Egyptian Informatics Journal 24 (2023) 313–325

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

A genomic rule-based KNN model for fast flux botnet detection

Femi Emmanuel Ayo [a], Joseph Bamidele Awotunde [b,*], Sakinat Oluwabukonla Folorunso [a], Matthew O. Adigun [c], Sunday Adeola Ajagbe [d]

[a] *Department of Mathematical Sciences, Olabisi Onabanjo University, Ago-Iwoye 120107, Ogun State, Nigeria*
[b] *Department of Computer Science, Faculty of Information and Communication Sciences, University of Ilorin, Ilorin 240003, Kwara State, Nigeria*
[c] *Department of Information Technology, Cape University of Technology, Cape Town, South Africa*
[d] *Computer & Industrial Production Engineering, First Technical University, Ibadan, 200255, Oyo State, Nigeria*

A R T I C L E   I N F O

Article history:
Received 30 March 2023
Accepted 5 May 2023
Available online 13 May 2023

Keywords:
Botnet Detection
Fast Flux Botnet
K-Nearest Neighbor
Genetic Algorithm
Fuzzy Logic

A B S T R A C T

Fast Flux Botnet (FFB) is an advance method developed by cyber criminals to perpetrate distributed malicious attacks. The major problems of existing FFB detection systems are the vulnerability to evasion mechanisms, long detection time, and high dimensionality of the feature set. In this study, an improved FFB detection architecture called Bot-FFX was developed to address some of these problems. The developed Bot-FFX consists of four modules: extractor, filter, resolver, and detector. The extractor module is responsible for Domain Name System (DNS) queries on domains. The filter module can classify the incoming domains as either blacklist or whitelist and sends the unclassified domains to the resolver. The resolver extracts all IP addresses associated with the domain at its Time-To-Live (TTL) within a time frame of 10 min. The detector module uses a rule-based Genetic Algorithm (GA) and K-Nearest Neighbor (KNN) for botnet detection. The detector computed the Standard Deviation of Round Trip Time (SDRTT), Average Google Hits (AGH) and Genetic Threshold Value (GTV) for all IP addresses associated with the domains. The detector, built on a decision tree rules and the K-Dimensional (KD) tree KNN algorithm, classified the domains using the set of IP addresses, SDRTT, AGH, and GTV. The Bot-FFX was implemented on a dataset of 2,000 benign domains and 1,630 botnet domains. The dataset was split into 50% training and 50% testing sets. The evaluation results on the same datasets showed that Bot-FFX is an effective FFB detection system with accuracy, false positive, and false negative of 99.178%, 0.8%, and 0.8% respectively.

1. Introduction

Botnet is an organized network of distributed and infected computers (zombies) executing malicious codes called bots, under the remote command of a human originator called botnetmaster [1]. The evolution of the Internet has played host to a network of distributed and compromised computers. Hence, the use of the Internet attracts certain risks which make Botnet one of the key issues in Internet security.

The network of anonymous users of the Internet unaware of the importance of security provides many opportunities for malicious users to exploit [2,3]. Increasingly, malicious users are constantly developing more advanced methods to profit from cybercrime activities [4,5]. This occurrence has led to the design and implementation of a distributed architecture of remotely controlled networks of infected hosts, called botnets, for performing malicious activities. With a single command from a Command and Control (C&C) server, botnetmaster can control networks of vulnerable hosts [6,7,8]. Botnetmaster usually performs maintenance and update of their C&C set-up on Fast-Flux Service Network (FFSN) to make the detection of bots difficult.

In FFSN, the Internet Protocol (IP) address of several zombies are advertized as phishing web servers in the Domain Name System (DNS). These phishing web servers act as a masquerader by redirecting requests to the C&C server to execute the intended malicious services. The botnetmaster frequently changes these phishing web servers to the IP addresses of other bots to exploit the weakness of the Hypertext Transfer Transmission Protocol (HTTP) to detect the C&C server [9,7,10,11]. This phenomenon

* Corresponding author.
E-mail addresses: ayo.femi@oouagoiwoye.edu.ng (F.E. Ayo), awotunde.jb@unilorin.edu.ng (J.B. Awotunde), sakinat.folorunso@oouagoiwoye.edu.ng (S.O. Folorunso), profmatthewo@gmail.com (M.O. Adigun), sunday.ajagbe@tech-u.edu.ng (S.A. Ajagbe).

https://doi.org/10.1016/j.eij.2023.05.002
1110-8665/© 2023 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Computers and Artificial Intelligence, Cairo University.

may limit researchers to the option of detecting Fast Flux Botnets (FFB) through the domain name masqueraded in spam mails and other public forums.

