IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 17, 2021, accepted February 8, 2021, date of publication February 22, 2021, date of current version April 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3060778

Multilayer Framework for Botnet Detection Using Machine Learning Algorithms

WAN NUR HIDAYAH IBRAHIM[1], (Member, IEEE), SYAHID ANUAR[2], ALI SELAMAT[1,3,4], (Member, IEEE), ONDREJ KREJCAR[4], RUBÉN GONZÁLEZ CRESPO[5], (Senior Member, IEEE), ENRIQUE HERRERA-VIEDMA[6,7], (Fellow, IEEE), AND HAMIDO FUJITA[8], (Life Senior Member, IEEE)

[1]School of Computing, Faculty of Engineering, Game Innovation Centre of Excellence (MaGICX), Universiti Teknologi Malaysia and Media, Universiti Teknologi Malaysia, Johor Baharu 81310, Malaysia
[2]Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Kuala Lumpur 54100, Malaysia
[3]Malaysia Japan International Institute of Technology (MJIIT), Universiti Teknologi Malaysia, Kuala Lumpur 54100, Malaysia
[4]Center for Basic and Applied Research, Faculty of Informatics and Management, University of Hradec Kralove, 500 03 Hradec Kralove, Czech Republic
[5]Department of Computer Science and Technology, Universidad Internacional de La Rioja (UNIR), 26006 Logroño, Spain
[6]Andalusian Research Institute DaSCI Data Science and Computational Intelligence, University of Granada, 18071 Granada, Spain
[7]Department of Electrical and Computer Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[8]Faculty of Software and Information Science, Iwate Prefectural University, 152-52 Sugo, Takizawa 020-0693, Iwate, Japan

Corresponding author: Ali Selamat (aselamat@utm.my)

This work was supported in part by Universiti Teknologi Malaysia (UTM) through the Research University Grant under Grant Vot-20H04, in part by the Malaysia Research University Network (MRUN) under Grant Vot4L876, in part by the Ministry of Higher Education through the Fundamental Research Grant Scheme under Grant FRGS/1/2018/ICT04/UTM/01/1, in part by the Specific Research Project (SPEV) by the Faculty of Informatics and Management, University of Hradec Kralove, Czech Republic, under Grant 2102–2021, and in part by the Hadiah Latihan Persekutuan (HLP) Scholarship through the Ministry of Education Malaysia.

ABSTRACT A botnet is a malware program that a hacker remotely controls called a botmaster. Botnet can perform massive cyber-attacks such as DDOS, SPAM, click-fraud, information, and identity stealing. The botnet also can avoid being detected by a security system. The traditional method of detecting botnets commonly used signature-based analysis unable to detect unseen botnets. The behavior-based analysis seems like a promising solution to the current trends of botnets that keep evolving. This paper proposes a multilayer framework for botnet detection using machine learning algorithms that consist of a filtering module and classification module to detect the botnet's command and control server. We highlighted several criteria for our framework, such as it must be structure-independent, protocol-independent, and able to detect botnet in encapsulated technique. We used behavior-based analysis through flow-based features that analyzed the packet header by aggregating it to a 1-s time. This type of analysis enables detection if the packet is encapsulated, such as using a VPN tunnel. We also extend the experiment using different time intervals, but a 1-s time interval shows the most impressive results. The result shows that our botnet detection method can detect up to 92% of the f-score, and the lowest false-negative rate was 1.5%.

INDEX TERMS Behavior-based analysis, botnet, flow-based feature selection, k-nearest neighbor, structure independent.

I. INTRODUCTION

Botnet is a term referring to infected devices that a hacker remotely controls called a ***botmaster***. The term botnet is a combination of robot and network, where the botnet acts as a foot soldier for its botmaster. The task of the botnet is to launch attacks based on the instructions given by its botmaster.

Botnet attacks are a serious issue and have become a significant threat to information security [1], [2]. The arms races between botmasters and botnet defenders (researchers) are ongoing. Each party keeps improving its skills to try to win the battle. The botnet's strength lies in the massive number of bots, which increases the strength of attacks. Also, botmasters' ability to hide the bots from detection by a security system becomes a significant factor strengthening the bots. One of the most popular botnets that shocked the world with the number of infected devices is the

The associate editor coordinating the review of this manuscript and approving it for publication was Aniello Castiglione.

TABLE 1. Comparison of the different detection model.

Detection Model	Obfuscation (Encryption)	Protocol-Independent	Structure-Independent	Time Interval
Zhuang and Chang [17]	√		P2P only	
AsSadhan et al. [35]		TCP only	IRC and HTTP	
Alauthaman et al. [37]		TCP only	P2P only	
Maimó et al. [38]		TCP/UDP		30–60 s
Koli and Chavan [39]	√	TCP/UDP	Only mention P2P	21.38 s
Bezerra et al. [36]	Focus on detecting botnet by using host-based features such as device's CPU utilization and temperature, memory consumption, and several running tasks.			1 s
Our approach	√	√	√	**1 s**

Mirai Botnet. The Mirai botnet spread through Trojans and exploited Internet-of-Things (IoT) devices such as closed-circuit television cameras (CCTV), web cameras, and other devices with low-security measures. The most significant Mirai attack involved 100,000 IoT devices that caused an attack of 1.2 Tbps [3].

The existing botnet detection methods are signature-based approaches that do well at detecting the same types of botnets or known botnets but become ineffective when faced with an unknown or evolved botnet [4]–[7]. Currently, botnets keep evolving to avoid detection by security systems. One of the strategies is to make sure no one can access the packet data, for example, by using a concealment technique such as encryption, obfuscation, or a virtual private network.

The limitation of signature-based detection, as stated in [8], and network-based IDS, as stated in [9], is that the current detection models are unable to detect malware when there are obfuscation techniques in use. Hence, researchers are moving forward to design a malware detection model without accessing the packet's content.

Other than that, the packet's content that may cause harm to individuals is the reason for the limited updated attacks dataset for research. One of the methods for analyzing network traffic without accessing the content is through behavior-based analysis. The behavior-based analysis uses the packet header instead of the payload not to interrupt the privacy of sensitive content in the packet data. The behavior-based analysis within the network traffic has the advantage of detecting malware with an encryption or obfuscation strategy such as a VPN. However, behavior-based malware detection commonly produces a high false-positive rate (FPR) [6], [10], [11] and an increased scanning time (time interval).

Due to the limitation of the signature-based analysis and the potential of improvement in several research areas on malware behavior [7], [12]–[15], we designed our detection model based on the behavior-based analysis. This research examines the features useful for creating a behavior-based analysis method for detecting botnets in network traffic that quickly produces good results. The main contributions of this research are as follows:

- This article presents the multilayer framework that can detect the Command and Control (C&C) server's botnet in hiding techniques such as obfuscation or encryption for both layers.
- Our works highlight the criteria of structure-independent and protocol-independent frameworks.
- Other than the framework's performance, our work also presents a short time interval (1 s) for aggregating the botnet behavior for both layers.
- The first layer of this framework is for filtering regular traffic. This layer can reduce the processing time and power by selecting suspicious groups for the second phase.
- The accuracy of both layers is more than 90%, and the false-negative rate is less than 2.5%.

The structure-independent and protocol-independent frameworks (second contribution) are based on [10], [16] where the analysis is not limited to a particular protocol and specific structure. Since that botnet is very flexible and evolves through multiple protocols and structures, this criterion is also included in designing the detection model. The highlight of these criteria can be seen in Section 4.1 and TABLE 1. In Section 4.1, we briefly explain the dataset that we used in TABLE 1. We make a comparison of these two criteria with another researcher's approach.

This work is organized as follows: we explain the botnet and related works in Section 2, including the current botnet behavior analysis in Section 2.1 and machine learning and oversampling technique in Section 2.2. Section 3 briefly explains the proposed framework, while Section 4 describes the experiment starting with data source and distribution, the evaluation and the result. The article ends with a discussion and conclusion in Section 5.

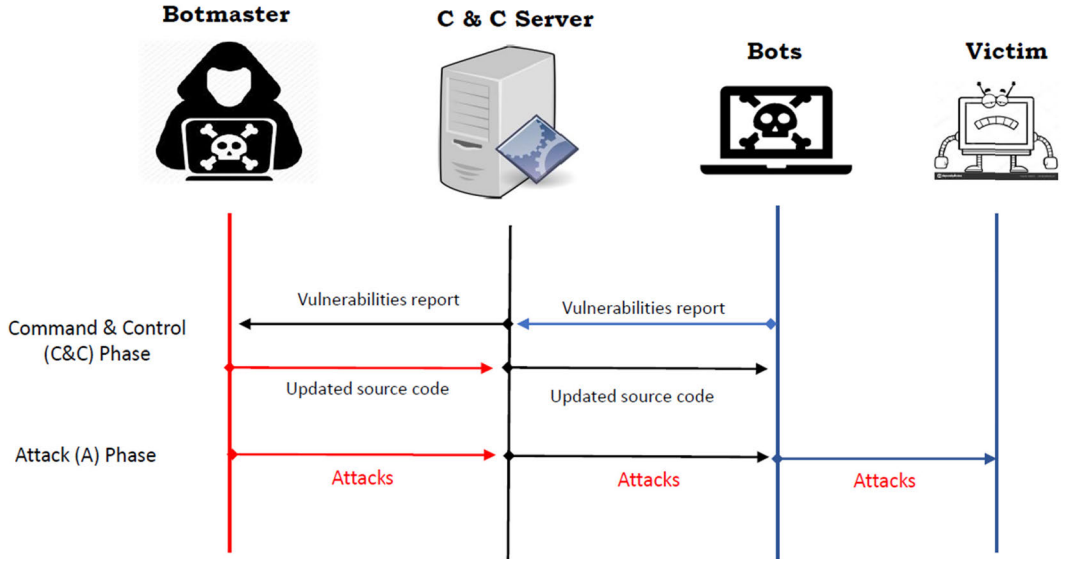


FIGURE 1. Botnet component and communication between component.

