https://doi.org/10.31436/iiumej.v23i1.1789

BOTNET DETECTION USING INDEPENDENT COMPONENT ANALYSIS

WAN NUR HIDAYAH IBRAHIM[1,2], MOHD SYAHID ANUAR[5], ALI SELAMAT[1,3,4*] AND ONDREJ KREJCAR[4]

[1]School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Media and Game Innovation Centre of Excellence (MaGICX), Universiti Teknologi Malaysia, 81310 Johor Baharu, Johor, Malaysia
[2]Jabatan Teknologi Maklumat dan Komunikasi (JTMK), Politeknik Sultan Idris Shah, 45200 Sg Lang, Selangor, Malaysia
[3]Malaysia Japan International Institute of Technology (MJIIT), Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra, 54100 Kuala Lumpur, Malaysia
[4]Center for Basic and Applied Research, Faculty of Informatics and Management, University of Hradec Kralove, Rokitanskeho 62, 500 03 Hradec Kralove, Czech Republic
[5]Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra, 54100 Kuala Lumpur, Malaysia

**Corresponding author: aselamat@utm.my*

(Received: 23rd January 2021; Accepted: 9th August 2021; Published on-line: 4th January 2022)

ABSTRACT: Botnet is a significant cyber threat that continues to evolve. Botmasters continue to improve the security framework strategy for botnets to go undetected. Newer botnet source code runs attack detection every second, and each attack demonstrates the difficulty and robustness of monitoring the botnet. In the conventional network botnet detection model that uses signature-analysis, the patterns of a botnet concealment strategy such as encryption & polymorphic and the shift in structure from centralized to decentralized peer-to-peer structure, generate challenges. Behavior analysis seems to be a promising approach for solving these problems because it does not rely on analyzing the network traffic payload. Other than that, to predict novel types of botnet, a detection model should be developed. This study focuses on using flow-based behavior analysis to detect novel botnets, necessary due to the difficulties of detecting existing patterns in a botnet that continues to modify the signature in concealment strategy. This study also recommends introducing Independent Component Analysis (ICA) and data pre-processing standardization to increase data quality before classification. With and without ICA implementation, we compared the percentage of significant features. Through the experiment, we found that the results produced from ICA show significant improvements. The highest F-score was 83% for Neris bot. The average F-score for a novel botnet sample was 74%. Through the feature importance test, the feature importance increased from 22% to 27%, and the training model false positive rate also decreased from 1.8% to 1.7%.

ABSTRAK: Botnet merupakan ancaman siber yang sentiasa berevolusi. Pemilik bot sentiasa memperbaharui strategi keselamatan bagi botnet agar tidak dapat dikesan. Setiap saat, kod-kod sumber baru botnet telah dikesan dan setiap serangan dilihat menunjukkan tahap kesukaran dan ketahanan dalam mengesan bot. Model pengesanan rangkaian botnet konvensional telah menggunakan analisis berdasarkan tanda pengenalan bagi mengatasi halangan besar dalam mengesan corak botnet tersembunyi seperti teknik penyulitan dan teknik polimorfik. Masalah ini lebih bertumpu pada perubahan struktur berpusat kepada struktur bukan berpusat seperti rangkaian rakan ke rakan (P2P). Analisis tingkah laku ini

seperti sesuai bagi menyelesaikan masalah-masalah tersebut kerana ianya tidak bergantung kepada analisis rangkaian beban muatan trafik. Selain itu, bagi menjangka botnet baru, model pengesanan harus dibangunkan. Kajian ini bertumpu kepada penggunaan analisa tingkah-laku berdasarkan aliran bagi mengesan botnet baru yang sukar dikesan pada corak pengenalan botnet sedia-ada yang sentiasa berubah dan menggunakan strategi tersembunyi. Kajian ini juga mencadangkan penggunakan Analisis Komponen Bebas (ICA) dan pra-pemprosesan data yang standard bagi meningkatkan kualiti data sebelum pengelasan. Peratusan ciri-ciri penting telah dibandingkan dengan dan tanpa menggunakan ICA. Dapatan kajian melalui eksperimen menunjukkan dengan penggunaan ICA, keputusan adalah jauh lebih baik. Skor F tertinggi ialah 83% bagi bot Neris. Purata skor F bagi sampel botnet baru adalah 74%. Melalui ujian kepentingan ciri, kepentingan ciri meningkat dari 22% kepada 27%, dan kadar positif model latihan palsu juga berkurangan dari 1.8% kepada 1.7%.

KEYWORDS: *botnet detection; flow-based; machine learning; independent component analysis; traffic analysis*

1. INTRODUCTION

A botnet is a collection of computers infected by malicious software (malware) that a botmaster manages. All Internet-of-Things (IoTs) devices, such as closed-circuit television cameras (CCTV), web cameras, computers, and mobile devices, can be infected devices. The vulnerabilities and the computing resources of these infected devices are exploited where they operate remotely as servants following the instructions given by their botmaster. The main aim of assigning a botnet is to launch an assault on the victim. However, the number of bots depends on the frequency of the attacks. Therefore, the most significant factor contributing to the frequency of the attacks is the number of bots in botnet environments [1], [2].

They can execute major attacks on victims, such as DDOS or email spam, because of the large number of bots, rendering victims unable to function for hours or days. For example, in the Mirai incident of 2016, the vast and unlimited number of bots produced a massive impact assault [3]. In the Mirai incident of 2016, the attacks were identified from 600,000 Internet-of-things (IoT) devices [4]. At that moment, Mirai attacks were noteworthy since the bots used Internet-of-things (IoT) devices, not just computers or laptops. Consider the result if 600,000 devices concurrently sent a ping to a specific website, leading to that website being overwhelmed, inaccessible, and its services slowed down.

The botnet detection model has become a hot topic among researchers due to the history of botnet attacks and their impact on the industry. The arms race never ends between the botmasters and the researchers trying to beat each one. Every group continues to develop its abilities, and this can be seen through the botnet revolution. Botnet evolves or mutates every day after the source code has been released to the public [5]. It can be seen in the Mirai botnet and the Mirai version. Two months after the release of the Mirai source code to the public, the bots multiplied with variant complexity, from 213,000 to 493,000, twice. They show the statistics of various botnet attacks on Securelist websites, and 39.35 percent of new botnet found in 2018 is based on the Kaspersky Lab Botnet Monitoring project compared to 2017 botnet attacks [6]. Subsection 1.1 briefly clarifies why the botnet relies heavily on the Rallying stage or C&C stage Command & Control server. The importance of preventing the server from being identified by the security system also explains the botnet structure's revolution. The botnet framework revolution switches from centralized to a decentralized. Centralized botnet, such as IRC and HTTP, via the primary server, call the Command & Control server. Decentralized botnet, such as Peer-to-peer (P2P), are more advanced since the bots themselves can act as servers. P2P is designed to hide the C&C server, as stated in [7], [8].

The botnet's strength lies in its capacity to elude security systems and carry out large-scale attacks thanks to various tactics such as packet data concealment and encrypted packet data [9]. A botnet can hide from the protection system and imitate the regular traffic flow where normal traffic is usually more random [10], it then waits for stage and imbalanced class distribution. The P2P technique is also a part of the concealment strategy to mask the C&C server [7], [8].

The botnet is now becoming a profitable business, according to [8], where the botmaster provides the service for any cyber-attacks. However, the current capability leading the business of these services must monitor the bots, advise of subsequent attacks, lengthen the duration of the attacks, and avoid monitoring its identity.