FFB detection is a traditional machine learning task where the features of an instance are fed to a classifier and the classifier attempts to detect the class membership of that instance. However, unlike common classification tasks where the feature set is fixed, FFB detection demands that the researcher learns new features or adopt known reliable existing subsets based on the literature [12,13,14]. A number of Botnet Detection Systems (BDS) have been developed, but the identification, adoption, and merger of reliable features for detection still remains a problem. This is because most of the botnet detection systems are limited in accuracy due to the unreliable nature of the existing features [15,16,3,17]. Secondly, the inability of a BDS to deal with botnetmaster constant evasion mechanisms to masquerade the operations of legitimate Internet devices [18,17]. Evasion mechanisms are the different techniques adopted by Botnetmaster to make the detection of their bots difficult. These techniques include advertizing the IP addresses of several zombies as phishing web servers and performing update operations on the C&C server.

The motivation of this study is to adapt a reliable and effective feature to deal with current evasion schemes adopted by botnetmasters. This study developed an improved Fast Flux Botnet Detection (Bot-FFX) that adopts a K-Nearest Neighbor (KNN) classifier rooted in rule-based Genetic Algorithm (GA) consisting of three features: Standard deviation of Round-Trip-Time, Average Google Hits, and number of IP address over a time window. The main motivation for adopting the KNN is to benefit from the algorithm's high detection accuracy. The adoption of the listed features is to tackle the problem of evasion as they exhibit different behaviours for both FFB and legitimate domains. Additionally, a rule-based GA technique is introduced to reduce the time taken to differentiate between legitimate and botnet domains advertizing the same set of IP addresses over the time window.

The rest of this paper is structured as follows: Section 2 presents related work. Methodology is presented in Section 3. The implementation, evaluation, and results are presented in Section 4. Section 5 concludes the work with future work.

2. Related work

A number of solutions have been developed by researchers in recent years for BDS. These solutions can be mainly classified into honeynets-based, intrusion-based, and heuristic-based detection [19]. For instance, solutions in [20,21,22] have developed different honeynet-based detection techniques. However, honeynet-based detection may not necessarily detect botnet attacks, but useful to understand botnet architecture and features. On the other hand, intrusion-based detection solutions have been useful for botnet attack detection. More so, heuristic-based detections are based on adjustable thresholds. The classification of botnet detection solutions can be summarized below.

2.1. Honeynets based detection

Honeynet is a collection of simulated servers called honeypots on a physical server. The honeypots are loopholes that are intentionally introduced to motivate attackers to attack the system. The main purpose is to gather bot signatures and mechanisms of the C&C server [23]. The honeynet-based detection usually generates a report regarding the detected bot signatures to better understand the penetration mechanisms of the botnet [24]. However, the damages caused by the botnet are not always detected. Honeypots can be classified as low-interaction and high-interaction honeypots based on their simulation capability. The low-interaction honeypots allow partial penetration to the attackers through the simulation of a few features that define the real system [25]. In other words, low-interaction honeypots limit the accessibility of the attackers to the real system through controlled features. For example, Provos [20] presented a low interaction honeypot framework called Honeyd. The framework simulates virtual honeypots with thousands of IP addresses at the network level. The developed Honeyd showed high security capability in botnet detection and prevention. On the other hand, the high-interaction honeypots allow full penetration to the attackers through the simulation of all features of the real system. In other words, high-interaction honeypots allow full accessibility of the attackers to the real system through uncontrolled features. For example, Vrable et al. [26] developed a honeypot architecture that can simulate the full features of a real system with high scalability to support potential hundreds of live virtual machines. The developed architecture was able to detect attacker behavior at a faster rate than related honeypots architectures. In another study, Bajtoš et al. [27] proposed a network of high-interaction honeypots for botnet detection. The proposed method was able to analyze botnets in the infection phase and detect botnet based on known signatures from their collected datasets. The developed method used Pearson's correlation coefficients to show dependencies between commands, and between commands and directories used for botnet infection. The developed method showed that it can detect various types of botnet attacks.