II. RELATED WORKS

A. TERMS AND DEFINITION

There are several terms used in the whole article that are not layman's terms. This first section will briefly give definitions of these terms.

- *BOTMASTER* This term refers to the mastermind that owns, instructs, and is responsible for launching the attacks. S/he is also the person that will keep communicating with the bots through the Command-and-Control server
- *BOTNET.* It is a group of infected devices that will send reports on the device and system vulnerabilities and exploit the information to perform attacks.
- *COMMAND & CONTROL (C&C) SERVER.* This term refers to the medium that acts as the bridge between the botmaster and the botnet. This C&C server is the main component in the botnet environment because, without the C&C server, the botmaster cannot control or send instructions to the bots. The structure of this server can be either centralized or decentralized.
- *STRUCTURE-INDEPENDENT & PROTOCOL-INDEPENDENT.*
  Structure-independent is a term that referring to the dataset that contains multiple structures. For this study, structure-independent means that the dataset consists of a centralized structure and a decentralized structure. In comparison, protocol-independent refers to the dataset containing multiple types of protocols such as IRC, HTTP, and P2P.

B. BOTNET COMPONENT AND LIFE-CYCLE

The botnet consists of four main components: the bots, botmaster, command and control (C&C) server, and the victims/target, as shown in FIGURE 1. To make it easier to understand, we can imagine the bots as soldiers in a troop (botnet) following the general's commands (botmaster) from afar, where the commands are transferred through a Command-and-Control Server.

The basic botnet life cycle contains four phases, as illustrated in II. The first phase is the Injection (I) phase. The injection phase is a spreading phase. There are many spreading methods, such as through drive-by-download, email, web-based, and online social media networks. In this phase, the hacker will maximize the number of army or bots by infecting other devices. Once the bots are downloaded and executed, the device/host becomes a bot and can be controlled by the botmaster.

The second phase is the Command and Control (C&C) phase, the phase we are currently studying. In this phase, the botmaster secures the botnet by requesting an information report, and the botnet will send an updated vulnerability report on the infected device. The botmaster communicates with the bots through the Command-and-Control Server to either direct an attack, receive a report, or send updated codes, as illustrated in FIGURE 1. This is the secret of how the botnet is robust and unable to be detected. This is also why the botnet has unique abilities to discover unknown devices' vulnerabilities and evolve autonomously [20], [21]. During the Command-and-Control phase, there is a situation where there is no communication between the bots and botmaster. This situation is called the *waiting stage* and happens either because the botmaster is still gathering the bots, or the attack time is not suitable yet. This situation makes it quite tricky to detect the bots, and it becomes a new criterion for the researcher.

The third phase is the Attack (A) phase. Once the bots' quantity is large enough to launch an attack, the botmaster's instruction will be sent to all the bots. Each of the bots will aim at the same victim. For example, in the DDOS (Distributed Denial of Service) attack in February 2018, a massive

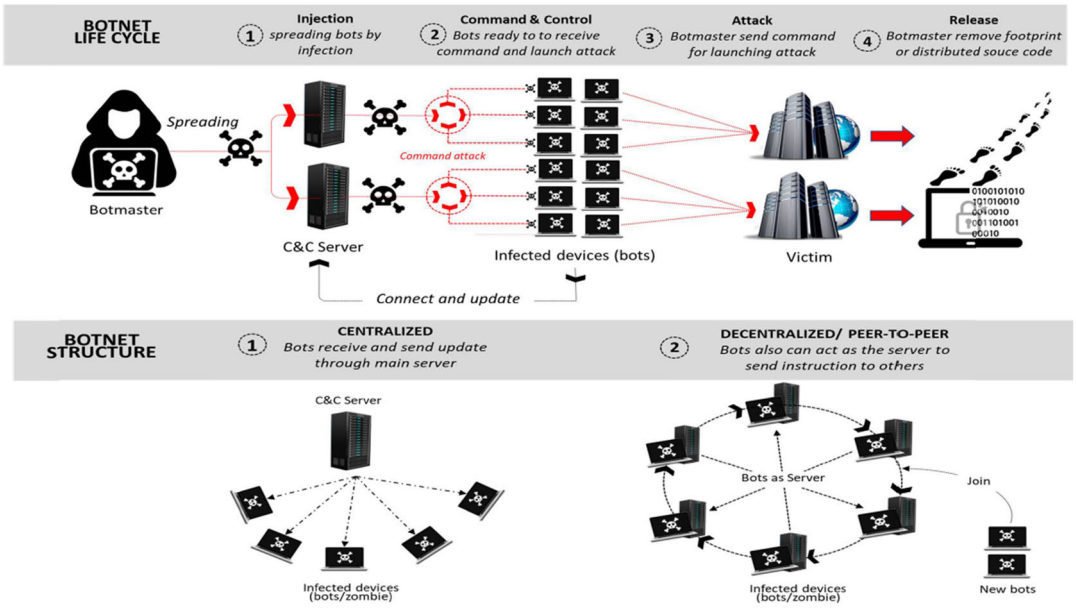


FIGURE 2. The life cycle and structure of the botnet.

botnet flooded the network by sending simultaneous requests (peaked at 1.35 Tbps) to the same target, GitHub; due to that, the GitHub service was offline for 10 min [19]. The most significant DDOS attack launched by bots was the Mirai attack in October 2016. Hundreds of websites such as Twitter, Netflix, Reddit, and GitHub were affected several hours when service provider Dyn has attacked 400,000 IoT devices as bots [19], [20].

The last phase is the Release (R) phase. In this phase, the botmaster decides to leave the bots because s/he is not needed or avoided by the authorities. Some botmasters decide to release their bot's source code to the public and remove their footprints [15] to confuse the authorities searching for the person responsible for the attacks. For example, the botnet's source codes were made publicly available in Bashlite and Mirai [21]. The best time for detecting the botnet is when they are in the Command-and-Control phase because, in the infection phase, it can spread in multiple ways. Therefore, it is quite difficult to stop during the infection phase, but it will be too late to stop in the attack phase.

C. CURRENT BOTNET BEHAVIOR ANALYSIS

The unique feature of the botnet is its ability to hide from a security system. A botnet can hide in many ways; for example, as stated below: -

- Concealment packet data. Concealment is a strategy to hide the content of the packet data in network traffic. As mentioned in Section 1, concealment examples include obfuscation, code encryption, oligomorphic strategy, polymorphic strategy, and metamorphic strategy. Research on the botnet detection model that highlights the concealment packet data include studies such as [7], [13], [17], [18].
- Mimicking regular traffic. This can either replicate normal traffic, which is usually more random than that produced by a botnet—research on the botnet detection model highlights mimicking benign behaviors in [14], [19].
- Botnet in the waiting stage. As explained in Section 1.2, the waiting stage is when the devices are already infected and are a part of the bots, but the attack's source code has not been launched yet. So, in this phase, communication between the bots and the botmaster is rare, so bots are quite challenging to detect. Research on the botnet detection model that highlights the waiting stage includes studies such as [14], [20], [21].
- Imbalanced class data. During the machine learning training session, if the class data are highly imbalanced, it will affect the classification. Research highlights the imbalance in studies such as [22], [23].

Due to the bot's hiding ability, an analysis that requires payload data such as deep packet inspection (DPI) cannot effectively function. The behavior-based study seems like a promising solution for detecting malware's current trends because this technique only requires the packets' header. The behavior-based analysis observes the pattern, connection, and action that are captured from the communication between the bots and the botmaster.

The malware behavior-based analysis has advantages compared to signature-based analysis in terms of processing time and power due to the need for examining each packet in the signature-based analysis [24]–[26]. Since behavior-based analysis is not content-based, it can also be implemented with network traffic that uses a VPN tunnel.

In trying to understand the botnet's behavior, we have extracted the frequency of communication-based on time.

TABLE 2. Comparison of features and time window used for detecting a botnet in the network.

Author(s)	Features	Time Window (seconds)
Khoshhalpour and Shahriari [15]	Group duration time, number of packets received, number of sent packets, distance from the previous group	Group Duration Time
Maimó et al. [29]	*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 total flows	20–30 s
Debashi and Vickers[46]	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
Garg *et al*. [47]	Send Syn, Recv ACK, Recv Rst, Send pkts, Recv pkts, ICMP unr, Send Len, Recv Len	120 s

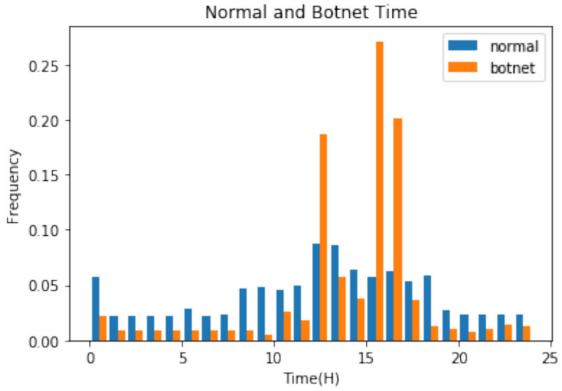


FIGURE 3. Histogram of normal and botnet traffic time in CTU13.

FIGURE 3 shows a histogram of botnet and regular traffic that we have extracted through the combination of 13 files in the CTU-13 dataset. From the histogram, we can see the botnet traffic and the standard traffic curve. The curves show that the highest peak from regular and botnet traffic is in the same range of time. The botmaster used the busiest time for normal traffic to connect with the bots to mimic normal communication. FIGURE 3 shows the bots replicating the peak time of reasonable traffic from 8h to 18h.