1.1 Botnet Life-Cycle and Structure

It is necessary to understand the life cycle of the botnet when designing a behavior-based analysis of the botnet detection model since the choice of related features of this model depends on it. Other than that, identification must also be carried out before the attacks occur and it is too late [11]. As below, the botnet life-cycle can be divided into four main stages: -

- Phase I: Injection (I)
- Phase II: Command & Control (C&C)
- Phase III: Attack
- Phase IV: Release

The first stage is injection or replication. This stage can be achieved in several ways on the network, such as exchanging folders, visiting malicious websites, or adding emails. The bot herder increases the number of bots at this level. The Command & Control stage or 'Rallying' stage is the second stage. The infected devices already behave like bots in this process. The bots keep updating the devices' status, and if necessary, the bot herder submits a new source code [12]. The revised source code and the vulnerability report are designed to ensure the bots are undetected and robust [13]. The third stage is the attack phase, where all bots are targeted at attacking specific victims. The botmaster gives an attack launch order, and the bots simultaneously launch the attack based on the command. The Release Period is the last stage. The release stage is where the botmaster removes fingerprints, substitutes new systems for identified bots, and does not leave a digital footprint behind. Often the botmaster distributes the source code to hinder government investigations. During this process, learning from the previous attack, the functionality of the bot system is also enhanced [14].

We concentrate on botnet activity in the process of Command & Control or Rallying for our study. The infected system continues to attempt to connect to the C&C server to send reports of the infected devices. The system also receives updated source code to keep hiding from protection [8]. Based on a Kaspersky Lab study [15], monitoring DDOS attacks is the correct time to detect botnet to intercept the command from the Command & Control.

1.2 Motivation and Contribution

Our inspiration is the potential and consequences of a botnet (a botnet attack), to discover its method before the attack is launched. However, our critical general incentive to build a model that can predict a novel botnet is due to the continuous evolution of the botnet. The technological motivation for using behavior-based patterns and flow-based functionality is due to the shortcomings in detecting the new forms of botnets in the signature-based detection model. Other than that, we were motivated by the research from [16] that combined Principal Component Analysis (PCA) for clustering with k-means.

https://doi.org/10.31436/iiumej.v23i1.1789

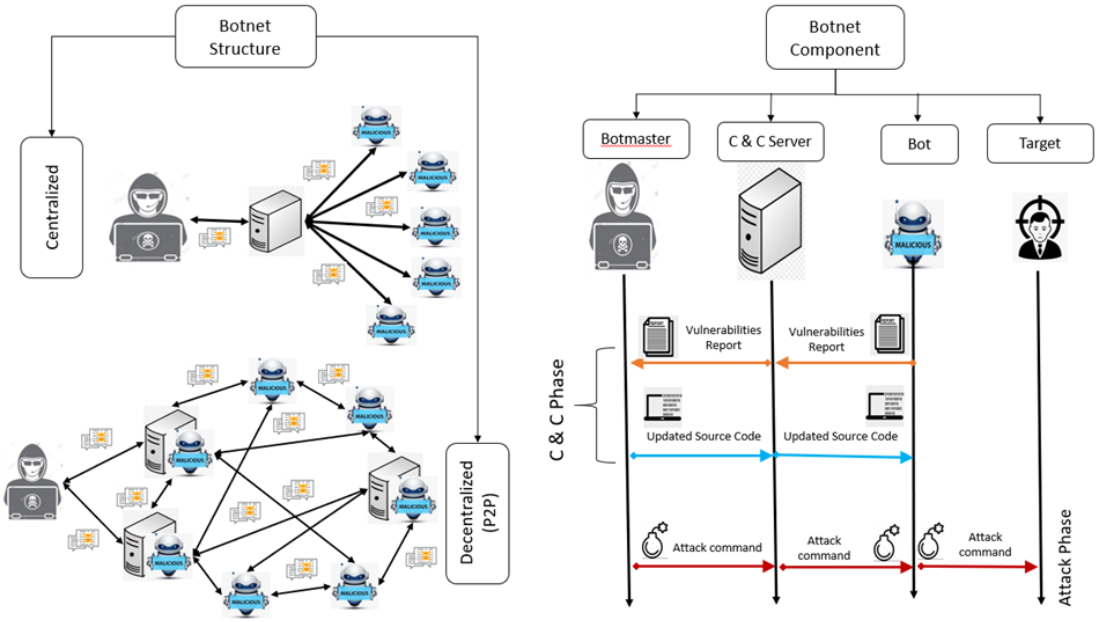


Fig. 1: The botnet structure and botnet component.

In statistics, principal component analysis is a technique used to describe a data set in terms of new uncorrelated variables ("components"). The components are ordered by the amount of original variance they describe, so the technology helps reduce the dimensionality of a data set. In comparison, Independent Component Analysis (ICA) is a machine learning technique used to distinguish independent sources from a mixed input. Unlike principal component analysis, which focuses on maximizing data point variance, independent component analysis emphasizes independence or independent components. Since we are using aggregation for pre-processing data, the vital information might be lost, resulting in decreased performance, so we agreed to use ICA. The explanation about ICA is in subsection 0. The significant contributions to this analysis are: -

- This method can detect network packets even in concealment strategies such as obfuscation, code encryption, oligomorphic strategy, polymorphic strategy, and metamorphic.
- This method used the CTU-13 botnet benchmark dataset that consists of centralized and decentralized structures. It proves that our framework can detect both structures.
- The evaluation of this framework used the different types of botnets, proving that our framework can detect novel botnets.
- Our result shows the average 74% f-score that tests on five types of novel botnets.
- The performance of the framework is compared with other researchers that used the same source of data.

2. RELATED WORKS

The latest developments in the concealment technique of packet data in network traffic make the signature-based or content-based inefficient in detecting new forms of botnets. For example, Singh et al. [17] suggested that the signature time-to-time was revamped by the botnet and significantly modified. These changes in behavior caused signature-based analysis output to drop on the new release botnet because signature-based analysis relied heavily on the bot's

signature. In addition, many concealing tactics are used to mask packet data content in network traffic, including obfuscation, code encryption, oligomorphic approach, polymorphic strategy, and metamorphic strategy [18].

Patsakis et al. [8] raised many concerns about DNS queries that have been used to conceal the botnet on the encrypted channel. Although AsSadhan et al. [17], claimed that packet data contents should be shielded to safeguard the identity of the private information of the individual or user, where only the header of the packets can be released to the public. This author also concentrated on analyzing traffic, exchanging packets, and providing a framework for lightweight security. But their work is considered DNS and is only for the DGA botnet. A model was also developed by this author using the actions of the botnet when interacting with others, but the time interval used for this study is 31-49 minutes.

Two common approaches, 1) payload-based and 2) traffic-based can be classified into machine learning models to detect network operations. The payload-based approach trains models based on characteristics derived from the payload/data portion of the packets transmitted over the network, as the name implies. The disadvantages of such models are the resource-intensive challenge (where features for each packet need to be evaluated), privacy problems, and encrypted information where features cannot be extracted [18,19]. By analyzing the communication packet headers or Netflow information, the traffic-based approach aims to mitigate some of the model's drawbacks. While privacy remains an issue with such an approach (such as individual IP addresses in features), this can be mitigated by aggregating time window records.

2.1 Behavior-based and Flow-based Features

What is behavior-based? What are the differences between behavior-based and signature-based? Behavior-based and signature-based are in contrast to each other. In computing, all objects have attributes that can be used to develop a custom signature. Signature-based analysis refers to detecting attacks by searching for specific patterns, like byte sequences in network traffic or known malicious instruction sequences used by malware [22]. This terminology is derived from anti-virus software, which refers to these detected patterns as signatures.

Although behavior-based analysis is an analysis that does not directly analyze the data like signature-based, there are some advantages of behavior-based analysis compared to signature-based analysis. For example, it is more secure or effective in detecting new and novel forms of malware threat. In addition, it can detect a single instance of malware that targets a person or organization. It can also identify what the malware does when files are opened in a specific environment and obtains comprehensive malware information. However, according to Resende and Drummond [21], most research defines behavior-based analysis with anomaly-based detection, but anomaly detection can also be done using signature-based analysis. So, it means that anomaly-based cannot be defined as behavior-based analysis in malware detection.