2.2. Intrusion based detection

The intrusion-based detection is a more effective method of botnet detection that collects bot signatures and trains classifiers to identify any abnormal activities. Intrusion-based detection can be classified into four methods, namely, signature-based detection, anomaly-based detection [28,29], DNS-based detection, and mining-based detection. Signature-based detection used known signatures of existing botnets for botnet detection. However, signature-based detection methods can only be used for known botnet detection. In other words, the method cannot be used for unknown botnet detection. For example, Gu et al. [30] presented a real-time botnet detector consisting of both rule-based and anomaly detector engines for attack signature gathering. The gathered signatures are parsed by a correlator engine for detection and collection of botnet infection trails. The authors evaluated the developed system in both virtual and live network honeynet environments. The results of the evaluation showed that the developed system is accurate and scalable for botnet detection. In a similar study, Xie et al. [31] developed a spam signature generation architecture called AutoRE to detect spamming botnets. The developed AutoRE was able to detect botnet spam with a low false alarm rate without any preclassified training data. Behal et al. [32] developed a signature-based botnet detection and prevention architecture. The developed architecture can first extracts network traffic to gather information of attack types and the output stored in a database. Alerts are then generated based on the attack types identified in the network. New rules/signatures from the alerts are then developed and updated to both a detection and prevention databases. The prevention database contains rules for filtering network traffic that has been detected by the rule-based detection database to contain botnet attacks. The developed system showed that it can dynamically develop new rules for new attacks and drop the traffic in real time. Anomaly-based detection attempts to detect botnet using some anomaly system behaviors such as high network latency and high traffic volume. These anomaly system behaviors indicate the presence of bot attack in the network. For example, Chen et al. [33] proposed an ensemble anomaly-based method that employed normal traffic for training. The method consists of two detectors which profile and analyze the anomaly behavior with increased accuracy and reduced false alarms compared to traditional anomaly detectors. Martinez-Bea et al. [34] proposed a hybrid real-time fast-flux detection model by building a linear SVM classifier that merged the feature sets of both McGrath et al. [35] and Hsu et al. [10] in a bid to utilize their collective strengths. The authors argued that using the feature set of McGrath et al. [35] and Hsu et al. [10] in isolation can lead to an unacceptable rate of false negatives and false positives respectively. The authors trained and validated their classifier with a k-fold cross validation method using dataset extracted from domain names advertised on various malware reporting forums. The scheme provided a lower false positive rate compared to reviewed works examined by the authors. Zhao & Traore [36] designed a detection system using flow metrics to create a decision tree based approach in detecting botnets. Six (6) features were proposed for botnet detection– mean reported TTL (MTTL) upon performing DNS queries, actual mean TTL (ATTL) as observed over time by the detection system, Total Unique A records (ARCRD) similar to Number of A records, A Record Change Variance (ARCRDV) similar to the frequency of A record change, A Record IP stability (ARCRDS) similar to the number of subnets observed, and Domain name confidence (DCONF) similar to the domain age feature. Initially, a fast rule based detector observes the highlighted attributes for each domain over a period of one (1) week, and schedules domains that suggest malicious fast flux behavior for extended monitoring. DNS records of suspected domains are polled continuously at a rate of half its TTL value on the A records, and the responses are captured. The data captured through this process are then transformed into a set of attributes which are then fed into a decision tree. The results showed reduced false positives compared to other schemes and not prone to disguise attacks. Celik & Oktug [14] proposed a detection system purely based on the DNS request and the corresponding response packets collected from recursive DNS server. Specifically, the authors constructed a 19-dimensional feature vector broadly categorized into five (5) groups-DNS Answer based, Domain name based, spatial based, Network based, and Timing-based. Their feature vector is a collection of features extracted from various existing schemes. Dataset extracted from the responses obtained during DNS queries were used to train a C4.5 decision tree classifier. In order to find the best subset of feature that accurately detects a fast-flux botnet, 10-fold cross validation approach was employed on their dataset. The authors initially trained and validated the classifier with each feature subset separately and observed their corresponding accuracy. Finally, all features were merged in order to evaluate the corresponding accuracy. The results showed a robust detection features that is not prone to disguise attacks. Vranken & Alizadeh [37] proposed a domain name generation algorithm (DGA) using Term Frequency Inverse Document Frequency (TF-IDF). The authors used TF-IDF to measure the rates of the most occurring terms in domain names, and use these as features for their learning algorithms. The results of their comparison showed that deep learning model using TF-IDF features yielded the best results achieving high classification accuracy. Cucchiarelli et al. [38] proposed an algorithmically generated malicious domain names detection based on n-grams features. The proposed scheme represented the domain names through a set of features using 2–3-grams in a single unclassified and classified domain names. The authors used the Kullback-Leibner divergence and the Jaccard Index to evaluate similarity, and deployed state-of-the-arts machine learning algorithms to classify each domain. The results showed that the proposed scheme yielded a good level of accuracy and the scheme was able to classify previously unseen domains. Muhammad et al. [39] proposed a machine learning approach for early stage botnet detection. In this paper, the authors proposed an approach for early-stage botnet detection. The proposed approach first selects the optimal features using Principal Component Analysis (PCA) and Information Gain (IG) feature selection techniques. The selected features were then fed into machine learning classifiers for botnet detection. The results revealed that the proposed approach is accurate with low false alarms for an early stage botnet detection. Haq & Singh [40] developed a machine learning scheme for botnet detection. The approach divided the adapted dataset into two subsets and then applied k-means clustering on one set and j48 classification on the other set. The mean of the accuracy of k-means clustering and j48 classification approach (hybrid approach) was calculated for botnet detection. The hybrid approach was compared with the clustering and classification approach. The results showed that the hybrid approach is balanced for classification and clustering of botnet attacks. Randhawa et al. [41] proposed a security hardening of botnet detectors using generative adversarial networks (GANs). The authors used GAN to generate an extended dataset to the original train set to mitigate adversarial evasion attacks in botnet detection. The results showed that GANs can provide quality botnet detection samples compared to the traditional traffic generation methods. Although, generated samples not valid as real-life traffic and the scheme is vulnerable to evasion mechanisms. Stiawan et al. [42] proposed a dimensionality reduction approach for machine learning based botnet detection in Internet of Things (IoT). The authors used random projection method for dimensionality reduction to enhance state-of-the-arts machine learning methods to detect botnet in IoT. The experiment results showed random projection method combined with decision tree was able to detect IoT botnet at fast time and high accuracy. Hosseini et al. [43] proposed a Convolutional Neural Network and Long Short Term Memory (CNN-LSTM) for botnet detection. The objective of the study is to detect botnets based on neural network and the Negative Selection Algorithm (NSA). The authors used data wrangling method on the adapted dataset for data normalization. The normalized data is then fed into the NSA phase to reduce dimension. The authors then used a data scaling method based on the z-score algorithm and the scaled data used on CNN-LSTM algorithm for botnet detection. The results showed shorter training time and high detection accuracy. Lefoane et al. [44] proposed an optimized feature selection based on machine learning approach (decision tree, logistic regression, and support vector machine)) for botnet detection. The first part of the study is the use of a feature selection approach to remove less important features for botnet attack detection. The feature selection is based on the frequency of occurrence of the counted values to total instances in each of the features. The second part used the selected features to build machine learning classifiers for botnet detection. The proposed approach was tested on a standard IoT dataset and the results revealed that the proposed feature selection approach has enhanced the detection accuracy of the machine learning classifiers with low false alarm rate. Kolpe & Kshirsagar [45] presented a botnet detection approach using Bayes classifier. The authors used different filter-based feature selection schemes to select the most important features for botnet detection and the selected features used as input into a naive Bayes classifier. The results of the study showed that naïve Bayes classifier achieved the best accuracy for botnet detection using the CICIDS-2017 DoS dataset. For further studies on anomaly-based detection, literatures such as [46,47,48,49,50,51] are recommended.