Although the malware behavior-based analysis has advantages over the signature-based analysis, most of the behavior-detection model is limited to a particular protocol and specific botnet structure. In TABLE 1, we compare related research on the detection of botnets with the three criteria that we highlight: protocol-independent, structure-independent, and the function of network traffic in situations such as encryption. Zhuang and Chang [14] focused on peer-to-peer application and peer-to-peer botnet only. In [27], the detection model is structure-independent; the authors mixed the types of the botnet, peer-to-peer (P2P), Internet-Relay-Chat (IRC), and Hypertext Transfer Protocol (HTTP), such that the botnet consisted of both centralized and decentralized structures. IRC and HTTP are examples of a botnet in a centralized structure. However, they used their capturing dataset and limited it to TCP protocol only. Other than that, the behavior-based analysis also required a significant time interval to capture the communication pattern effectively. For example, in [25], the author used to extract the periodic pattern was 33.3 min or 49 min. Since we aim to design a detection model in a short time interval, we found an article by Bezerra *et al.* [28] that uses a 1-s time interval. These authors believe that faster botnet identification can be achieved by using a smaller time interval.

However, Bezerra *et al.* [28] did not focus on botnet detection using network traffic; their focus was on botnet detection utilizing the device's CPU utilization and temperature, memory consumption, and several running tasks. The highest F-score for their experiment using a 1-s time interval was 83.85%. We preprocessed the dataset with a 1-s time interval to test botnet network traffic and regular traffic for our experiment.

The most challenging part of designing a behavior-based detection model is the feature selection. It is not straightforward to know which features should be used and how to extract the pattern [29]. Botnet communication is very different from regular human traffic, and the features selected to be aggregated must be representative of it. TABLE 2 shows the features and the observing time window used by researchers in designing the botnet detection model. The features selected by the researchers in TABLE 2 became our reference for choosing our botnet behavior features. The process of feature selection for our experiment is explained in Section 3.1.

Based on [30], the botnet is about malware and the technology of communication between devices. Other *good botnets* use the same technology for communicating, sharing computer resources, and storage, such as the BOINC Project. BOINC (Berkeley Open Infrastructure for Network Computing) is a volunteer project whereby participants share their computer resources and storage to support a specific project in the list [31]. According to the author, the biggest BOINC project is the seti@home project, which has 1,648,000 users and 4,059,000 hosts. In a BOINC project, the participant needs to install the software so the primary server can access their storage and computing resources. The BOINC project and botnet's communication method are quite

TABLE 3. Research on the botnet using machine learning and oversampling.

Author	Classifier	Oversampling	Best Result
Pajouh et al. [32]	Naive Bayes, Support Vector Machine, Multilayer Perceptron	SMOTE	Decision tree-J48 with SMOTE-5x: Accuracy 96.62%, False alarm 4.0%
Alam and Vuong [33]	Bayes Net, J48 decision tree, Logistic Regression, Multilayer Perceptron, Naive Bayes, and Random Forest	SMOTE	RF: Accuracy 99.9% and OOB varies between 0.0002 and 0.0004
Sewak et al. [34]	Random Forest and Deep Neural Network	ADASYN	Accuracy: Random Forest with variance threshold, 99.78%
Fiore et al. [35]	Deep Neural Network	SMOTE, GAN	GAN because of the f-measure for SMOTE shows the limited variation
Kudugunta and Ferrara [36]	Logistic Regression, SGD, Random Forest, AdaBoost and Contextual LSTM	SMOTOMEK , SMOTENN	Contextual LSTM with SMOTE: 99%

similar, but the BOINC project was not developed for an inappropriate reason.

D. MACHINE LEARNING AND OVERSAMPLING TECHNIQUE IN BOTNET DETECTION MODEL

The implementation of machine learning in malware identification led to impressive performance. The need for machine learning in malware identification is due to the complex and sophisticated [37] patterns that require time-consuming processes through human monitoring [38]. Machine learning was able to learn the sample data pattern and recognized a similar pattern, although it was intricate [39]. Machine learning techniques can be divided into supervised, semi-supervised, and unsupervised techniques. The supervised technique uses labeled data to train the algorithm to predict the class; this is called classification. The unsupervised technique uses unlabeled data, and the algorithm will plot a similar pattern into clusters; this is called clustering.

The oversampling technique is a supervised resampling technique that uses a k-Nearest Neighbor (k-NN) to generate new synthetic data based on the best location. TABLE 3 shows the combination of classifiers with oversampling used by other researchers and the best combination for each publication. In Pajouh *et al.* [32] and Alam and Vuong [33], the authors used the Synthetic Minority Oversampling Technique (SMOTE), combining several classifiers such as Naive Bayes, Support Vector Machine, Multilayer Perceptron, and Decision Tree j48 to detect malware. SMOTE was used to double, triple, or quintuple the original size. The best combination was using a Support Vector Machine (SVM) with a Radial Base Function (RBF) kernel; this achieved 91% success with a false alarm rate of 3.9%. If using Decision tree-J48 with SMOTE-5x, the accuracy was 96.62%, and the false alarm rate is 4.0. In Fiore *et al.* [35], the experiment compared SMOTE and GAN, which were combined with a deep neural network. Their results show that GAN's f-score was higher than that for SMOTE, but GAN was more complex than SMOTE. In Kudugunta and Ferrara [36], the model's performance increased with the combination of contextual LSTM with SMOTE compared to the results that only use contextual LSTM. The combination of oversampling techniques and classifiers in TABLE 3 led to an increase in the detection model's performance.

III. PROPOSED MULTILAYER FRAMEWORK FOR BOTNET DETECTION

The proposed method consisted of two main modules, namely the Filtering Module and Detecting C&C Server Module, as shown in FIGURE 4. Both modules used flow-based features and are behavior-based. The first module's purpose was to filter and reduce network traffic for the second module. The filtering module used a semi-supervised concept whereby we used partly labeled datasets to determine a similar pattern of other unlabeled data. The unsupervised algorithm clustered the uncertain network traffic with the labeled data (normal and botnet). Since the purpose is to filter the network traffic, we minimized the number of features and grouped the network traffic in the minimum time interval (1-s time interval).

Once the module clustered the uncertain data in the botnet cluster, the network traffic from this cluster transferred to the second module to detect the Command-and-Control server.

Meanwhile, the purpose of the second module was to detect the botnet C&C server to take down the botnet by blocking the source IP from entering the network. In this module, the network traffic was extracted and aggregated based on the Source Address (Sip) within the observing time (t). This module used supervised labeled data for classification.

A. FEATURE SELECTION

The first and second modules used different feature selection, but both used flow-based features. Due to botnet trends that used the concealment technique, where the payload is inaccessible, we opted to use flow-based features that analyzed the packet header. Flow-based features do not use the content or payload of the data; therefore, if the packet is

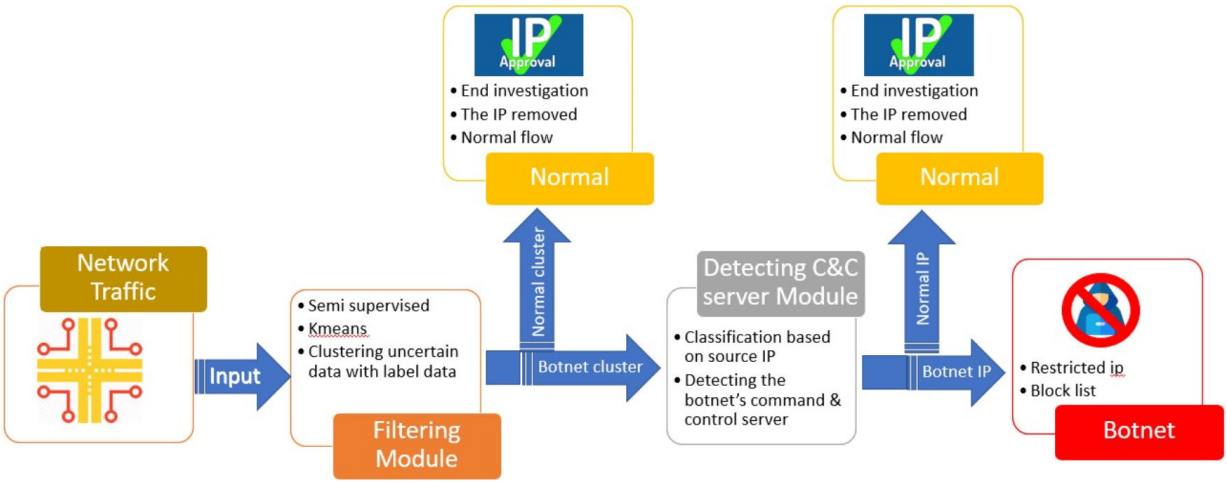


FIGURE 4. Block diagram for the proposed multilayer framework for botnet detection using machine learning algorithms.

encrypted [40]–[42] or uses a VPN tunnel, the performance is not decreased. The features selected in this experiment were derived based on the botnet's communication pattern and its botmaster during the C&C stage. As mentioned in Section 2.2, during the C&C stage, the bots communicate with the botmaster periodically [43], [44]. While communicating, their behavior is consistent, and the requested and updated sessions result in many uniformly sized, small packets that occur continuously.

B. CLASSIFICATION & OVERSAMPLING

After selecting features, the data were aggregated to be the input in the following process, which for the first module was clustering, and for the second was classification. For this study, we used a k-means algorithm. The clustering was done through Weka, a machine learning tool and library, and the results proceeded to the evaluation process.

The second module is the classification module to detect the Command-and-Control server through the source IP. To find the best classifier for our features, we compared three classifiers, k-NN, SVM, and Multilayer Perceptron. These three classifiers use very different approaches. The k-NN is a distance-based supervised algorithm that classifies an input based on the distance to the nearest number of k, while SVM is an algorithm that classifies data based on a hyperplane. The SVM algorithm calculates the optimal hyper-plane to separate each class. The SVM is versatile and can be set based on the kernel; for this research, the kernel chosen was a radial basis function (RBF). Multilayer Perceptron is a technique that combines input and output with at least one hidden layer with learning rules to update the weight.