The definition of behavior-based Resende and Drummond [21] is the most accurate to our definition. Resende and Drummond [21] define behavior-based analysis in Network Intrusion Detection Systems as detection techniques that are not evaluated or referred directly to the source, destination, and payload of packets. It is an analysis that assesses the behavior of an object. Behavior-based detection can be performed by using API call logs [24], network flow (NetFlow) [10], and is also a hybrid between API call and Netflow [25]. A flow is a collection of packets that come from the same source and destination. Flow-based botnet detection techniques employ statistics of all packet headers in a flow (flow record). Because the flow-

based approach only catches packet header information, it can reduce the computational complexity [24-26] and be processed very quickly.

Flow-based features are the characteristics chosen to illustrate the network flow pattern or connection to distinguish either the usual network or a botnet network. Flow-based characteristics also relate to packet data information, such as total packets per second, bytes per packet, total packet bytes, and the number of packets [29]. The description of flow-based functionality is shown in Table 1 and consists of the features, the time window, and the data tools derived from published work. Thus, when designing our characteristics and the time window, Table 1 became our crucial guide.

The feature selection process was interpreted in the same concept as used in aggregation. As mentioned in Gezer et al. [40], one of the challenges in machine learning is feature selection because feature selection needs a good understanding of the domain knowledge. Not all features in the data are relevant to building the model (Resende and Drummond [21]). On the other hand, aggregation is a process of transforming the data based on a specific theory, and it enables the extraction of sufficient data for analysis. Based on [30], aggregation is a part of the data mining technique in machine learning for efficient knowledge discovery about network flows.

2.2 Independent Component Analysis

Independent Component Analysis (ICA) is a source separation technique in signal processing. According to [31], in their survey, ICA and PCA are among the popular methods used to select essential network features. PCA extracts and reduces the dimension of features, while ICA separates the noise to enhance and maximize each feature's data pattern [32]. The authors in [33] claim that principal component analysis (PCA) is a technique for reducing features by identifying the relevant feature set. The implementations of ICA and PCA clustering algorithm in feature selection has been reported [16]. It is a semi-supervised model where the author combined unsupervised and supervised techniques.

In ICA, the mutual connection between features is minimized by maximizing the non-Gaussianity. Research from Palmieri et al. [34] is the most similar to our approach. The author used ICA in Network Anomaly Detection from the University of Naples, Italy's network traffic. On the other hand, we try to find the implementation of ICA in detecting botnet, and we only found an article from Mao et al. [35] where this author used ICA in detecting spamming botnet.

We can summaries that behavior-based analysis that used the flow-based features can solve the issues of concealment botnet, but it produces high false alarm (false positive rate). High false alarm in machine learning occurs due to the unclear separation between classes that also come from the unclear pattern produce by the data. Although some attempts have been made to address this issue, it still puts limitations on ICA implementation.

3. PROPOSED FRAMEWORK AND PRE-EVALUATION RESULT

In order to reach the objective, this study proposed a new framework as shown in Fig. . The proposed framework starts with selecting the network traffic dataset and pre-processing the dataset. This study highlighted the pre-processing phases where the data is provided to produce a high-performance during classification.

https://doi.org/10.31436/iiumej.v23i1.1789

Table 1: Summary of behavior-based analysis in previous works

Author	Features	Time Window (seconds)	Data resources
Ehsan and Hamid [36]	Group duration time, Number of Receive Packets, Number of Send packets, Distance from the previous group	Group duration time	Kelihos, CVUT Malware Capture Facility Project
Stevanovic and Pedersen [37]	* Basic conversation features: Port number, layer 7 protocol, duration (last pkt - first pkts), the total number of packets, total number of bytes, mean of the number of bytes per packet & Std of the number of bytes per packet * Time-based-based features: number of packets & bytes per second, mean & std of packets inter-arrival time * Bidirectional features: a ratio of number of packets & bytes in and out, a ratio of inter-arrival times in and out * TCP specific features: such as percentage TCP SYN packets, percentages of TCP SYN-ACK packets, percentages of TCP ACK & percentages of TCP ACK PUSH packets	300 s	Combination of the dataset: - • ISCX, ISOT • Contagio • Honeyjar • Malware Capture Facility Project (MCFP)
Fernandez Maimo et al. [38]	* Number of flows, number of incoming flows, number of outgoing flows; * % of incoming and outgoing flows over the total * % of symmetric and asymmetric incoming over the total * Sum, maximum, minimum, mean, and variance of IP, packets per incoming outgoing, and totals flows	20 - 30 s	CTU-13
Debashi and Vickers [39]	Array 1:(Src Addr), Dst Port, Packet Count Algorithm 2: Dst Addr, Dst Port, Packet count. Array 2: Dst Addr, Src addr list, Src Addr count, Dst Port List, Dst Port Count	60 s	ISOT
Gezer et al. [40]	* Total forward & backward volume * Max forward & backward packet length * Min, Max, Mean before idle and before active. * Max backward inter-arrival time	600 s	capture own network data
Garg et al. [41]	Send Syn, Recv ACK, Recv Rst, Send pkts, Recv pkts, ICMP unr, Send len, Recv Len	120 s	Combination of P2P botnet

3.1 Input: Data Source and Data Distribution

For this study, the following vital data are extracted from the botnet benchmark dataset CTU-13 [42] from the website of the Stratosphere Research Laboratory. This dataset consists of 13 files with several types of botnets in different protocols and different structures. Since this framework aims to detect novel botnet, this dataset is divided into two sets, training and testing, for building the model and evaluating the dataset, as shown in Table 2. The model is evaluated with data from the evaluating dataset separated from data for building the model. The separation of this data ensures that the model is derived and tested using a different set of data, as explained in Step 2: Dividing Dataset in Section 3.2.

Table 2: The distribution of bot in training and evaluating dataset

Data File No	Duration (hrs)	Bot Name	No of Bots	Training Dataset	Evaluating Dataset
1	6.15	Neris	1		√
2	4.12		1		√
3	66.85	Rbot	1	√	
4	4.12		1	√	
5	11.63	Virut	1	√	
6	2.18	Menti	1		√
7	0.38	Sogou	1	√	
8	19.5	Murlo	1		√
9	5.18	Neris	10		√
10	4.75	Rbot	10	√	
11	0.26		3	√	
12	1.21	Nsis.ay	3	√	
13	16.36	Virut	1	√	

Table 2 summarizes the distribution of data based on the Data File No. In the third column are the names of bots in the dataset. The explanation of the bot name, bot category, and structure are given in Table 3. It is essential to have both structures (centralized and decentralized) in this research. As a result, our data source selection appears reasonable in terms of independent structure and bot reliability. Columns 5 and 6 in Table 2 show the separation of training and evaluating data for the novel bot.

Table 3: The description of the bot based on the name

	Bot Name	Bot Category	Structure
1	Neris	IRC	Centralized
2	Rbot	IRC	
3	Virut	HTTP	
4	Menti	IRC	
5	Soguo	HTTP	
6	Murlo	IRC	
7	Nsis.ay	P2P	Decentralized

3.2 Data Pre-processing

Pre-processing data is the phase in which the data is prepared before being incorporated into the algorithm to construct the prediction model. Since we used behavior-based analysis, the information needed to go through several steps. A behavior-based analysis is not a straightforward extraction process but rather a tool for analyzing the raw data. There are several vital components or measures that we have grouped into the pre-processing data

https://doi.org/10.31436/iiumej.v23i1.1789

module, such as Labeling, Cleaning, Dividing Dataset, Feature Selection, Aggregation, and Data Quality Process Implementation.