2.3. DNS-based detection

The botnetmaster frequently masquerades phishing web servers as legitimate DNS and then redirecting users to the C&C server for malicious attacks. Therefore, DNS-based detection is based on the monitoring and detection of DNS traffic anomalies generated by a botnetmaster. The same anomaly-based detection algorithms can be utilized for DNS-based detection. For example, Alieyan et al. [52] proposed a DNS rule-based method for botnet detection. The method applied DNS query and response rules to detect any anomaly DNS query and response activities. The proposed method showed an improved accuracy and low false alarm rate for botnet detection. For further studies on DNS-based detection, literatures such as [53,54,55,56] are recommended.

2.4. Mining-based detection

Mining-based detection uses several data mining and machine learning algorithms to detect botnet C&C traffic. For example, Ibrahim et al. [57] proposed a multilayer framework that consists of filtering and detection modules for botnet detection. The filtering module was to filter and reduce the number of network features and group the network traffic in the minimum time interval. The detection module then used the reduced network features for botnet detection based on a multilayer framework. The result showed that the proposed method can detect botnet with good accuracy. For further studies on mining-based detection, literatures such as [58,59,60,61] are recommended.

2.5. Heuristic-based detection

Heuristic-based detection system (HBDS) employs a dynamic threshold score calculation based on some rules or statistical analysis of the network traffic for attack classification. HBDS used an adjustable threshold score to adjust to patterns in network traffic and reduce the false alarm rate [62]. The limitation of HBDS is the ability to correctly classify attacks based on the optimization of their threshold decision. For example, Ramachandran et al. [63] proposed a set of heuristics methods to detect DNS-based Black-hole List (DNSBL) lookup queries executed by a botmaster to know whether their bot have been blacklisted. The proposed heuristic method was able to provide counter intelligent measures to the methods used by botmasters to determine blacklisted bots. The goal of the proposed heuristic model is to detect in real-time DNSBL queries executed by botmasters from legitimate DNSBL queries. However, the proposed heuristic model cannot handle distributed DNSBL queries by botmaster.

Table 1 shows the detailed literature surveys and their limitations.

2.6. Motivation of the work

To address some of the limitations highlighted in the summary of related works, this study adapted a reliable and effective feature to deal with current evasion schemes adopted by botnetmasters. To overcome the identified problems, this study developed an improved Bot-FFX that adopts a KNN classifier rooted in rule-based GA consisting of three features: Standard deviation of Round-Trip-Time, Average Google Hits, and number of IP address over a time window. The main motivation for adopting the KNN is to benefit from the algorithm's high detection accuracy. The adoption of the listed features is to tackle the problem of evasion as they exhibit different behaviours for both FFB and legitimate domains. Additionally, a rule-based GA technique is introduced to reduce the time taken to differentiate between legitimate and botnet domains advertizing the same set of IP address over the time window.