The second module performed the classification process using the Python language, Scikit-learn (Python library). The dataset was split along a 70-30 ratio, where 70% was the training set and 30% was the testing set. The evaluation and prediction were run on the testing dataset only. The second module is a binary classification (''Normal'' or ''Botnet''), shown in Equation 1. In this experiment, we compared several classifiers: Multilayer Perceptron (MLP), k-Nearest Neighbor (k-NN), and Support Vector Machine (SVM). The classifier is combined with an oversampling technique to explore whether oversampling can improve the classifiers' performance.

$$x = \begin{cases} Normal, & if \ \ x = 0 \\ Botnet, & if \ \ x = 1 \end{cases} \tag{1}$$

1) DETERMINING THE K-VALUE

Since the algorithm that we chose included the k-algorithm, k-means, and k-NN, we needed to determine the k-value first. Several techniques can be used to find the optimal value of k; we have tried two techniques that used the dendrogram and elbow method. The dendrogram is a visualization tree that shows the data as a point, and the points are plotted based on the distance from each other. The dendrogram involves bottom-to-top plotting, and from it we can decide the distance (y-axis) that we set for points. For example, in FIGURE 5, a distance point of 100 was selected, and four was the optimal number of clusters. Unfortunately, when we increased the number of samples, the dendrogram could not plot due to memory error.

The elbow method is a technique that helps to determine the optimal number of k in either k-means or the k-NN algorithm. The elbow method for plotting a graph is where the whole graph is called the arm, and the point of inflection on the curve is the elbow. The elbow method is calculated by using the metric of Within Cluster Sum of Squares (WCSS), which calculates the sum of squared distances from each point to its assigned center. Algorithm 1 shows the Python code for generating the elbow method using the Scikit-learn (Python library). In contrast, FIGURE 6 is an example of the elbow method for Experiment B, where the x-axis is the number of the cluster, while the y-axis is the average of WCSS. So, based on this elbow method, the k-value was decided to be 4.

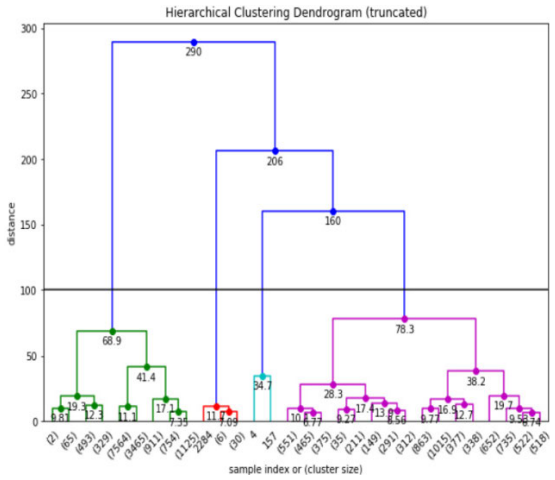


FIGURE 5. The dendogram for determining the k-value.

Algorithm 1 :Python Code for WCSS Elbow Method

```
from sklearn.cluster import kMeans
Wcss = []
for i in Range (1,11):
    Kmeans = kMeans (n_cluster =i, Init = 'k-means ++',
max_iter = 300, n_init = 10,random_state = 0)
    Kmeans. fit(X)
    wcss.append(kmeans.inertia_)
```

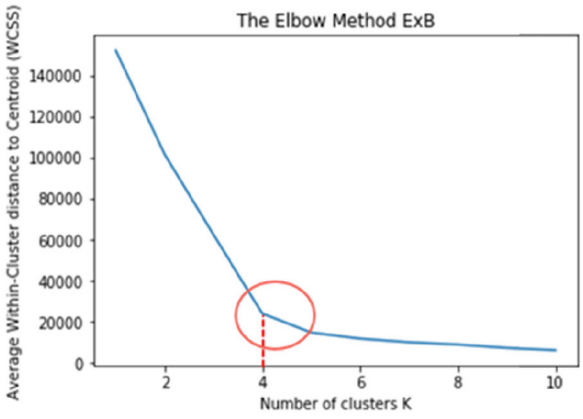


FIGURE 6. An example of the elbow method to determine k-value.

2) OVERSAMPLING TECHNIQUE

The oversampling technique is a technique to duplicate data, commonly used for a highly imbalanced dataset so that all classes have a similar amount of data. In the meantime, undersampling will reduce the majority class percentage until the amount is equivalent to the minority class.

Although the data distribution in this research was not highly imbalanced, we wanted to explore how oversampling and undersampling or generating synthetic data can contribute to the classifiers' performance. Since we used Scikit-learn, the Python library, the oversampling/undersampling technique that we choose is the Synthetic Minority Oversampling Technique (SMOTE), a combination of SMOTE, Edited Nearest Neighbors (SMOTEENN), and random oversampling (ROS).

SMOTE, is a distance-based algorithm where these algorithms identify objects as determined by distance measure via the dissimilarity between them. A random example of the minority class is chosen first. For that case, k of the nearest neighbors is then found. A random neighbor is selected, and a synthetic example is generated between the two examples in the feature space at a randomly selected point.

While SMOTEENN is a combination of oversampling and undersampling, the oversampling of SMOTE combine with undersampling, Edited Nearest Neighbours (ENN) for cleaning. ENN excludes any example whose class mark varies from that of at least two of its three closest neighbors.

ROS is the most straightforward oversampling technique where it was randomly picking, deleting, and adding to the training dataset examples from the minority class. This experiment explores the effectiveness of the oversampling technique in three different oversampling approaches, the simplest one, the basic, and the combination of over & under-sampling.

IV. EXPERIMENTAL

The experiment for this research used Python and Scikit-learn (python library) for the whole process. The experiment ran in Anaconda (Python prepackaged distribution), consisting of Jupyter Notebook, an open-source web application. Processes such as feature selection and aggregation of the dataset occur through the first module and second module.

The feature selection and the aggregation process are preprocessing to prepare the dataset for the experiment. Before we explain this experiment's process, the next subsection details the dataset used in this experiment and why we chose to use it.

A. DATA RESOURCES

The dataset that we used in this experiment was from the CTU-13 dataset [30]. CTU-13 is a dataset of network traffic that was captured at CTU University, Czech Republic, in 2011 and stored in. pcap files. The CTU-13 dataset is a labeled dataset that contains 13 scenarios labeled Normal, Attack, or Background. The 13 files contain different botnet types, as shown in TABLE 5, including centralized or decentralized structures and various protocols. This study focused on designing botnet detection that is structure-independent and protocol-independent, this dataset suited our purpose.

In the first module, we aimed to explore the unsupervised algorithm that can cluster the data group that can differentiate benign and botnet groups. The algorithm also needed to be robust to noise or uncertain data because uncertain data are more prevalent in real network traffic than regular and botnet traffic [40], [43]. We tested four types of the botnet, Neris, Virut, Murlo, and NSIS, where the combination of these botnets consisted of both structures, centralized, and decentralized. Each of these botnet types was combined with the uncertain data or not to produce a comparison. The expla-

TABLE 4. Feature description for the experiment.

No.	Main features	Type of data	Description of aggregation features	First Module	Second Module
1	Source address	categorical	Distinct* number (Sip)	√	√
			Number of transactions (Length)		√
2	Destination address	categorical	Distinct* number (Dip)	√	√
3	Destination port	categorical	Distinct* number (Dport)	√	
4	Packet bytes	continuous	Total bytes sent (TotBytesSend)		√
			Total bytes received (TotBytesRcv)		√
5	Number of packets	continuous	Total number of packets sent (Psend)		√
			Total number of packets received (Prcv)		√
6	Start time	continuous	Time difference from first packets to the last packet in time interval given (Time)		√
Total features				3	8

TABLE 5. Distribution of botnet name, structure in CTU-13.

Dataset File No.	Duration (h)	Botnet name	Structure	No. of Bots
1	6.15	Neris	IRC	1
2	4.21			1
3	66.85	Rbot	IRC	1
4	4.21			1
5	11.63	Virut	HTTP	1
6	2.18	Menti	IRC	1
7	0.38	Sogou	HTTP	1
8	19.5	Murlo	IRC	1
9	5.18	Neris	IRC	10
10	4.75	Rbot	IRC	10
11	0.26			3
12	1.21	Nsis.ay	P2P	3
13	16.36	Virut	HTTP	1

TABLE 6. The percentage of distribution data for the filtering module.

Experiment No.	No. of flows			Botnet Name and Types
	Normal	Botnet	Uncertain	
Experiment A	14,706 (66%)	7450 (33%)	x	Neris (IRC)
Experiment B	14,706 (29%)	7450 (15%)	28,527 (56%)	
Experiment C	23,749 (87%)	3601 (13%)	x	Virut (http)
Experiment D	23,479 (42%)	3601 (6.48%)	28,527 (51%)	
Experiment E	23,479 (90%)	2483 (9.6%)	x	Murlo (IRC)
Experiment F	23,479 (39%)	2483 (4%)	33,513 (56%)	
Experiment G	3443 (95.6%)	157 (4.4%)	x	NSIS (P2P)
Experiment H	3443 (12.7%)	157 (0.57%)	23,479 (86.7%)	

nation for the distribution of data is shown in TABLE 6. Experiments A, C, E, and G were the experiments without uncertain data.

In contrast, Experiments B, D, F, and H were the experiments where the input was a combination of a regular, botnet, and uncertain network traffic. In TABLE 6, we show the distribution and the ratio of Normal, Botnet, and Uncertain for each experiment. We kept the real network traffic ratio, which was highly imbalanced, where the uncertain data had the highest percentage and the botnet traffic the lowest.