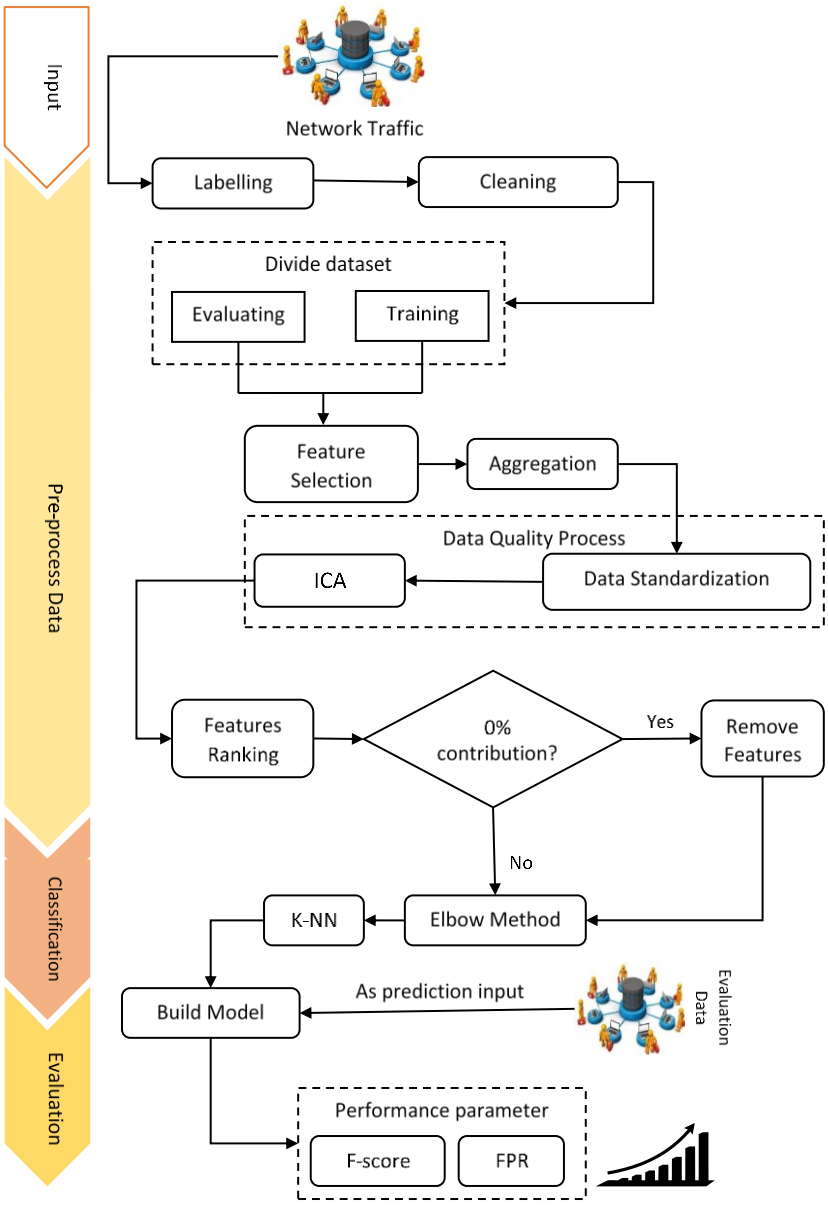


Fig. 2: The proposed framework for pre-processing flow-based features.

3.2.1 Step 1: Labeling and Cleaning

The first step was re-labeling the dataset. Although the CTU-13 dataset is supervised, the dataset contains labels but those labels are in string/text, not in numbers. There are 74 types of descriptive labels in CTU13, as we have shown in Appendix A, but basically, the label is based on 3 types of labels: 'Normal', 'Botnet', and 'Background'. Due to that, we re-labeled the CTU-13 as stated in (1) below. Once the labeling was completed, we removed the uncertain data in Label = 2 for the cleaning process.

$$Label = \begin{cases} 0 & if\ the\ label\ state\ is\ "Normal" \\ 1 & if\ the\ label\ state\ is\ "botnet" \\ 2 & if\ the\ label\ state\ is\ "background" \end{cases} \quad (1)$$

https://doi.org/10.31436/iiumej.v23i1.1789

3.2.2 Step 2: Dividing Dataset

After the cleaning process, the data was split into two main data sets: Creating Model Data and Analysing Data. This separation aimed to ensure that the construction model evaluation is performed on a novel botnet. The model was based on Constructing Model Data, divided into 70-30 ratios of training and testing data.

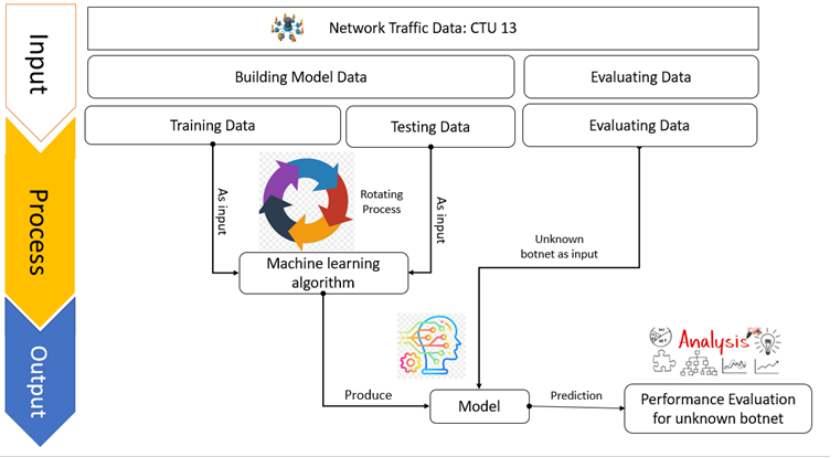


Fig. 1: Process of dividing dataset for novel botnet.

3.2.3 Step 3: Features Selection and Aggregation

Once the dataset division was completed, the dataset was ready for the following process: selecting the features and aggregating them in a specific time interval. We aimed to build the fastest detection, so for this research we chose a short time interval (1 sec) for aggregation.

Feature selection is a process of selecting specific variables/features/attributes in the data. The purpose of this process was to reduce the complexity and processing time. We chose the features based on the theory of communication. This theory is between botmaster and its bots during the C&C stage in the botnet life-cycle for this research. As mentioned in the bot life-cycle in subsection 1.1: Botnet Life-Cycle & Structure, during the C&C phase, bots and botmaster keep communicating. This communication pattern is different from the regular communication pattern, where a typical communication pattern is usually more random. In contrast, a bot's communication pattern is more uniform with the same amount of transferring data to multiple destinations. The features that we used for this research are shown in Table 4.

Table 4: Data type for features in the dataset

No	Feature	Data Types	Calculation
1	Destination Address	Categorical data	n(x)
2	Destination port	Categorical data	n(x)
3	Packet data	Continuous data	Min, Max, Median, std deviation, n(x)
4	Time	Categorical data	Δt

The data are aggregated or grouped in two parameters: time interval (t) and source address (Sip), as shown in Eq. (2). We used the aggregation technique to calculate the occurrence number of the communication within the time interval that represented the bot's behavior in a particular given time.

$$[Sip, t] = \{X_0\ ,\ X_1\ , \ldots\ldots, X_n\} \quad (2)$$

Since the data in the NetFlow is in continuous type and categorical data type, the aggregation of these two types of data is different, as shown in Eq. (3). If the data was continuous, we implemented the statistical technique such as minimum, maximum, median, standard deviation, and specific number, n(x). But if the data type was categorical data, we implemented only the total distinct number, n(x), where the total distinct number (n(x)) that define as the frequency of unique elements in the set can be described as shown in Eq. (4).

$$f\ (x) = \begin{cases} Min, Max, Median, std\ deviation, n(x) & if\ data\ is\ continuous\ number \\ Total\ Distint\ Number, n(x) & if\ data\ is\ categorical\ number \end{cases} \quad (3)$$

$$n(x) = \left\{X_i\ , X_j\ , \ldots\ldots, X_n \middle| X_i\ \neq\ X_j\ , i\ \neq j,\ i\ \geq 0, j = 1, \ldots\ldots, n\right\} \quad (4)$$

While for time, this data is used, whereas the aggregation or rounding process is shown in Eq. (1). Time is also used in calculating the Different Time (Δt) between the last time, tn and the start time, t1 in 1 second (time interval) duration. The equation for Different Time (Δt) is shown in Eqs. (5) and (6).

$$f(t) = \{t_1, t_2, \ldots\ldots\ldots, t_n\} \quad (5)$$

$$\Delta t = t_n - \ t_1 \quad (6)$$

3.2.4 Step 4: Data Quality Process

This process is the point where the most interesting things occur. This process improves the quality of the data and features, improving the performance of detecting novel botnets. Therefore, we label it as Data Quality Process to be represented as the objective of this combination process. This process utilized a two-step approach of standardization and Independent Component Analysis (ICA). Specifically, we used this theory because the classifier we chose was related to a distance-based classifier.

