several conclusions can be drawn from these results. First, the highest scores on both datasets prove that this method can be used to detect botnet C&C communication with high precision. Second, even though the feature selection was done on the ETF-IoTB dataset, the results of the detection method on the IoT23 dataset that contains a different set of malware samples (Muhstik, Kenjiro, Tori in addition to those in the ETF-IoTB dataset) revealed similar precision compared to the other research papers. It can be concluded that the key C&C features in different malware samples are not largely different in two datasets. Also training a model, which uses features obtained from the intraflow statistics, can detect the C&C communication of the malware samples that are not in the dataset used to train the model, so the zero-day malware detection is possible. Third, the results show that for both datasets, KNN and Logistic Regression classifiers have given the best results, while having little discrepancies between the precision and recall scores. This means that there is little difference between the number of false negatives and false positives, demonstrating detection stability. The consistency in classifier performance shows that the proposed method performs well regardless of the dataset used. On one hand, KNN classifier shows a slightly better score than Logistic Regression classifier, while on the other KNN has much greater memory usage and is impractical to be used in network elements. Since ultimately these models should be implemented in a security system, using a Logistic Regression classifier is recommended, because it gives nearly the best scores among all the classifiers, while keeping the memory usage at a lower level.

Papers in references [11] to [13] have, similar to our approach, sought to detect botnets and ransomware via the C&C communication. In paper [11], the research showed an accuracy of 80% in classifying botnet periodicity, which is lower than scores obtained in our research for the best classifiers. Tables 5 and 6 show research results presented in papers [12] and [13], respectively. In paper [12], the BotMark detection system has shown a high accuracy (0.9994) and low false positive rate scores. However, since the F1-scores are low for all previous research methods (0.115 for BotMark, compared to 0.915 and 0.85 for PI-BODE Logistic Regression) while the false positive rate is also low, this means that the false negative rate is high, which further implies that the stability of the BotMark classifiers is low, and certainly much lower than in PI-BODE.

Table 5. Scores for all the methods in paper [12]

Method	F-score	Accuracy	True Positive Ratio	False Positive Ratio
Similarity	0.087247	0.9904	0.9836	0.009645
Stability	0.060732	0.9868	0.9115	0.013181
C-flow	0.152788	0.9949	0.9836	0.005108
Graph	0.034811	0.9166	0.9836	0.083478
BotMark	0.115207	0.9994	0.9836	0.000641

Table 6. Scores for all the methods in paper [13]

No of features	Precision	Recall	F1-score
28	0.83	0.89	0.86
8	0.86	0.87	0.87

Authors in [13] use precision, recall and F1-score as scores for their classifier. In both cases presented in [13], with 8 and 28 features, score values are in the range 0.83 to 0.89. This is worse classification performance than with the PI-BODE whose scores when the system is trained, verified and tested on the ETF-IoT dataset, are in the range 0.914 to 0.936 for the KNN and Logistic Regression classifiers. Even with the classifier training on the ETF-IoT dataset and testing on the IOT23 dataset (zero-day detection case), the scores are in the range 0.83 to 0.88 which is comparable to the scores obtained in [13]. Also, what needs to be taken into account is the fact that both of these papers show only the scores in regards to the whole dataset. In the case of balanced datasets, scores can be presented this way. However, since malware represents a minor part of each network traffic, such datasets are disbalanced by nature. Scores presented in this way should be treated with caution, since there is no information on the machine learning model's performance on both normal and malware traffic. At last, but not at

least ransomware C&C communication detection described in [13] is based on full packet capture analysis meaning a much higher network and storage overhead.

7. Conclusion

In this paper, we proposed and demonstrated a novel approach for the detection of botnet C&C communication based on SDN and intraflow statistics. It provides more traffic description features than flow-based methods, while not using the analysis of all packets on a link. Thus, this method represents an in-between solution that saves storage and processing power up to two orders of magnitude, yet provides enough detail about network conversations for distinguishing malware and normal traffic. Experimental results have shown that PI-BODE achieves the same or higher level of botnet detection accuracy as in the similar systems which use full traffic analysis. Other conclusions can be drawn from the presented research. First, the results prove that botnet C&C communication can be detected using the intraflow statistics and proposed traffic features. Second, since PI-BODE training and evaluation have been done on the datasets that contain various malware samples, it shows that different versions of botnet malware have similar C&C communication characteristics and that new malware samples can be detected with the system trained on the older malware samples. Third, using an SDN controller as an intraflow statistics gathering tool allows creating a simple intrusion prevention system for the devices that lack security, such as IoT devices. Using the PI-BODE approach as a basis, future research should attempt to explore additional statistical parameters of the network flow, analyze other malware dynamic properties that can be used for the detection, explore the use of other programmable platforms (e.g., P4-based) and to assess how to improve the malware detection speed. The research presented in this paper can be extended to using other state-of-the-art machine learning algorithms, such as boosting models (such as extreme gradient-boosting and light gradient-boosting machine) and deep learning models. Also, since computer viruses are constantly evolving, training a dataset on current botnets may not be applicable to detecting botnets in the future. Therefore, future research should be directed towards developing a machine learning model with on-line learning capabilities.