The second module was the classification module using labeled data. For this module, we used a combination of normal and botnet network traffic. TABLE 7 shows the distribution of data and the combination of files for the training and testing process. Once again, these files consisted of centralized and decentralized structured botnets.

B. FEATURE SELECTION

The features selected for this study are listed in TABLE 4. In TABLE 4. The features for both modules are listed and these features are represented as X in Equation (2) and Equation (3).

The features used in the first module were source address (Sip), destination address (Dip), and destination port (Dport). Since the data for these three features are categorical data, the analysis is performed by calculating each feature's distinct number in the time interval.

The second module used five main features. The main features are then extended to several features for considering the communication pattern in two ways, either the source address is sending or receiving packets. We believe that the communication between the botnet and its botmaster can be detected within a short time, so the default time for this experiment was t = 1 s. The feature description is shown in TABLE 4. The aggregation of the first module and second module can be represented by Equations (2) and (3) where X_1, X_2 until X_n are the features that form an array:

$$[t(1s)] = [X_i, X_2\ , \ldots\ldots\ldots, X_n] \tag{2}$$

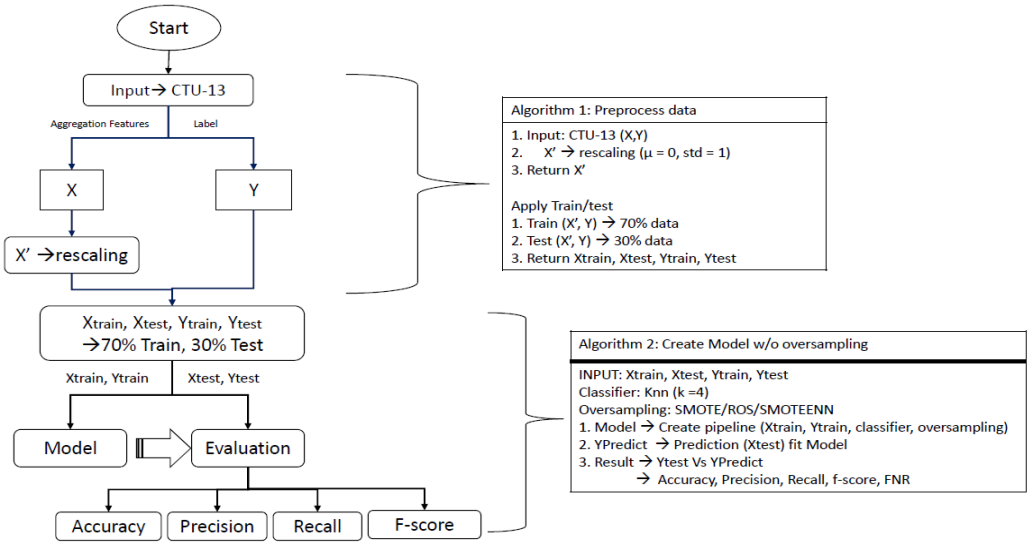


FIGURE 7. The pseudo-code and the flow chart for second module.

TABLE 7. The data distribution for the second module.

Data File	Duration (h)	Bot Name	Botnet Type
3	66.85	Rbot	IRC
4	4.21	Rbot	IRC
5	11.63	Virut	http
7	0.38	Sogou	http
10	4.75	Rbot	IRC
11	0.26	Rbot	IRC
12	1.21	Nsis.ay	P2P
13	16.36	Virut	http

$$[Sip, t] = [X_i, X_2, \ldots\ldots\ldots, X_n] \tag{3}$$

In TABLE 4, at the column 4 that shows the description of the aggregation features, we marked the word of Distinct with *. In this study, a distinct number equal to the number of unique elements in the set or in the time interval. The distinct number also can represent as shown in Equation (4), where X is the features and n(x) is the distinct number: -

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

C. CLASSIFICATION

After preprocessing, the data are ready to insert into the machine learning algorithm. The first module used K-means in WEKA, while the second module used three classifiers from Scikit-Learn for the classification process. The classifiers used are k-Nearest Neighbor (k-NN), Support Vector Machine (SVM), and Multilayer Perceptron (MLP).

In both layers, the aggregated data were then rescaled using Standard Scaler from Scikit Learn. The data were rescaled to ensure the mean value was zero and the standard deviation is equal to 1. The equation for rescaling the data is shown in Equation (4) where μ is the data mean, and s is the standard deviation.

$$z = {}^{(x-\mu)}/_{s} \tag{5}$$

For the first layer, FIGURE 8a-c) shows the sample data in the step-by-step process. FIGURE 8a gives the aggregated data after preprocessing. FIGURE 8 has the data after the rescaling process, and FIGURE 8c provides the result extracted from WEKA. As shown in FIGURE 8, the class/label attribute was removed and not rescaled with the other three features. The data in FIGURE 8b are the data inserted into WEKA. After WEKA clustered the data, the class/label feature that was removed earlier was combined with the data and the cluster number (WEKA result) to make it ready for evaluation.

For the second layer, the rescaled data then go through the pipeline process from Scikit Learn. The pipeline process is a process that is sticking multiple processes together into a single estimator. After the data were pipelined, they

	SrcAddr	DstAddr	Dport	Label
0	24	32	15	1
1	44	39	18	1
2	54	67	20	1
3	41	34	19	1
4	50	47	23	1

a)The aggregated data

	SrcAddr	DstAddr	Dport
0	1.738778	1.171694	0.513257
1	3.728956	1.527983	0.666113
2	4.724045	2.953143	0.768016
3	3.430430	1.273491	0.717064
4	4.326010	1.935172	0.920871

b)After rescale

	SrcAddr	DstAddr	Dport	cluster	Label
0	1.738778	1.171694	0.513257	cluster4	1
1	3.728956	1.527983	0.666113	cluster3	1
2	4.724045	2.953143	0.768016	cluster3	1
3	3.430430	1.273491	0.717064	cluster3	1
4	4.326010	1.935172	0.920871	cluster3	1

c)Result from WEKA

FIGURE 8. Step-by-step data changes in the first module.

TABLE 8. Determining cluster based on the percentage of majority (botnet, normal & uncertain data).

Cluster No	No of uncertain data	No of Botnet Flow	No of Normal Flows	Percentage of Botnet	Percentage of Normal Flows	Remark
1	6200	277	2326	0.1064	**0.8934**	Normal Cluster
2	748	89		**1**	0	Botnet Cluster
3	4126	401	22	**0.9480**	0.0520	Botnet Cluster
4	141	7	9	0.4375	**0.5625**	Normal Cluster
5	464	14	276	0.0483	**0.9517**	Normal Cluster

were classified and oversampled according to the classifier and oversampling technique mentioned in Section 3.2. The classification process and the oversampling process were in a confusion matrix and ready for evaluation. FIGURE 7 shows the flow chart of the process in the second module with the pseudo-code as well.

V. EVALUATION AND RESULT

The evaluation of this study was based on a confusion matrix for both modules. Although the first module used a clustering algorithm, we evaluated it as a semi-supervised technique and evaluated the botnet and normal labels. The uncertain data were not calculated in the evaluation because the insertion of uncertain data was considered to create noise. Before we generated the confusion matrix, we needed to determine whether it was a botnet cluster or a normal cluster based on the majority, as shown in TABLE 8. TABLE 8 is an example of the calculations used for determining the clusters for the experiment with and without uncertain data. As shown in TABLE 8, the number of uncertain data points was not calculated when determining the cluster.

Confusion Matrix is the most common metric used in evaluating the performance of the machine learning model. By generating a confusion matrix from the model, the distribution of the results can be seen clearly. Both modules evaluated only two (2) classes, so, the confusion matrix consisted of a specific two-dimensional table layout with the classes ''Actual'' and ''Cluster/Prediction'' in one dimension. In contrast, the other dimension had ''Botnet'' as positive and ''Normal'' as negative. The instances were categorized into four fractions, namely False Positive, False Negative, True Positive, and True Negative, as shown in TABLE 9, while the explanation of each fraction is given in TABLE 10.

The essential criterion for evaluating the Machine Learning Models is that they must suit the business impact and goal. Hence, from the confusion matrix, we expanded the performance evaluation. For this study, the prediction of binary classification was either the network traffic containing botnet attempts (positive) in the network or not.

The most common necessary measure is accuracy. Still, according to Muller and Guido [46], accuracy is not sufficient to assess classifiers' performance, so we also included other performance parameters in our evaluation, such as Precision, Recall, False Negative Rate (FNR), and f-score.

The equation for each performance parameter is in Equation 4 until Equation 8, and the description of the evaluation parameter is listed in TABLE 12:

TABLE 9. Confusion matrix.

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

TABLE 10. The fraction of the confusion matrix for the botnet classification.

Fraction	Module 1	Module 2
True Positive (TP)	TP is counted when the botnet traffic is in the botnet cluster	TP is counted when the model correctly predicts the botnet as the botnet traffic/IP
True Negative (TN)	TN is counted when the normal data are in the normal cluster	TN is counted when the model correctly predicts the benign as the benign traffic/IP
False Positive (FP)	FP is counted when the normal data are in the botnet cluster.	FP is counted when the model incorrectly predicts the benign as the botnet traffic/IP
False Negative (FN)	TP is counted when the botnet data are in the normal cluster.	FN is counted when the model incorrectly predicts the botnet as the benign traffic/IP

$$Accuracy = \frac{TP + TN}{\sum data} \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall\ (TPR) = \frac{TP}{TP + FN} \tag{8}$$

$$F\_score = 2 * \frac{Precision * Recall}{Precisin + Recall} \tag{9}$$

$$FNR = \frac{FN}{FN + TP} \tag{10}$$

In this experiment, the prediction classes included either positive or negative for botnet traffic or normal traffic. The precision is the percentage of true positives compared to all the positive predictions. This shows how well the classifier predicts the positive botnet traffic as positive. Recall, also called Sensitivity or True Positive Rate (TPR), is the percentage of positive predictions from overall positive instances. F-score is a harmonic combination between precision and

TABLE 11. The k-means result for the botnet behavior.