A) Standardization

Standardization is a re-scaling process for the distribution of the dataset to obtain the mean of the data equal to 0, and the standard deviation equal to 1. In other words, standardization is a process of centering the data. Standardizing a data set for a wide range of machine learning estimators is a common need. However, it could be harmful if the individual features do not look more or less like standard normally distributed data (e.g., Gaussian with 0 mean and unit variance). Therefore, test x is calculated as the standard value of:

$$z = {(x - \mu)}/{s} \quad (7)$$

where μ is the mean of the training samples or zero if, with mean= False, s is the standard deviation of the training samples or with std= False, respectively.

Centering and scaling occurred on each feature independently by computing the relevant statistics on the samples in the training set. Mean and standard deviation were then stored in a transform to be used on later data.

B) Independent Component Analysis

For this research, we used FastICA from *sklearn*. decomposition package in python, as shown in Algorithm 1 below. As the name FastICA implies, it is the short version of ICA. FastICA rotates the data until the data looks non-Gaussian in every axis. By making the

https://doi.org/10.31436/iiumej.v23i1.1789

mean equal to zero and normalization the variance in all directions, the algorithm can rotate the data in any direction. The process of normalizing the variance is called the whitening process. As shown in Algorithm 1, the python code to implement ICA is through the whitening process. The whitening process is the decorrelation to ensure that all features are treated equally before the Algorithm of ICA run.

After the centering process (mean equal to zero) and the whitening process (normalization of variance), the data ran the ICA algorithm. The main goal of ICA is to find the unmixing vector of W, where W is the inverse of A, X is the input data, and A is the mixing signal. The equation is shown in Eqs. (8) to (10):

$$X = A\,S \tag{8}$$

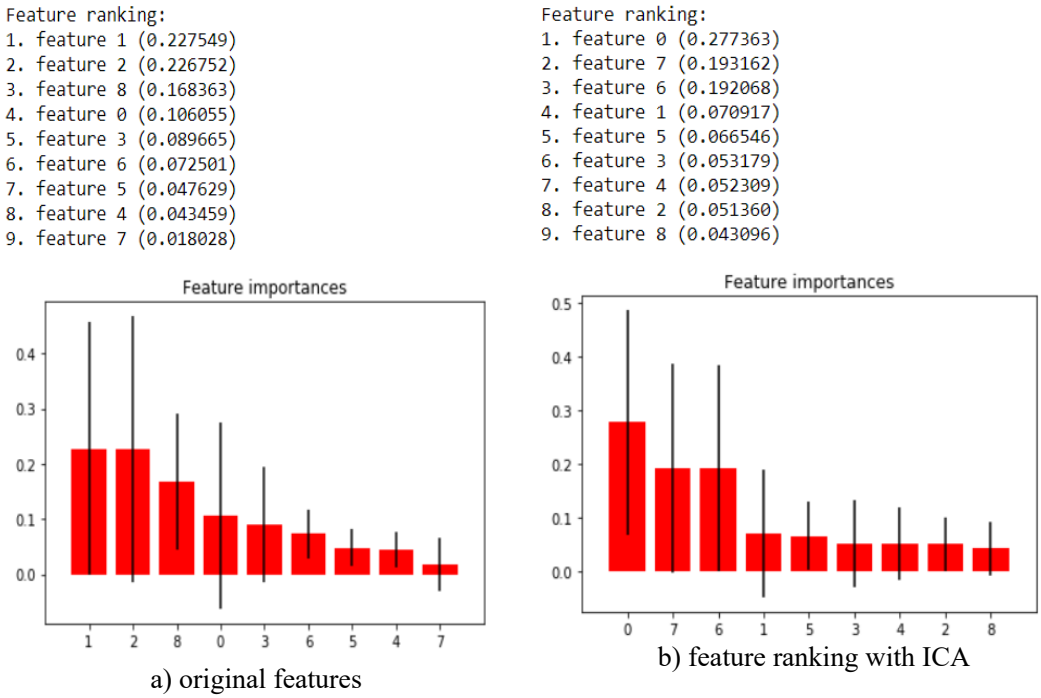
$$S = A^{-1}X \tag{9}$$

$$S = W\,X \tag{10}$$

C) Features Importance Ranking

We evaluated our feature selection through Features Importance Ranking calculated using Extra Tree Classifier, as shown in Algorithm 2. Extra Trees Classifier for features importance in *Scikit*.learn module is based on impurity-based importance where it calculated the importance of training data without reflecting the prediction ability.

From the feature's importance ranking in Fig. **2**, we can see the percentage of the highest contribute features. For example, the 1st feature increased from 22.75% to 27.74% and the lowest contributing feature, the 9th feature, increased from 1.8% to 4.31%. Since no feature had a 0% contribution, the removal process was not executed.



a) original features b) feature ranking with ICA

Fig. 2: The comparison of features ranking with and without ICA.

https://doi.org/10.31436/iiumej.v23i1.1789

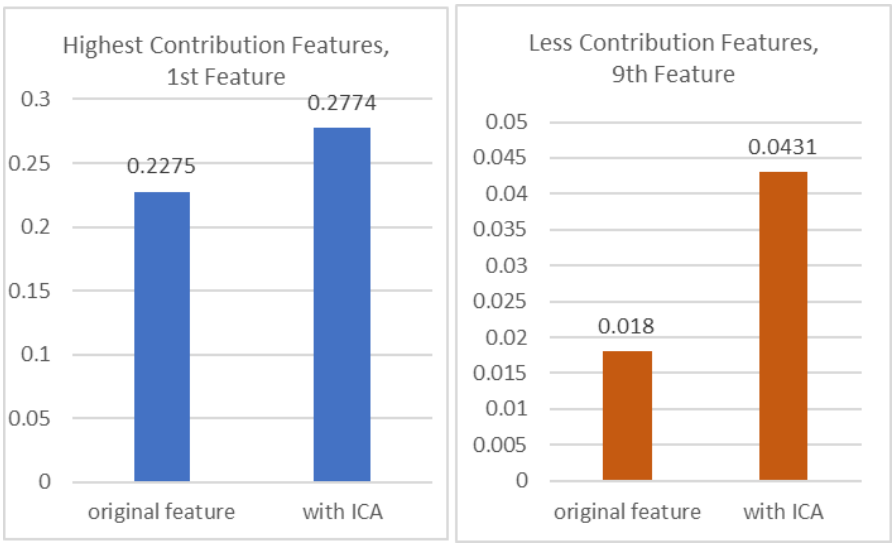


Fig. 3: Percentage differences in the highest and lowest ranking features.

3.3 Building Model using Classification

Once the data completed pre-processing, we moved to the classification process. The classifier that we chose is K-Nearest Neighbor (K-NN). K - Nearest Neighbor (KNN) is one of the most straightforward classification machines that stores all available cases and classifies new cases based on a similarity measure [2,41]. The idea behind KNN is that if a sample belongs to a specific class in the space of several similar samples (k), the sample is also in the category. Thus, techniques based on Nearest-Neighbor classify samples based on the similarity of the population. KNN falls into the algorithm family of supervised learning. Informally, this means that a labeled data set consisting of training observations (x, y) is provided, and the relationship between x and y wants to be captured. More formally, our objective was to learn a function h:X→Y to predict the corresponding y output confidently with an unseen observation x. First, we needed to determine the k-value of the number of groups (cluster) to use K-NN. For this research, we used Elbow Method to determine the k value.