2.7. Genetic algorithm

Genetic algorithms (GAs) used the computer to simulate the process of natural selection and evolution [64]. This notion originates from the "adaptive survival in natural organisms". GAs were first proposed by Goldberg & Holland [65] and have been successfully applied to the field of machine learning [66,67]. The algorithm begins with a randomly generated population of individual programs. The determination of how good an individual is in a population is rooted in their performance evaluation and based on various types of fitness measures. Then, at every iteration, a computerized genetic recombination and mixing is performed on the current population of individual programs to replace a less fit individual program by a high performing individual program. That is, a program with a low fitness value is removed and replaced by programs with high fitness value for the next computer iteration.

2.8. K-Nearest Neighbor

K-nearest neighbor (*k-NN*) is one of the simplest of all machine learning techniques. It is regarded as the traditional nonparametric technique in pattern recognition for the categorization of data [68,69]. It classify and assigns objects to the modal class of its predefined nearest neighbors. K represents the number of predefined nearest neighbors for an object and denotes a vital factor that determines the performance of the classifier. Different k-values will trigger different performances in the classifier and thus a considerably small positive integer is needed for k-value. A big and even number k-value can adversely affect the classification time and impact the prediction accuracy, while a small and odd number k-value can increase the prediction accuracy [70]. *k-NN* is termed instance-based learning because of its peculiarity compared to the inductive learning methods [71]. Thus, *k-NN* as an instance-based learning, does not include a model training phase, instead it determines the instances of input attributes and classifies new instances based on the determined k-nearest neighbor of the new instance.

2.9. Decision tree algorithm

Decision tree is a non-parametric supervised learning method commonly used for classification [72,73]. In other words, it does not require any prior assumptions regarding the type of probability distributions satisfied by the class or other attributes. The goal of a decision tree is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features [74,75]. In a decision tree, each leaf node is assigned a class label. The non-terminal nodes, which include the internal nodes and root node contain attribute test conditions to separate records that have different characteristics.

3. Methodology

In this study, a Bot-FFX was developed to accurately differentiate between legitimate and botnet domains. The developed system uses a rule-based GA technique and *k-NN* algorithm for fast flux botnet detection. The architecture of the Bot-FFX is described in Fig. 1.

The Bot-FFX is divided into modules which include –.

3.1. The extractor

The extractor is responsible for DNS queries on domains to:

- extract domains and IP addresses

Table 1
Summary of related works.

Author(s) and year	Method	Strength	Limitation
Provos [20]	Honeyd	-Security -Spam prevention	-Vulnerability to evasion mechanisms -Long detection time
Vrable et al. [26]	Potemkin	-Scalable -Security	-Vulnerability to attacks -Not completely scalable to denial-of-Service
Bajtoš et al. [27]	Honeynet	-Botnet attacks detection -Botnet analysis	-High dimensionality of feature set -Vulnerability to evasion mechanisms
Gu et al. [30]	IDS-Driven Dialog Correlation	-Accurate for botnet detection -Scalable	-Vulnerability to evasion mechanisms -High dimensionality of feature set
Xie et al. [31]	Spamming Botnets	-High detection accuracy -Low false alarm rate -Ability to detect frequent domain modifications	-High dimensionality of feature set -Vulnerability to evasion mechanisms
Behal et al. [32]	Signature-based botnet detection	-Dynamic rule generation for botnet detection -Real-time monitoring and detection	-It requires access to a current database of attack signatures -Vulnerability to evasion mechanisms
Chen et al. [33]	Ensemble anomaly-based method	-Increased accuracy -Reduced false alarms	-High dimensionality of feature set -Vulnerability to evasion mechanisms
Alieyan et al. [52]	DNS rule-based method	-High detection accuracy -Low false alarm rate	-It cannot detect Peer-to-Peer botnets -Vulnerability to evasion mechanisms
Ibrahim et al. [57]	Multilayer framework	-It can detect botnet with good accuracy -Low false-negative rate	-Long processing and detecting time -Reduced performance while clustering decentralized botnets
Ramachandran et al. [63]	Heuristic method	-It can detect DNS-based Black-hole List (DNSBL)	-It cannot handle distributed DNSBL queries by botmaster -High dimensionality of feature set
Martinez-Bea et al. [34]	SVM classifier	-Resilience of the scheme to evasion techniques	-False positives -False negatives
Zhao & Traore [36]	Decision tree	-Reduced false positives -Not prone to disguise attacks	-Does not provide real time detection
Celik & Oktug [14]	DCA-based on n-grams features	-Robust detection features. -Not prone to disguise attacks	-Does not provide real time detection -Computationally expensive scheme to implement
Vranken & Alizadeh [37]	DCA-based on TF-IDF features	-High classification accuracy -Usage of TF-IDF for feature selection	-Lack of comparative analysis with related works -Vulnerability to evasion mechanisms -High dimensionality of feature set
Cucchiarelli et al. [38]	DCA-based on n-grams features	-Effective classification of previously unseen domains -High classification accuracy	-Long processing and detecting time -High dimensionality of feature set
Muhammad et al. [39]	Machine learning approach	-Optimal features selection -High detection accuracy	-It cannot detect decentralized P2P based botnets -Vulnerability to evasion mechanisms
Haq & Singh [40]	Hybrid machine learning approach	-High clustering accuracy	-Vulnerability to evasion mechanisms
Randhawa et al. [41]	Generative adversarial network	-Generate quality botnet detection samples -Robust to data imbalance -Decrease false positives	-Generated samples not valid as real-life traffic -Vulnerability to evasion mechanisms
Stiawan et al. [42]	Random projection method based on machine learning method	-High detection accuracy -Low false positive rate -Fast detection time	-Vulnerability to evasion mechanisms
Hosseini et al. [43]	CNN-LSTM	-Shorter training time -High accuracy	-Vulnerability to evasion mechanisms
Lefoane et al. [44]	Feature selection-based machine learning approach	-Good accuracy -Low false alarm	-Vulnerability to evasion mechanisms
Kolpe & Kshirsagar [45]	Bayes classifier	-Optimal feature selection -High detection accuracy	-Vulnerability to evasion mechanisms