Botnet Type	Experiment No.	Accuracy	Precision	Recall	F-score	FNR
Neris	A	0.9978	0.998	0.9949	0.9968	0.0051
(IRC)	B	0.9978	0.998	0.9949	0.9968	0.0051
Virut	C	1.00	1.00	1.00	1.00	0
(HTTP)	D	0.9998	1.00	0.9986	0.9993	0.0014
Murlo	E	0.9939	1.00	0.9360	0.9669	0.0640
(IRC)	F	0.9992	0.9988	0.9928	0.9958	0.0072
Nsis	G	0.9934	0	0	0	1.0
(P2P)	H	0.9963	0.9726	0.4522	0.6174	0.5478

TABLE 12. Description of evaluation term.

Evaluation Term	Description
Accuracy	The overall performance of the model.
Precision	The percentage of classified botnet instances that are truly botnets
Recall (TPR)	The effectiveness of the model in detecting botnets.
F_score	The harmonic combination of recall and precision.
FNR	Type II error. This is the rate of error when the model wrongly predicts/clusters the botnet as normal.

TABLE 13. Confusion matrix for experiment G.

		Cluster	
		Normal	Botnet
Actual	Normal	TP: 0	FP: 970
	Botnet	FN: 2466	TN: 13053

recall. It is the simplest way to measure use one evaluation and compare it to the two used values. Other than that, since this study seeks to minimize Type II error, the False Negative Rate was also included in the evaluation. Among all these parameters, we highlight the F-score and FNR because F-score is a harmonic combination between Recall and Precision.

TABLE 11 shows the results for the first module that used the k-means algorithm with all the measurement parameters. Based on TABLE 11, we see that the accuracy of all the experiments, from A to H, was in the range of 99% and 100% for all types of the botnet. However, we can see that the F-score for the Nsis botnet, which was a decentralized P2P botnet, was 0% for experiment G (without uncertain data) and 62% for experiment H. If we compared the results of FNR, the same would be true: in experiments G and H, the FNR was higher than in the other experiments. We highlighted in red the Precision, Recall, and F-score that showed a 0 value. TABLE 13 shows the confusion matrix for experiment G; based on this table, the reason why Precision, Recall, and F-score became 0% was that the True Positive was 0.

TABLE 14 shows the results for the second module. All the highest scores for each of the measurement parameters are highlighted in bold. Referring to this table, we can see that this experiment's overall accuracy performance varied from 83% to 92%, while the f-score for the classifier varied from 82% to 92%. The highest accuracy and f-score used k-NN without any oversampling technique. However, the lowest FNR used a combination of k-NN with SMOTE.

FIGURE 9 is a graph representing TABLE 14. In FIGURE 10, we extract the results of accuracy and f-score of each classifier, with and without oversampling. Among these three classifiers, k-NN showed consistent values for accuracy and f-score, with or without the oversampling technique.

The performance for SVM increased when it was combined with SMOTEENN compared to SVM with other oversampling techniques. However, the performance of MLP in this experiment showed the lowest results and did not significantly change when combined with an oversampling technique.

Based on TABLE 14, the highest f-score is obtained by using the k-NN algorithm without any oversampling technique with a 1-s time interval. We extend the experiment to explore the changes that result if we use a different time interval. We test the k-NN algorithm with five-time intervals (1, 30, 60, 90, or 120 s). Changing the dataset's time interval means that we need to re-aggregate the CTU13 dataset before the classification process and evaluation. The result for k-NN using different time intervals is shown in TABLE 15. Based on TABLE 15, the highest f-score is still from using k-NN without any oversampling technique and a 1-s interval.

VI. DISCUSSION

The behavior-based analysis focuses on selecting features based on a particular concept or pattern that can extract different behavior patterns over time. In this case, we chose the flow-based features based on the theoretical relationship between the command and control server that is used by the botmaster with the botnet. The time interval for our experiment was 1 s. We chose 1 s because we wanted to test whether, within a short period, the pattern of the behavior can be differentiated. Through the botnet's life cycle, we understood that the command and control server is the most important

TABLE 14. The classification result for the botnet behavior model.

Algorithm	Oversampling	Accuracy	Precision	Recall (TPR)	f-score	FNR	TNR
k-NN	None	**0.922**	0.930	**0.900**	**0.9151**	0.0594	0.9405
	ROS	0.9165	0.9653	0.8519	0.9050	0.0268	0.9731
	SMOTE	0.9205	**0.9790**	0.8480	0.9088	**0.0158**	**0.9841**
	SMOTEENN	0.9193	0.9695	0.8541	0.9081	0.0236	0.9764
SVM-RBF	None	0.8595	0.8503	0.8487	0.8495	0.1309	0.8690
	ROS	0.8596	0.8500	0.8493	0.8496	0.1313	0.8686
	SMOTE	0.8596	0.8499	0.8494	0.8497	0.1314	0.8685
	SMOTEENN	0.9134	0.9642	0.8463	0.9014	0.0276	0.9724
MLP	None	0.8458	0.8195	0.8591	0.8388	0.1658	0.8341
	ROS	0.8429	0.8161	0.8567	0.8359	0.1692	0.8308
	SMOTE	0.8428	0.8161	0.8566	0.8359	0.1691	0.8308
	SMOTEENN	0.8311	0.8015	0.8486	0.8244	0.1843	0.8157

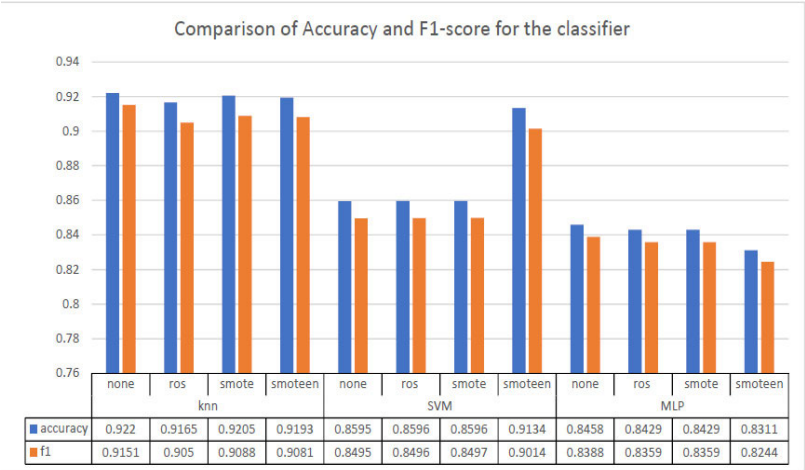
TABLE 15. The classification result for the k-NN with a different time interval (second).

Time (second)	Oversampling	Accuracy	Precision	Recall (TPR)	f-score	FNR
1	*None*	*0.922*	*0.930*	***0.900***	***0.9151***	*0.0594*
	ROS	0.9165	0.9653	0.8519	0.9050	0.0268
	SMOTE	0.9205	**0.9790**	0.8480	0.9088	**0.0158**
	SMOTEENN	0.9193	0.9695	0.8541	0.9081	0.0236
30	None	**0.9591**	0.9507	0.8	0.8688	0.2
	ROS	0.9432	0.8091	0.8707	0.8387	0.1292
	SMOTE	0.9548	0.8706	0.8615	0.866	0.1384
	SMOTEENN	0.9383	0.7805	0.8847	0.8293	0.1152
60	None	0.9515	0.9238	0.7899	0.8516	0.21
	ROS	0.9326	0.7639	0.8939	0.8238	0.106
	SMOTE	0.9434	0.8323	0.8497	0.8409	0.1502
	SMOTEENN	0.9154	0.7088	0.8823	0.7861	0.1176
90	None	0.9366	0.9208	0.7249	0.8112	0.275
	ROS	0.9255	0.7978	0.8075	0.8026	0.1924
	SMOTE	0.9285	0.8186	0.7953	0.8068	0.2046
	SMOTEENN	0.926	0.7762	0.8509	0.8118	0.149
120	None	0.9293	0.904	0.7019	0.7903	0.298
	ROS	0.908	0.7174	0.8493	0.7778	0.1506
	SMOTE	0.9265	0.8135	0.7947	0.804	0.2052
	SMOTEENN	0.9127	0.7418	0.8278	0.7824	0.1721

thing for a botnet to function. The current trends of botnets are changes in structure and the obfuscation technique on the packet data, which creates challenges for researchers designing detection models. Several research pieces show that traditional signature-based or content-based methods are unable to detect botnets. Still, with behavior-based and flow-based methods, it may be possible to solve the problem. The imbalanced distribution of normal and botnet traffic can also contribute to the failure to detect botnet traffic. The meager amount of botnet data compared to the very high amount of benign packet data means that the botnet traffic often goes unseen.

The comparison made with other research on botnet detection shows that researchers tend to design botnet detection only for a particular structure and protocol. Hence, for our study, we have highlighted criteria independent of structure and protocol by selecting the CTU-13 dataset, consisting of both types of structure, centralized and decentralized, and a combination of the protocols. CTU-13 also represents real-time traffic and contains a highly imbalanced distribution of botnet and benign data.

Based on the results, our method, starting with the selection of features and continuing through the preprocessing, the chosen time interval, and the algorithm, achieved impres-



	knn				SVM				MLP			
	none	ros	smote	smoteen	none	ros	smote	smoteen	none	ros	smote	smoteen
accuracy	0.922	0.9165	0.9205	0.9193	0.8595	0.8596	0.8596	0.9134	0.8458	0.8429	0.8429	0.8311
f1	0.9151	0.905	0.9088	0.9081	0.8495	0.8496	0.8497	0.9014	0.8388	0.8359	0.8359	0.8244

FIGURE 9. Comparison of accuracy and F-score of the classifiers.

sive results. This proves that behavior-based analysis and flow-based features without accessing the payload can determine the botnet traffic, even for an imbalanced class dataset.