3.3.1 Determine k-value (Elbow Method)

Elbow method ran the k-NN algorithm several times and calculated the WSS error for different values of k. To find the optimal value of k, we used the elbow method that derives from the Within-cluster Sum of Squared (WSS). The Elbow method is a heuristic approach in determining the number of clusters for k-means or k-NN. The equation of WSS is described by Eq. (11):

$$WSS = \sum_{k=1}^{k} \sum_{X_i \in C_k} (X_i - \mu_k)^2 \quad (11)$$

where;
Ck = cluster of k
μk = the mean value of the data that point to the cluster
Xi = an observation to the Ck

The optimal value of k is at the elbow curve, or the distortion point that starts decreasing linearly, as described in Fig. 6. Although the value of k = 2 looks like there is a curve/ distortion from k= 2 until the k = 4, the decrease is still significant and not linear. Due to that, for this research, the k-value was k = 4.

https://doi.org/10.31436/iiumej.v23i1.1789

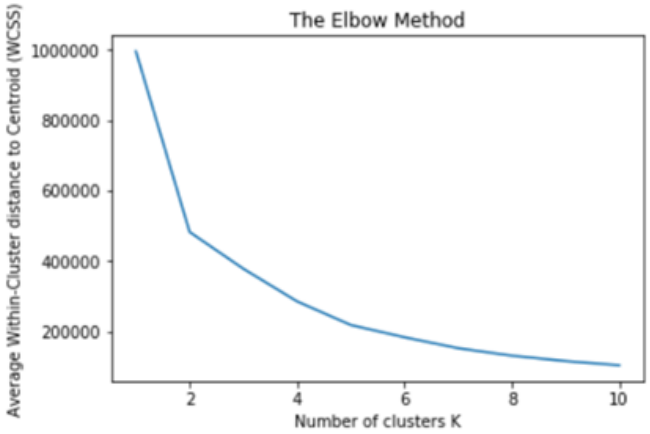


Fig. 4: The Elbow method.

3.3.2 Building the Model

Once we have determined the k-value for K-NN, we started training and building the machine learning model. The Building Model Data was divided into training data and testing data with a 70:30 ratio, as explained in Section 3.2. During the building model process, once the model was built, the prediction of the Evaluation Data was ready to start. We tested it file-by-file to evaluate how well the model could predict a particular bot.

4. EVALUATION

We evaluated the performance of our techniques based on the Confusion Matrix in terms of accuracy, precision, recall, f-score, false-negative rate (FNR), and false-positive rate (FPR). A confusion matrix is the most widely used method to evaluate a machine-learning model's performance. The distribution of the results can be seen clearly by creating a confusion matrix from the model. The confusion matrix consisted of a two-dimensional table with the class "actual" and "cluster/projection" in a single-dimension structure and evaluated only two (2) classes. The other dimension was rated as "Botnet" positive and "Human" negative. Thus, the cases were classified into four fractions: False Positive (FP), False Negative (FN), True Positive (TP), and True Negative (TN), as shown in Table 5.

Table 5: The Confusion Matrix for this article

		Prediction (Cluster)	
		Normal	**Botnet**
Actual (Label)	**Normal**	TN	FP
	Botnet	FN	TP

When the data is in the state “True”, either TP or TN, it shows that the classifier predicted it in the correct class. In the “False” state, there was an incorrect prediction class. For example, when the data was in a False Negative state, it means that the classifier Falsely predicted as Negative (Normal) where the data was positive (botnet), while the Positive and Negative indicate Botnet or Normal class.

https://doi.org/10.31436/iiumej.v23i1.1789

Table 6: The explanation of Confusion Matrix's fraction used in this article

Fraction	Module 2
True Positive (TP)	TP is counted when the model **CORRECT**ly predicted the botnet traffic/IP
True Negative (TN)	TN is counted when the model **CORRECT**ly predicted the normal (Negative) traffic/IP
False Positive (FP)	FP is counted when the model **INCORRECT**ly predicted the normal (Negative) traffic as the **BOTNET** (Positive) traffic/IP
False Negative (FN)	FN is counted when the model **INCORRECT**ly predicted the botnet (Positive) as the normal (**NEGATIVE**) traffic/IP

The Confusion Matrix can generate several performance evaluation parameters, but for this research, we focused on accuracy, precision, recall, f-score, false-negative rate (FNR), and false-positive rate (FPR). These parameters were chosen to make a comparison with other researcher's results that used the same dataset. The overall performance was from the Accuracy, but we preferred to compare the overall performance using the f-score.

4.1 Performance Parameter

4.1.1 Accuracy

Accuracy is often used to measure the overall performance of the machine learning classifier because it is a parameter that measures how often the algorithm correctly classifies a data point. Accuracy is the number of correctly predicted data points from all data points where it can be described in the Eq. (12) below:-

$$Accuracy = \frac{\sum total\ correct\ prediction}{\sum data} = \frac{TP+TN}{TP+TN+FN+FP} \tag{12}$$

4.1.2 Precision

The 'Precision' parameter is the count of data classified as a botnet (positive) that are genuinely botnet. Precision also can be described as in equation (13):-

$$Precision = \frac{Positive\ prediction\ that\ truly\ positive}{\sum Total\ positive\ prediction} = \frac{TP}{TP+FP} \tag{13}$$

4.1.3 Recall

A recall is also known as Sensitivity, where it is the fraction of actual positives that are identified correctly. Recall also can be described as the ability of a model to find the relevant cases.

$$Recall\ (TPR) = \frac{Positive\ prediction\ that\ truly\ positive}{All\ botnet\ data} = \frac{TP}{TP+FN} \tag{14}$$

4.1.4 F-score

F1 score is the harmonic combination of recall and precision. F1 score is the equal weight.

$$F_{score} = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{15}$$

https://doi.org/10.31436/iiumej.v23i1.1789

5. RESULTS AND DISCUSSION

The evaluation for the framework was conducted in 2 parts; one was when building the model and the other one was the prediction of novel data using the Evaluating Data.

5.1 Building Model

This section summarizes the findings during the Model Building. The performance of the flow-based features and K-NN framework was compared to the performance of the same framework but with additional ICA during pre-processing. Both performance evaluations are shown in Table 7.

Table 7: The performance comparison of the training model with and without ICA

	Accuracy	Precision	Recall	F1-score	FNR	FPR
without ICA	0.9689	0.9831	0.958	0.9704	0.0419	0.01875
with ICA	0.969	0.9846	0.9568	0.9705	0.0431	0.0169

This result was evaluated on the testing data split from the Building Model Data illustrated in Fig. 3. Table 7 lists the parameters used in the evaluation. From the results in Table 7, if we compare the Accuracy and F-score, there are tiny increments, but the False Positive Rate (FPR) decreased from 1.88% to 1.69%.

5.2 Prediction Novel Bots

CTU-13, the benchmark botnet dataset, is also used by several researchers to evaluate their prediction model's performance. So, we compared the result from another article that used the same data source and the same evaluation parameter. There were 5 files in Evaluating Data that were separated before the building model process. The data is described in Table 8.

Table 8: The data number and the botnet type in Evaluating Data

Data Number	Bot Name
data 1	Neris
data 2	
data 6	Menti
data 8	Murlo
data 9	Neris

Table 9: Performance differences between models with and without ICA against novel botnet

K-NN Model	Data Number	Accuracy	F1-score	FPR
without ICA	data 1	0.6708	0.6895	0.2187
	data 2	0.7583	0.784	0.0679
	data 6	0.7032	0.6041	0.1826
	data 8	0.7658	0.2871	0.1547
	data 9	0.5067	0.6436	0.1891
with ICA	data 1	0.7912	0.8298	0.278
	data 2	0.6734	0.6958	0.1262
	data 6	0.7767	0.7673	0.2995
	data 8	0.8005	0.5859	0.2244
	data 9	0.7134	0.8222	0.3309

Based on Table 9, the model with ICA showed a better F score than the Model Without ICA. But, the model without ICA showed the lowest FPR compared to the other model. Thus, the yellow box in Table 9 indicates the best value, either the highest F score or the lowest FPR. Since we were focused on the F-score for comparison, we extracted it to Table 10.