- extract Round Trip Times (RTT) of associated IP addresses
- extract the Google Hits of each IP address.

3.2. The filter

The filter mechanism of the Bot-FFX uses the blacklist and whitelist filtering concepts to classify the incoming domains. The actions executed by the filter mechanism are defined as follows:

- Deny Access to domain, if $(D \in B)$ //Botnet
- Grant Access to domain, if $(D \in W)$ // Benign
- Send domain to resolver, if $(D \notin \{B \cup W\})$

Where, $D =$ Domain name, $B =$ Blacklist of known botnet domains and $W =$ Whitelist of known benign domains.

3.3. The resolver

The filter mechanism can send the unclassified domains to the resolver. The resolver extracts all IP addresses associated with the domain at its Time-To-Live (TTL) within a time frame of 10 min. The operations of the resolver are defined in three phases:

Name-server resolution: The set of authoritative name servers of the domain is extracted. The output of the operation results in a

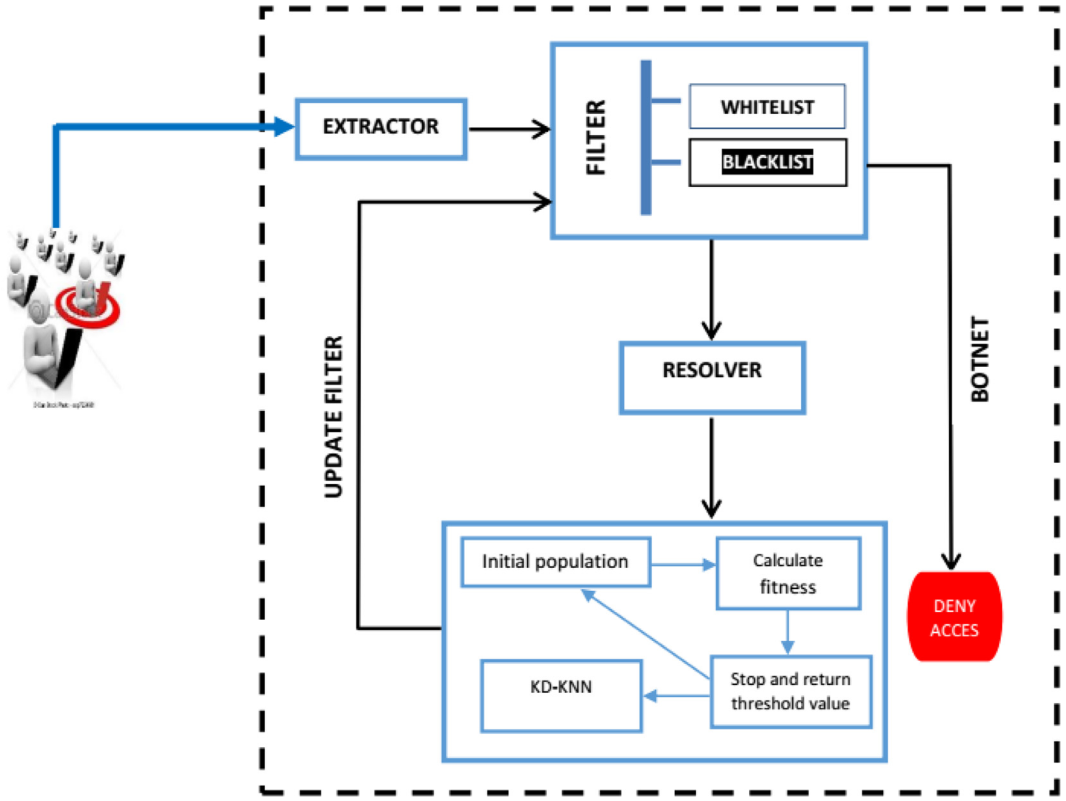


Fig. 1. Bot-FFX architecture.