VII. CONCLUSION

As mention in the literature review referring to TABLE 3, our outcome is in total contrast with the previous researcher's result. TABLE 3 shows that oversampling improves the result that produces by the classifier. However, surprisingly, oversampling in our research did not show any significant change. The k-NN algorithm alone has a result that overcomes the result produce by combining k-NN with oversampling. This result determines our next steps to extends the experiment where we will use k-NN without oversampling technique.

Since we aimed to maximize the f-score, the highest result obtained for the f-score was through the k-NN without any oversampling technique, which was 91.51% with a 1-s time interval. Although we changed the time interval to 1, 30, 60, 90, or 120 s, the highest f-score was still obtained by using the 1-s time interval. Although we used a behavior-based method to analyze the botnet in network traffic, this proved that we do not need a longer time interval to observe the communication pattern among bots and its botmaster.

There are still some issues that need to be addressed in a future study. As we can see, the performance decreased while clustering the decentralized botnet (experiment G with the NSIS botnet). In the future, we would like to expand our method to test novel types of botnets and evaluate them based on performance and time (processing and detecting time). We would like to create a dynamic framework that would predict future botnet behavior and test it with several benchmark botnet datasets.

ACKNOWLEDGMENT

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. The authors would like to acknowledge that this work was supported/funded by the Ministry of Higher Education under the Fundamental Research Grant Scheme (FRGS/1/2018/ICT04/UTM/01/1). The work was also partially supported by the Specific Research project (SPEV) at the Faculty of Informatics and Management, University of Hradec Kralove, Czech Republic, under Grant 2102-2021. The authors are grateful for the support of student Sebastien Mambou in consultations regarding application aspects. The authors also wish to thank the Ministry of Education Malaysia for the Hadiah Latihan Persekutuan (HLP) scholarship to complete the research.

REFERENCES

[1] X. D. Hoang, "Botnet detection based on machine learning techniques using DNS query data," *Future Internet*, vol. 10, no. 5, pp. 1–11, 2018.
[2] P. Wainwright and H. Kettani, "An analysis of botnet models," in *Proc. 3rd Int. Conf. Compute Data Anal.*, New York, NY, USA, Mar. 2019, pp. 116–121.
[3] J. Johnson, "Lost your data in the flood? 5 tips to data recover in a flash! Antara WhatsApp & Telegram: Komunikasi alaf baru yang digemari," CyberSecurity, Kuala Lumpur, Malaysia, Tech. Rep., Feb. 2017, vol. 43.
[4] J. A. Cid-Fuentes, C. Szabo, and K. Falkner, "Adaptive performance anomaly detection in distributed systems using online SVMs," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 5, pp. 928–941, Sep./Oct. 2018.
[5] E. Bou-Harb, M. Debbabi, and C. Assi, "Big data behavioral analytics meet graph theory: On effective botnet takedowns," *IEEE Netw.*, vol. 31, no. 1, pp. 18–26, Jan. 2017.
[6] R. Chen, W. Niu, X. Zhang, Z. Zhuo, and F. Lv, "An effective conversation-based botnet detection method," *Math. Problems Eng.*, vol. 2017, pp. 1–9, Apr. 2017.
[7] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013.
[8] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *Proc. 5th Conf. Inf. Knowl. Technol.*, May 2013, pp. 113–120.
[9] J. A. Caicedo-Muñoz, A. L. Espino, J. C. Corrales, and A. Rendón, "QoS-classifier for VPN and non-VPN traffic based on time-related features," *Comput. Netw.*, vol. 144, pp. 271–279, Oct. 2018.
[10] S. Alrabaee, M. Debbabi, and L. Wang, "On the feasibility of binary authorship characterization," *Digit. Invest.*, vol. 28, pp. S3–S11, Apr. 2019.
[11] R. Rapuzzi and M. Repetto, "Building situational awareness for network threats in fog/edge computing: Emerging paradigms beyond the security perimeter model," *Future Gener. Comput. Syst.*, vol. 85, pp. 235–249, Aug. 2018.

[12] P. Sun, J. Li, M. Z. A. Bhuiyan, L. Wang, and B. Li, "Modeling and clustering attacker activities in IoT through machine learning techniques," *Inf. Sci.*, vol. 479, pp. 456–471, Apr. 2019.

[13] S.-H. Li, Y.-C. Kao, Z.-C. Zhang, Y.-P. Chuang, and D. C. Yen, "A network behavior-based botnet detection mechanism using PSO and K-means," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 1, pp. 1–30, Apr. 2015.

[14] D. Zhuang and J. M. Chang, "Enhanced PeerHunter: Detecting peer-to-peer botnets through network-flow level community behavior analysis," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 6, pp. 1485–1500, Jun. 2019.

[15] K. Ehsan and R. S. Hamid, "BotRevealer: Behavioral detection of botnets based on botnet life-cycle," *Int. J. Inf. Secur.*, vol. 10, no. 1, pp. 55–61, 2018.

[16] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 128, pp. 33–55, Feb. 2019.

[17] L. Mathur, M. Raheja, and P. Ahlawat, "Botnet detection via mining of network traffic flow," *Procedia Comput. Sci.*, vol. 132, pp. 1668–1677, Jan. 2018.

[18] N. B. Said, F. Biondi, V. Bontchev, O. Decourbe, T. Given-Wilson, A. Legay, and J. Quilbeuf, "Detection of mirai by syntactic and behavioral analysis," in *Proc. IEEE 29th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2018, pp. 224–235.

[19] Z. Wang, M. Tian, and C. Jia, "An active and dynamic botnet detection approach to track hidden concept drift," in *Proc. Int. Conf. Inf. Commun. Secur.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 10631, Berlin, Germany, 2018 pp. 646–660.

[20] K. M. Prasad, A. R. M. Reddy, and K. V. Rao, "BARTD: Bio-inspired anomaly based real time detection of under rated app-DDoS attack on Web," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 1, pp. 73–87, Jan. 2020.

[21] I. Sreeram and V. P. K. Vuppala, "HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm," *Appl. Comput. Informat.*, vol. 15, no. 1, pp. 59–66, Jan. 2019.

[22] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, "A LSTM based framework for handling multiclass imbalance in DGA botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, Jan. 2018.

[23] M. Pawlicki, M. Choraś, and R. Kozik, "Defending network intrusion detection systems against adversarial evasion attacks," *Future Gener. Comput. Syst.*, vol. 110, pp. 148–154, Sep. 2020.

[24] Z. Berkay Celik, R. J. Walls, P. McDaniel, and A. Swami, "Malware traffic detection using tamper resistant features," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2015, pp. 330–335.

[25] D. Santana, S. Suthaharan, and S. Mohanty, "What we learn from learning—Understanding capabilities and limitations of machine learning in botnet attacks," 2018, *arXiv:1805.01333*. [Online]. Available: https://arxiv.org/abs/1805.01333

[26] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Comput. Secur.*, vol. 70, pp. 238–254, Sep. 2017.

[27] B. AsSadhan, A. Bashaiwth, J. Al-Muhtadi, and S. Alshebeili, "Analysis of P2P, IRC and HTTP traffic for botnets detection," *Peer-Peer Netw. Appl.*, vol. 11, no. 5, pp. 848–861, Sep. 2018.

[28] V. H. Bezerra, V. G. T. da Costa, S. B. Junior, R. S. Miani, and B. B. Zarpelão, "IoTDS: A one-class classification approach to detect botnets in Internet of Things devices," *Sensors*, vol. 19, no. 14, p. 3188, Jul. 2019.

[29] L. F. Maimo, A. L. P. Gomez, F. J. G. Clemente, M. G. Perez, and G. M. Perez, "A self-adaptive deep learning-based system for anomaly detection in 5G networks," *IEEE Access*, vol. 6, pp. 7700–7712, 2018.

[30] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.

[31] I. Kurochkin and A. Saevskiy, "BOINC forks, issues and directions of development," *Procedia Comput. Sci.*, vol. 101, pp. 369–378, Jan. 2016.

[32] H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K.-R. Choo, "Intelligent OS X malware threat detection with code inspection," *J. Comput. Virol. Hacking Techn.*, vol. 14, no. 3, pp. 213–223, Aug. 2018.

[33] M. S. Alam and S. T. Vuong, "Random forest classification for detecting Android malware," in *Proc. IEEE Int. Conf. Green Comput. Commun., IEEE Internet Things, IEEE Cyber, Phys. Social Comput.*, Aug. 2013, pp. 663–669.

[34] M. Sewak, S. K. Sahay, and H. Rathore, "Comparison of deep learning and the classical machine learning algorithm for the malware detection," in *Proc. 19th IEEE/ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, Jun. 2018, pp. 293–296.

[35] U. Fiore, A. De Santis, F. Perla, P. Zanetti, and F. Palmieri, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Inf. Sci.*, vol. 479, pp. 448–455, Apr. 2019.

[36] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Inf. Sci.*, vol. 467, pp. 312–322, Oct. 2018.

[37] M. Aamir and S. M. A. Zaidi, "Clustering-based semi-supervised machine learning for DDoS attack classification," *J. King Saud Univ.-Comput. Inf. Sci.*, 2019, doi: 10.1016/j.jksuci.2019.02.003.

[38] K. Alieyan, A. ALmomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on DNS," *Neural Comput. Appl.*, vol. 28, no. 7, pp. 1541–1558, Jul. 2017.

[39] M. Stevanovic and J. M. Pedersen, "On the use of machine learning for identifying botnet network traffic," *J. Cyber Secur. Mobility*, vol. 4, nos. 2–3, pp. 1–32, 2015.