Table 10: The F score for both models (with and without ICA)

	data 1	**data 2**	**data 6**	**data 8**	**data 9**
without ICA	0.6895	0.784	0.6041	0.2871	0.6436
with ICA	0.8298	0.6958	0.7673	0.5859	0.8222

Table 10 shows that 4 files from 5 novel botnet files had the highest F-score using the K-NN model with ICA. Only one file, File No 2, showed the opposite result. Due to that, we agree that the K-NN Model with ICA performed better than the K-NN Model without ICA. The result for the K-NN Model with ICA was compared to other researcher's results. The results were directly compared with previously reported findings on a novel botnet prediction model that used the same data sources. Table 11 summarizes the comparison result and lists the parameters used in every previous article.

Table 11: Performance model comparison between previous researchers

Parameter	**Data No/ researcher**	**Garcia, S., Zunino, A., & Campo, M. (2014)[42]**	**Garcia, S. (2015). [44]**	**Kozik, R. (2018) [45]**	**Wang, J. & Paschalidis, I.C. (2017)[46]**	**Fernandez Maimo et al. [38]**	**wnh ibrahim et al.**
Recall	1	0.4	1	0.077	0.91	0.91	0.8357
	2	0.3	0.88	0.046	0.55	0.95	0.5687
	6	0	0	0.12	0.95	1	0.8836
	8	0.1	0	0.044	0.76	0.26	0.9424
	9	0.1	0.38	0.08	0.38	0.99	0.7171
Precision	1	0.5	0.87	0.86	0.73	0.68	0.824
	2	0.6	0.96	0.8	0.65	0.88	0.8961
	6	0.4	-1	0.69	0.7	0.92	0.6781
	8	0.2	0	0.6	0.4	0.47	0.4251
	9	0.4	0.72	0.73	0.95	0.89	0.9635
F1 Score	1	0.48	0.93	0.14	0.81	0.77	0.8298
	2	0.41	0.92	0.088	0.59	0.92	0.6958
	6	0.04	0	0.21	0.8	0.96	0.7673
	8	0.14	-1	0.082	0.53	0.33	0.5859
	9	0.25	0.5	0.14	0.54	0.94	0.8222

Based on Table 11, in the F-Score parameter, our technique defeated other results for Data number 8 (red font). However, three out of five novel botnet files hade the highest f-score from Fernandez Maimo et al. [31]. To measure the overall performance, we calculated the average for each parameter. The average of each data (Data from File 1,2,6,8,9) and parameter (Precision, Recall, and F Score) from Table 11 are illustrated in Fig. 7. These plots show that our method proposed here outperformed the other approaches except Fernandez Maimo et al. [31].

Since the Fernandez Maimo et al. [31] techniques outperformed the overall evaluation, we compared the different approaches they used. For example, Fernandez Maimo et al. [31] also used flow-based features, but they considered the features from dual-direction, incoming, and outgoing traffic. Their approach resembled our technique in that both methods focused on concealment network traffic and used statistical analysis to aggregate the data.

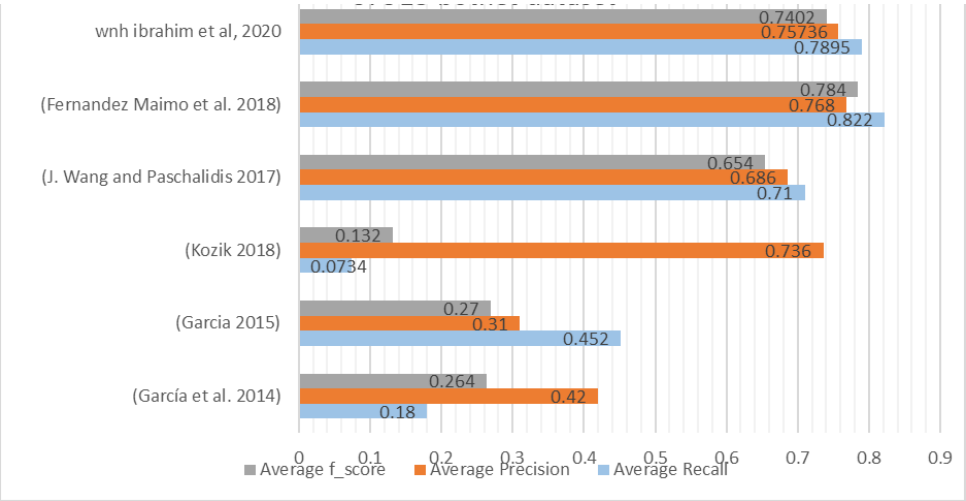


Fig. 5: The performance comparison among other researchers on botnet classification.

6. CONCLUSION

This paper proposes the framework for novel botnet detection that implements data standardization and Independent Component Analysis (ICA) during flow-based features pre-processing data. The strength of our framework is that we used flow-based features that have the benefits of detecting traffic from the concealment network. Other than that, the complexity and processing time can also be minimized by flow-based features compared to content-based ones. Also, for aggregation, for the quick detection method, we used the shortest time interval. Our approach can be applied to botnet concealment and new/novel forms of a botnet.

The use of data standardization and Independent Component Analysis (ICA) improves key rating attributes and classification outcomes. However, the overall result is still not the best relative to other previous approaches. Nevertheless, it generated an improved result using Data Standardization and Independent Component Analysis (ICA). We can also assume that behavior analysis caused some noise to the pattern.

Future directions are connected to enhancing the collection of functionalities. Further developments are expected to lead to a deeper understanding of the nature of the selection of functions.

ACKNOWLEDGEMENT

The authors wish to thank Universiti Teknologi Malaysia (UTM) for its support under Research University Grant Vot-20H04, Malaysia Research University Network (MRUN) Vot 4L876, and the Fundamental Research Grant Scheme (FRGS) Vot (FRGS/1/2018/ICT04/UTM/01/1) supported by the Ministry of Higher Education Malaysia. The work is partially supported by the SPEV project (ID: 2102-2021), Faculty of Informatics and Management, University of Hradec Kralove. We are also grateful for the

support of Ph.D. students Michal Dobrovolny and Sebastien Mambou in consultations regarding application aspects from Hradec Kralove University, Czech Republic. The APC was funded by the SPEV project 2102/2021, Faculty of Informatics and Management, University of Hradec Kralove.