References

1. Vormayr, G., Zseby, T., Fabini, J.: Botnet Communication Patterns, IEEE Commun. Surv. Tutorials, vol. 19, no. 4, pp. 2768–2796, 10.1109/COMST.2017.2749442 (2017)
2. Chen, R., Niu, W., Zhang, X., Zhuo, Z., Lv, F.: An Effective Conversation-Based Botnet Detection Method, Math. Probl. Eng., vol. 2017, pp. 1–9, 10.1155/2017/4934082 (2017)
3. B. Krebs, Zyxel Flaw Powers New Mirai IoT Botnet Strain, Krebs on Security website, https://krebsonsecurity.com/2020/03/zxyel-flaw-powers-new-mirai-iot-botnet-strain/ (accessed on December 21st 2022)

4. Štampar, M., Fertalj, K.: Applied Machine Learning in Recognition of DGA Domain Names, Computer Science and Information Systems, vol. 19, No. 1, 205-227., 10.2298/CSIS210104046S (2022)
5. Jovanović Đ., Vuletić P.: Analysis and Characterization of IoT Malware Command and Control Communication, Telfor Journal Vol.12 No.2, p. 80-85, 10.5937/telfor2002074B (2020)
6. Ibrahim, J., Gajin, S.: Entropy-based Network Traffic Anomaly Classification Method Resilient to Deception. Computer Science and Information Systems, Vol. 19, No. 1, 87-116., 10.2298/CSIS201229045I (2022)
7. Asadi, M., Jabraeil Jamali, M. A., Parsa, S., Majidnezhad, V.: Detecting botnet by using particle swarm optimization algorithm based on voting system. Future Generation Computer Systems, vol. 107, 95–111., 10.1016/j.future.2020.01.055 (2020)
8. Livadas C., Walsh, R., Lapsley, D.E., Strayer, W.T.: Using machine learning techniques to identify botnet traffic, LCN, pp. 967–974., 10.1109/LCN.2006.322210 (2006)
9. Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B.: Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset, Future Generation Computer Systems, vol. 100, p. 779-796, 10.1016/j.future.2019.05.041 (2019)
10. Lee, J.-S., Jeong, H., Park, J.-H., Kim, M., Noh, B.-N.: The activity analysis of malicious http-based botnets using degree of periodic repeatability, 2008 International Conference on Security Technology, pp. 83–86., 10.1109/SecTech.2008.52 (2008)
11. Eslahi, M., Rohmad, M. S., Nilsaz, H., Naseri, M. V., Tahir, N. M., Hashim, H.: Periodicity classification of HTTP traffic to detect HTTP Botnets, 2015 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), p. 119–123. 10.1109/ISCAIE.2015.7298339 (2015)
12. Wang, W., Shang, Y., He, Y., Li, Y., Liu, J.: BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors, Information Sciences, Vol. 511, p. 284–296. 10.1016/j.ins.2019.09.024 (2020)
13. Cusack, G., Michel, O., Keller, E.: Machine Learning-Based Detection of Ransomware Using SDN, In Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization - SDN-NFV Sec'18, pp. 1–6., 10.1145/3180465.3180467 (2018)
14. Shahzana Liaqat, S., et al.: SDN orchestration to combat evolving cyber threats in Internet of Medical Things (IoMT), Computer Communications, Volume 160, p. 697-705, 10.1016/j.comcom.2020.07.006 (2020)
15. Bilge, L. et al.: DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis, ACSAC '12, 10.1145/2420950.2420969 (2012)
16. Blaise, A., Bouet, M., Conan, V., Secci, S.: Detection of zero-day attacks: An unsupervised port-based approach, Computer Networks, vol.180, 10.1016/j.comnet.2020.107391 (2020)
17. De La Torre Parra, G., Rad, P., Choo, K.-K. R., Beebe, N.: Detecting Internet of Things attacks using distributed deep learning, Journal of Network and Computer Applications, vol. 163, 10.1016/j.jnca.2020.102662 (2020)