set of name servers $X\{x_1, x_2, \cdots, x_n\}$ such that $|X| \geq 2$. The query syntax used by the resolver for name-server resolution is -

$$dig + shortNSdomain \tag{1}$$

where, $+short =$ command to restrict output to only name-server records, $NS =$ command that specifies the request for name server records and $domain =$ the domain whose name server record is needed

Time-To-Live resolution: This involves the extraction of the TTL of the domain to define the moving window for IP address resolution. The query syntax used by the resolver for TTL extraction is as follows:

$$dig@domain + tracettlid \tag{2}$$

where, $@domain =$ the domain name whose TTL value is needed, $+trace =$ indicates the downward traversal from the root name server to the authoritative name server of the domain name and $ttlid =$ indicator specifying the request for TTL value

IP Address resolution: For each TTL window, the set of IP address mapped to the domain is extracted for a specified time frame of 10 min. The query syntax for IP address extraction is as follows:

$$dig@NSdomain \tag{3}$$

where, $@NS =$ a name server x_i such that $x_i \in X$, $domain =$ the domain whose IP address are needed. This query returns a set of IP address $Y\{y_1, y_2, \cdots, y_n\}$ such that $|Y| \geq 1$. The specified time frame of 10 min for resolution is required to enable the accumulation of several IP addresses of the domain. Researchers such as Knysz et al. [18], and Hsu et al. [17] revealed that the IP addresses of botnet domains increased rapidly after the first DNS query. This phenomenon is adopted by botnetmasters to evade real-time detection solutions that focus on 1 DNS query for the accumulation of domain IP addresses.

3.4. The detection

This module is responsible for classification in cases where the filter mechanism cannot determine the benign nature of the domain. It comprises of three modules, namely, SDRTT, AGH, and detector.

Standard Deviation of Round-Trip-Times (SDRTT): This part of the detection computes the standard deviation of Round-Trip-Time (SDRTT) on the set of IP addresses of the domain. The distances between the system and the bots are expected to be quite large and the SDRTT will be a relatively large value. SD-RTT is computed as:

$$\sigma RTT = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (t_i - f)^2} \tag{4}$$

Where, t_i = the RTT between the system and IP address i, $n =$ the total number of IP addresses, f = the mean of the total RTT.

Average Google Hits (AGH): This part of the detection computes the average number of hits returned by querying a search engine using all IP addresses associated with the domain. The average Google hit for each domain is computed as –

$$AGH = \frac{\sum_{i=1}^{n} GoogleHit(i)}{n} \tag{5}$$

Where, $i =$ the ith IP address of the domain and $n =$ the total number of IP address associated with the domain.

Detector: The C4.5 decision tree was built to automatically generate the detection rules for the K-NN. The detector computes Genetic Threshold Value (GTV) for IP addresses using the GA-KD-*k-NN* algorithm. It performs detection based on the decision tree rules and the GTV using *k-NN* distance measure as in (4). The KNN used KD tree for the search of K-values. The Manhattan distance was adapted because of its suitability for high dimensionality in data. The *k-NN* add IP address to the whitelist if the IP address is less than the *k-NN* distance value (D_{knn}) as computed in equation (8). Otherwise, the IP address is added to the blacklist. Once an IP is added in the blacklist, a further request from the IP is automatically rejected by the firewall filtering unit of the detector. The fitness can be computed as –

$$fitness = \sum_{i=0}^{n} match \times weight_i \tag{6}$$

where, n is the number of genes present in each chromosome. In this case, the gene means properties to be checked for each network domain, where each network domain is equal to a chromosome. The properties which might be considered as genes for a network domain includes: source IP address, destination IP address, source port number, destination port number, size of packet, number of hops between the source and destination, TTL, packet type, payload, checksum, sequence number etc.

$$GTV = \frac{bestfitness - worstfitness}{totalnumberoffitness} \tag{7}$$

$$D_{knn} = \sum_{i=1}^{k} GTV \times |x.value(i) - y.value(i)| \tag{8}$$

where, $x.value$ = input query, $y.value$ = known data point closest to the input query, k = the k-value. Table 2 shows the overall *GA-KD-KNN algorithm.*

4. Implementation, results and discussion

The implementation was carried out on an Intel(R) Pentium(R) CPU N3710 @ 1.60 GHz with 4 GB RAM running on Windows 10 operating system. The experimentation for the developed Bot-FFX was implemented with JAVA, NetBeans 8, MySQL, JavaML, Jgap, Weka J48, JKDTreeKNN and LibSVM API, Jsoup Java API for Google search query, JFreeChart Java API and Microsoft Excel for chart development.