REFERENCES

[1] Ibrahim, W.N.H., Selamat, A., Anuar, S., & Krejcar, O. (2019). Clustering botnet behavior using K-means with uncertain data, Frontiers in Artificial Intelligence and Applications vol. 318. pp.244–257.
[2] Liang, X. & Znati, T. (2019). On the performance of intelligent techniques for intensive and stealthy DDos detection, Computer Networks vol. 164. p.106906.
[3] Gross, G. (2016). Detecting and destroying botnets, Network Security vol. 2016, no. 3. pp.7–10.
[4] WHITE OPS. (2018). Retrieved June 1, 2020, https://www.whiteops.com/blog/9-of-the-most-notable-botnets.
[5] Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DDoS in the IoT: Mirai and other botnets, Computer vol. 50, no. 7. pp.80–84.
[6] Eremin, A. (2019). Retrieved June 1, 2020, https://securelist.com/bots-and-botnets-in-2018/90091/.
[7] Khan, R.U., Zhang, X., Kumar, R., Sharif, A., Golilarz, N.A., & Alazab, M. (2019). An adaptive multi-layer botnet detection technique using machine learning classifiers, Applied Sciences (Switzerland) vol. 9, no. 11. p.2375.
[8] Patsakis, C., Casino, F., & Katos, V. (2020). Encrypted and covert DNS queries for botnets: Challenges and countermeasures, Computers & Security vol. 88. p.101614.
[9] Bezerra, V.H., da Costa, V.G.T., Barbon Junior, S., Miani, R.S., & Zarpelão, B.B. (2019). IoTDS: A one-class classification approach to detect botnets in internet of things devices, Sensors (Switzerland) vol. 19, no. 14. p.3188.
[10] Wang, Y.-H., Li, Z.-N., Xu, J.-W., Yu, P., Chen, T., & Ma, X.-X. (2020). Predicted Robustness as {QoS} for Deep Neural Network Models, Journal of Computer Science and Technology vol. 35, no. 5. pp.999–1015.
[11] Prasad, K.M., Reddy, A.R.M., & Rao, K.V. (2020). BARTD: Bio-inspired anomaly based real time detection of under rated App-DDoS attack on web, Journal of King Saud University - Computer and Information Sciences vol. 32, no. 1. pp.73–87.
[12] Su, S.C., Chen, Y.R., Tsai, S.C., & Lin, Y.B. (2018). Detecting P2P Botnet in Software Defined Networks, Security and Communication Networks vol. 2018. pp.1–13.
[13] Mahmoud, M., Nir, M., & Matrawy, A. (2015). A Survey on botnet architectures, detection and defences, International Journal of Network Security vol. 17, no. 3. pp.272–289.
[14] Mathur, L., Raheja, M., & Ahlawat, P. (2018). Botnet Detection via mining of network traffic flow, Procedia Computer Science vol. 132. pp.1668–1677.
[15] Kupreev, O., Badovskaya, E., & Gutnikov, A. (2019). Retrieved June 1, 2020, https://securelist.com/ddos-report-q3-2019/94958/.
[16] Aamir, M. & Zaidi, S.M.A. (2019). Clustering based semi-supervised machine learning for DDoS attack classification, Journal of King Saud University - Computer and Information Sciences.
[17] Singh, M., Singh, M., & Kaur, S. (2019a). Detecting bot-infected machines using DNS fingerprinting, Digital Investigation vol. 28. pp.14–33.
[18] Bazrafshan, Z., Hashemi, H., Fard, S.M.H., & Hamzeh, A. (2013). A survey on heuristic malware detection techniques, IKT 2013 - 2013 5th Conference on Information and Knowledge Technology no. May. pp.113–120.
[19] AsSadhan, B., Bashaiwth, A., Al-Muhtadi, J., & Alshebeili, S. (2018). Analysis of P2P, IRC and HTTP traffic for botnets detection, Peer-to-Peer Networking and Applications vol. 11, no. 5. pp.848–861.

[20] Alauthaman, M., Aslam, N., Zhang, L., Alasem, R., & Hossain, M.A. (2018). A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks, Neural Computing and Applications vol. 29, no. 11. pp.991–1004.
[21] Santana, D., Suthaharan, S., & Mohanty, S. (2018). What we learn from learning - Understanding capabilities and limitations of machine learning in botnet attacks.
[22] Rauf, M.A.A.A., Asraf, S.M.H., & Idrus, S.Z.S. (2020). Malware Behaviour Analysis and Classification via Windows DLL and System Call, Journal of Physics: Conference Series vol. 1529, no. 2.
[23] Resende, P.A.A. & Drummond, A.C. (2018). A survey of random forest based methods for intrusion detection systems, ACM Computing Surveys vol. 51, no. 3. pp.1–36.
[24] Tobiyama, S., Yamaguchi, Y., Shimada, H., Ikuse, T., & Yagi, T. (2016). Malware Detection with Deep Neural Network Using Process Behavior, Proceedings - International Computer Software and Applications Conference vol. 2. pp.577–582.
[25] Muhtadi, A.F. & Almaarif, A. (2020). Analysis of Malware Impact on Network Traffic using Behavior-based Detection Technique, International Journal of Advances in Data and Information Systems vol. 1, no. 1. pp.17–25.
[26] Apruzzese, G. & Colajanni, M. (November 2018). Evading botnet detectors based on flows and random forest with adversarial samples. Paper presented at NCA 2018 - 2018 IEEE 17th International Symposium on Network Computing and Applications.
[27] Beigi, E.B., Jazi, H.H., Stakhanova, N., & Ghorbani, A.A. (2014). Towards effective feature selection in machine learning-based botnet detection approaches, 2014 IEEE Conference on Communications and Network Security, CNS 2014 pp.247–255.
[28] Cabeza, L.F., Solé, C., Castell, A., Oró, E., & Gil, A. (2016). Unsupervised Network Intrusion Detection Systems for Zero-Day Fast- Spreading Attacks and Botnets Payam, International Journal of Digital Content Technology and its Applications(JDCTA) vol. 10, no. 2.
[29] Singh, M., Singh, M., & Kaur, S. (2019b). Detecting bot-infected machines using DNS fingerprinting, Digital Investigation vol. 28. pp.14–33.
[30] Malik, R. & Alankar, B. (2019). Botnet and Botnet Detection Techniques, International Journal of Computer Applications vol. 178, no. 17. pp.8–11.
[31] Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). 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. pp.779–796.
[32] Huda, S., Abawajy, J., Al-Rubaie, B., Pan, L., & Hassan, M.M. (2019). Automatic extraction and integration of behavioural indicators of malware for protection of cyber–physical networks, Future Generation Computer Systems vol. 101. pp.1247–1258.
[33] Bou-Harb, E., Debbabi, M., & Assi, C. (2014). On fingerprinting probing activities, Computers and Security vol. 43, no. January 2012. pp.35–48.
[34] Palmieri, F., Fiore, U., & Castiglione, A. (2014). A distributed approach to network anomaly detection based on independent component analysis, Concurrency and Computation: Practice and Experience vol. 26, no. 5. pp.1113–1129.
[35] Mao, C.-H., Lin, C.-C., Pan, J.-Y. (Tim), Chang, K.-C., Faloutsos, C., & Lee, H.-M. (2012). EigenBot: Foiling spamming botnets with matrix algebra. Paper presented at Proceedings of the ACM SIGKDD Workshop on Intelligence and Security Informatics - ISI-KDD '12, New York, New York, USA.
[36] Ehsan, K. & Hamid, reza shahriari. (2018). BotRevealer: Behavioral Detection of Botnets based on Botnet Life-cycle, International Journal of Information Security vol. 10, no. 1. pp.55–61.
[37] Stevanovic, M. & Pedersen, J.M. (2015). On the use of machine learning for identifying botnet network traffic, Journal of Cyber Security and Mobility vol. 4, nos. 2–3. pp.1–32.
[38] Fernandez Maimo, L., Perales Gomez, A.L., Garcia Clemente, F.J., Gil Perez, M., & Martinez Perez, G. (2018). A Self-Adaptive Deep Learning-Based System for Anomaly Detection in 5G Networks, IEEE Access vol. 6. pp.7700–7712.
[39] Debashi, M. & Vickers, P. (2018). Sonification of Network Traffic for Detecting and Learning about Botnet Behavior, IEEE Access vol. 6. pp.33826–33839.

[40] Gezer, A., Warner, G., Wilson, C., & Shrestha, P. (2019). A flow-based approach for Trickbot banking trojan detection, Computers and Security vol. 84. pp.179–192.
[41] Garg, S., Peddoju, S.K., & Sarje, A.K. (2016). Scalable P2P bot detection system based on network data stream, Peer-to-Peer Networking and Applications vol. 9, no. 6. pp.1209–1225.
[42] Garcia, S., Zunino, A., & Campo, M. (2014). "Identifying, Modeling and Detecting Botnet Behaviors in the Network".
[43] Han, W., Xue, J., Wang, Y., Liu, Z., & Kong, Z. (2019). MalInsight: A systematic profiling based malware detection framework, Journal of Network and Computer Applications vol. 125, no. October 2018. pp.236–250.
[44] Garcia, S. (2015). Modelling the Network Behaviour of Malware To Block Malicious Patterns . The Stratosphere Project : a Behavioural Ips, Virus Bulletin no. September. pp.1–8.
[45] Kozik, R. (2018). Distributing extreme learning machines with Apache Spark for NetFlow-based malware activity detection, Pattern Recognition Letters vol. 101. pp.14–20.
[46] Wang, J. & Paschalidis, I.C. (2017). Botnet Detection Based on Anomaly and Community Detection, IEEE Transactions on Control of Network Systems vol. 4, no. 2. pp.392–404.