18. Kurniabudi, et al.: CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection, IEEE Access, Volume 8, p. 132911-132921, 10.1109/ACCESS.2020.3009843 (2019)
19. Sharifnya, R., Abadi, M.: DFBotKiller: Domain-flux botnet detection based on the history of group activities and failures in DNS traffic, Digit. Investig., vol. 12, pp. 15–26, 10.1016/j.diin.2014.11.001 (2015)
20. Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E.: A Comprehensive Measurement Study of Domain Generating Malware, Open access to the Proceedings of the 25th USENIX Security Symposium is sponsored by USENIX, pp. 1996–2014, 10.5555/3241094.3241115 (2016)
21. Tong, V., Nguyen, G.: A method for detecting DGA botnet based on semantic and cluster analysis, ACM Int. Conf. Proceeding Ser., vol. 08, pp. 272–277, 10.1145/3011077.3011112 (2016)
22. Wang, T. S., Lin, H. T., Cheng, W. T., Chen, C. Y.: DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis, Computer Security, vol. 64, pp. 1–15, 10.1016/j.cose.2016.10.001 (2017)
23. Al-Hadhrami, Y., Hussain, F. K.: Real time dataset generation framework for intrusion detection systems in IoT, Future Generation Computer Systems, Vol. 108, p. 414–423., 10.1016/j.future.2020.02.051 (2020)
24. de Souza, C. A., et al.: Hybrid approach to intrusion detection in fog-based IoT environments, Computer Networks, 180, 107417., 10.1016/j.comnet.2020.107417 (2020)
25. Hosseini, S., Zade, B. M. H.: New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN, Computer Networks, Vol. 173, 10.1016/j.comnet.2020.107168 (2020)
26. Zhou, Y., Cheng, G., Jiang, S., Dai, M.: Building an efficient intrusion detection system based on feature selection and ensemble classifier, Computer Networks, 174, 10.1016/j.comnet.2020.107247 (2020)
27. Gardiner, J., Cova, M., Nagaraja, S.: Command & Control: Understanding, Denying and Detecting, vol. cs.CR, no. February, 10.48550/arXiv.1408.1136 (2014)
28. Antonakakis, M. et al.: Understanding the Mirai botnet, SEC'17, p. 1093–1110., 10.5555/3241189.3241275 (2017)
29. Shafiq, M., et al.: Selection of effective machine learning algorithms and Bot-IoT attacks traffic identification for internet of things in smart city, Future Generation Computer Systems, Vol. 107, p. 433–442. 10.1016/j.future.2020.02.017 (2020)
30. Parmisano, A., Garcia, S., Erquiaga, M. J.: A labeled dataset with malicious and benign IoT network traffic. Stratosphere Laboratory, 10.5281/zenodo.4743746 (2020)
31. Jovanovic, G., Vuletić, P.: ETF IoT Botnet Dataset, Mendeley Data, V1, 10.17632/nbs66kvx6n.1 (2021)
32. abuse.ch, URLHaus, a database of malware URLs, https://urlhaus.abuse.ch/ (accessed on December 21st 2022)
33. García et al.: An Empirical Comparison of Botnet Detection Methods, Computers & Security, 10.1016/j.cose.2014.05.011 (2014)

34. Joy, A package for capturing and analyzing network flow data and intraflow data, for network research, forensics, and security monitoring, https://github.com/cisco/joy, (accessed on September 2nd 2020)
35. Skiena, S. S.: The Data Science Design Manual, Springer, 10.1007/978-3-319-55444-0 (2017)
36. Raschka, S. :About Feature Scaling, https://sebastianraschka.com/Articles/2014_about_feature_scaling.html (accessed on May 21st 2022)
37. sklearn documentation, GridSearchCV, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, (accessed on May 21st 2022
38. Fernández, A. et al.: Learning from Imbalanced Data Sets, Springer, 10.1007/978-3-319-98074-4 (2018)
39. Gibert, D., et al.: The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of Network and Computer Applications, Vol. 153, 10.1016/j.jnca.2019.102526 (2020)
40. Apruzzese, G. et al.: Evaluating the effectiveness of Adversarial Attacks against Botnet Detectors, 2019 IEEE NCA, 978-1-7281-2522-0/19/ (2019)
41. Dubitzky, W., Granzow, M., Berrar, D.: Fundamentals of data mining in genomics and proteomics, Springer Science & Business Media, p. 178., 10.1007/978-0-387-47509-7 (2007)

Đorđe Jovanović obtained his BSc and MSc in Computer Networks from University of Belgrade, School of Electrical Engineering (ETF). Since 2019, he began his work as an assistant researcher-trainee at the Mathematical Institute of Serbian Academy of Sciences and Arts (MISANU/SASA). Since 2022, he works as an assistant researcher at MI SANU. His research interests encompass Software-Defined Networks (SDN), computer networks, botnet detection, machine learning, optimization, and metaheuristics.

Pavle Vuletić obtained his BSc, MSc and PhD in Computer Systems and Network Architecture from University of Belgrade, School of Electrical Engineering (ETF). He used to work on all positions from network engineer to the deputy director of AMRES, national research and education network where he participated in the establishment of the first national CSIRT team. He is currently an associate professor at the ETF teaching Data Security, Computer Systems and Network Security, Advanced Computer Networks and SDN courses, leads the Laboratory for Information Security at ETF. His research interests span from data protection and privacy, network and systems security, network and system performance evaluation to the programmable networks and network and systems management.

Received: February 22, 2023; Accepted: September 10, 2023.