[40] R. U. Khan, X. Zhang, R. Kumar, A. Sharif, N. A. Golilarz, and M. Alazab, "An adaptive multi-layer botnet detection technique using machine learning classifiers," *Appl. Sci.*, vol. 9, no. 11, p. 2375, Jun. 2019.

[41] A. Gezer, G. Warner, C. Wilson, and P. Shrestha, "A flow-based approach for trickbot banking trojan detection," *Comput. Secur.*, vol. 84, pp. 179–192, Jul. 2019.

[42] L. Chen, Y. Ye, and T. Bourlai, "Adversarial machine learning in malware detection: Arms race between evasion attack and defense," in *Proc. Eur. Intell. Secur. Informat. Conf. (EISIC)*, Sep. 2017, pp. 99–106.

[43] C.-Y. Wang, C.-L. Ou, Y.-E. Zhang, F.-M. Cho, P.-H. Chen, J.-B. Chang, and C.-K. Shieh, "BotCluster: A session-based P2P botnet clustering system on NetFlow," *Comput. Netw.*, vol. 145, pp. 175–189, Nov. 2018.

[44] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Comput. Secur.*, vol. 81, pp. 123–147, Mar. 2019.

[45] A. C. Müller and S. Guido, *Introduction to Machine Learning With Python: A Guide for Data Scientists*, 1st ed. Newton, MA, USA: O'Reilly Media, 2016.

[46] M. Debashi and P. Vickers, "Sonification of network traffic for detecting and learning about botnet behavior," *IEEE Access*, vol. 6, pp. 33826–33839, 2018.

[47] S. Garg, S. K. Peddoju, and A. K. Sarje, "Scalable P2P bot detection system based on network data stream," *Peer-to-Peer Netw. Appl.*, vol. 9, no. 6, pp. 1209–1225, 2016.

WAN NUR HIDAYAH IBRAHIM (Member, IEEE) received the B.S. degree in engineering (electrical) and the master's degree in technical education (TVET) from Universiti Teknologi Tun Hussein Onn (UTHM), in 2006 and 2008, respectively. She is currently pursuing the Ph.D. degree with Universiti Teknologi Malaysia, Skudai. Her thesis focuses on detecting botnet in network traffic. From 2009 until 2015, she was a Senior Lecturer with the Department of Electrical Engineering, Polytechnic Sultan Idris Shah, Selangor, Malaysia, where she was teaching in Information and Communication Technology, from 2015 until 2017. Her research interests include machine learning, data analytics, malware, network security and generative adversarial network (GAN).

SYAHID ANUAR is currently a Senior Lecturer with Universiti Teknologi Malaysia Kuala Lumpur, under Razak Faculty of Technology and Informatics. His research interests include teaching machine learning, data mining, and cloud computing subjects. He is also as a Leader in a research project named the IoT and machine learning to detect driving behavior. He is a Team Member of research project named machine learning in cybersecurity for botnet prediction.

ALI SELAMAT (Member, IEEE) has also been the Dean of the Malaysia Japan International Institute of Technology (MJIIT), UTM, since 2018. An academic institution established under the cooperation of the Japanese International Cooperation Agency (JICA) and the Ministry of Education Malaysia (MOE) to provide the Japanese style of education in Malaysia. He is currently a Full Professor with Universiti Teknologi Malaysia (UTM), Malaysia, where he is also a Professor with the Software Engineering Department, Faculty of Computing. He has published more than 60 IF research papers. His H-index is 20, and his number of citations in WoS is more than 800. His research interests include software engineering, software process improvement, software agents, Web engineering, information retrievals, pattern recognition, genetic algorithms, neural networks, soft computing, computational collective intelligence, strategic management, key performance indicator, and knowledge management. He is on the Editorial Board of the *Journal Knowledge-Based Systems* (Elsevier). He has been serving as the Chair for the IEEE Computer Society Malaysia, since 2018.

ONDREJ KREJCAR is a full professor in systems engineering and informatics at the University of Hradec Kralove, Faculty of Informatics and Management, Center for Basic and Applied Research, Czech Republic; and Research Fellow at Malaysia-Japan International Institute of Technology, University Technology Malaysia, Kuala Lumpur, Malaysia. In 2008 he received his Ph.D. title in technical cybernetics at Technical University of Ostrava, Czech Republic. He is currently a vice-rector for science and creative activities of the University of Hradec Kralove from June 2020.

At present, he is also a director of the Center for Basic and Applied Research at the University of Hradec Kralove. In years 2016-2020 he was vice-dean for science and research at Faculty of Informatics and Management, UHK. His h-index is 19, with more than 1300 citations received in the Web of Science. In 2018, he was the 14th top peer reviewer in Multidisciplinary in the World according to Publons and a Top Reviewer in the Global Peer Review Awards 2019 by Publons. Currently, he is on the editorial board of the MDPI Sensors IF journal (Q1/Q2 at JCR), and several other ESCI indexed journals. He is a Vice-leader and Management Committee member at WG4 at project COST CA17136, since 2018. He has also been a Management Committee member substitute at project COST CA16226 since 2017. Since 2019, he has been Chairman of the Program Committee of the KAPPA Program, Technological Agency of the Czech Republic as a regulator of the EEA/Norwegian Financial Mechanism in the Czech Republic (2019-2024). Since 2020, he has been Chairman of the Panel 1 (Computer, Physical and Chemical Sciences) of the ZETA Program, Technological Agency of the Czech Republic. Since 2014 until 2019, he has been Deputy Chairman of the Panel 7 (Processing Industry, Robotics, and Electrical Engineering) of the Epsilon Program, Technological Agency of the Czech Republic. At the University of Hradec Kralove, he is a guarantee of the doctoral study program in Applied Informatics, where he is focusing on lecturing on Smart Approaches to the Development of Information Systems and Applications in Ubiquitous Computing Environments.

His research interests include Control Systems, Smart Sensors, Ubiquitous Computing, Manufacturing, Wireless Technology, Portable Devices, biomedicine, image segmentation and recognition, biometrics, technical cybernetics, and ubiquitous computing. His second area of interest is in Biomedicine (image analysis), as well as Biotelemetric System Architecture (portable device architecture, wireless biosensors), development of applications for mobile devices with use of remote or embedded biomedical sensors.

RUBÉN GONZÁLEZ CRESPO (Senior Member, IEEE) received the Ph.D. degree in computer science engineering. He is currently the Dean of the Higher School of Engineering, Universidad Internacional de La Rioja (UNIR), and the Director of the AENOR (Spanish Association for Standardization and Certification) Chair of Certification, Quality and Technology Standards. He is also a member of different committees with the ISO Organization. He is also an Advisory Board Member of the Ministry of Education at Colombia and an Evaluator of the National Agency for Quality Evaluation and Accreditation of Spain (ANECA).

ENRIQUE HERRERA-VIEDMA (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the University of Granada, Granada, Spain, in 1993 and 1996, respectively.

He is currently a Professor of computer science and A. I, and the Vice-President of Research and Knowledge Transfer, University of Granada. His H-index is 69, with more than 17 000 citations received in the Web of Science and 85 in Google Scholar, with more than 29 000 cites received. He has been identified as one of the World's most influential researchers by the Shanghai Centre and Thomson Reuters/Clarivate Analytics in both the scientific categories of computer science and engineering, from 2014 to 2018. His current research interests include group decision making, consensus models, linguistic modeling, aggregation of information, information retrieval, bibliometric, digital libraries, Web quality evaluation, recommender systems, block chain, smart cities, and social media. He is the Vice-President of Publications of the SMC Society and an Associate Editor of several JCR journals, such as IEEE TRANSACTIONS ON FUZZY SYSTEMS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, *Information Sciences*, *Applied Soft Computing*, *Soft Computing*, *Fuzzy Optimization and Decision Making*, *Journal of Intelligent and Fuzzy Systems*, *International Journal of Fuzzy Systems*, *Engineering Applications of Artificial Intelligence*, *Journal of Ambient Intelligence and Humanized Computing*, *International Journal of Machine Learning and Cybernetics*, and *Knowledge-Based Systems*. He is also the Editor-in-Chief of the *Journal Frontiers in Artificial Intelligence* (Section Fuzzy Systems).

HAMIDO FUJITA (Life Senior Member, IEEE) received the B.S. degree in electrical engineering from the University of Manchester, Manchester, U.K., in 1979, and the master's and Ph.D. degrees in information engineering from Tohoku University, Sendai, Japan, in 1985 and 1988, respectively. He is currently a Professor of artificial intelligence with Iwate Prefectural University, Takizawa, Japan, as the Director of intelligent software systems. He is an Adjunct Professor of computer science and artificial intelligence with Stockholm University, Stockholm, Sweden; the University of Technology Sydney, Ultimo, NSW, Australia; the National Taiwan Ocean University, Keelung, Taiwan, and others. He has supervised Ph.D. students jointly with the University of Laval, Quebec City, QC, Canada; the University of Technology Sydney; Oregon State University, Corvallis, OR, USA; the University of Paris 1 Pantheon–Sorbonne, Paris, France; and the University of Genoa, Genoa, Italy. He is also a Highly Cited Researcher in Cross-field for the year 2019 by Clarivate Analytics. He has given many keynotes in many prestigious international conferences on intelligent system and subjective intelligence. He headed a number of projects including intelligent HCI, a project related to mental cloning for healthcare system as an intelligent user interface between human users and computers, and SCOPE project on virtual doctor systems for medical applications. He is the recipient of the Honorary Scholar Award from the University of Technology Sydney, in 2012. He has four international patents in software system and several research projects with Japanese industry and partners. He is the Editor-in-Chief for *Knowledge-Based Systems*. He is the Vice President of International Society of Applied Intelligence, and currently the Editor-in-Chief of Applied Intelligence (Springer).