4.1. Description of dataset

The dataset adopted for the evaluation of the developed Bot-FFX consists of 2,000 benign domains collected from Alexa website and 1630 malicious domains obtained from various malware reporting media such as Domain Name System Black List (DNSBL), Zeus tracker monitor, and DNS Black Hole (DNSBH) project. For each domain, several queries were performed to extract the specified

Table 2
The GA-KD-KNN algorithm.

```
Step 1.   Generate the initial population, $n$
Step 2.   Fitness value estimation
          for $i := 1$ to $n$
              Apply elitism by copying the best GA individual to the next generation
              Apply tournament selection
              Apply uniform crossover
              Apply mutation operator
              Calculate the fitness function for each individual using equation (6)
              Compute GTV using equation (7)
          end
Step 3.   Select the best individual that has the highest fitness value
Step 4.   Apply k-NN based on GTV in 7
          for $i := 1$ to $n$
              Compute
              $D_{knn} = \sum_{i=1}^{k} GTV \times |x.value(i) - y.value(i)|$
              Compute the group of nearest neighbors
          end
Step 5.   //k -d tree nearest neighbor search
          $T \leftarrow T1 + T2$
          $T1 \leftarrow$ first side of the splitting plane
          $T2 \leftarrow$ other side of the splitting plane
          While ($current\_node \leftarrow root\_node.T1$)
          do
              if $search\_point(current\_node) \leq current\_node\,value\ (D_{knn})$ then
                  $search\_first := left$
              else
                  $search\_first := right$
              end
              if $leaf\_node$ is reached then
                  $current\_best := node\_point$
              end
          end
          Repeat Step 5 for T2
          if ($current\_best$ exist in $T2$)
          $current\_best := node\_point$ end
          Return $current\_best$
Step 6.   //return the majority class label for unknown domain
          if $IP\_address < D_{knn}\,value$ then
              add IP address to whitelist
          else
              add IP address to blacklist
          end
Step 7.   Block further request from the blacklisted IP addresses
```

features (Standard deviation of Round-Trip-Time, Average Google Hits, and Number of IP address over a time window). The collected datasets were also used by other authors in the literature for performance evaluation. Table 3 shows the summary of the adopted datasets.

4.2. Experimentation

In the development of Bot-FFX, a number of experiments were conducted. The experiments are described and analyzed in this section. When tested and used for the experiment, the Bot-FFX was observed to successfully perform the following tasks:

(i) Executing DNS query on domains to extract IP addresses
(ii) Extracting Round Trip Times (RTT) of associated IP addresses
(iii) Extracting the Google Hits of each IP address
(iv) Calculate GTV for each IP address
(v) Build the C4.5 decision tree
(vi) Build the *k-NN* detector
(vii) Update the blacklist based on the detection

In order to generate the rules for the k-NN detector, the C4.5 decision tree was used. The result of this decision tree can be described as set of rules, encapsulating the adopted feature set, that are used during detection. The rules (e.g. rules 001 to 004) generated by the decision tree are then extended by the distance value based on the genetic threshold value (e.g. rules 005 to 006) for botnet detection. The rule representation of the C4.5 decision tree in Fig. 2 is as follows:

001: IF $att1 \leq 129.51$ THEN *domain* is benign.

002: IF $att1 > 129.51$ AND $att0 \leq 8$ THEN *domain* is benign.

003: IF $att1 \geq 129.51$ AND $att0 > 8$ AND $att2 \leq 866$ THEN *domain* is benign.

004: IF $att1 \geq 129.51$ AND $att0 > 8$ AND $att2 > 866$ THEN *domain* is botnet.

005: 003: IF $att1 \geq 129.51$ AND $att0 > 8$ AND $att2 \leq 866$ AND att3 > 8 THEN *domain* is benign.

006: 003: IF $att1 \geq 129.51$ AND $att0 > 8$ AND $att2 \leq 866$ AND att3 less than 8 THEN *domain* is botnet.

Where, $att0 =$ number of IP address associated with the domain, $att1 =$ standard deviation of round-trip time for the domains' set of IP addresses, $att2 =$ average google hits for the domains' set of IP addresses and $att3 =$ distance value based on the GTV.

4.3. Result analysis and evaluation

The datasets obtained after extracting the specified features were adopted for performance evaluation using benchmarked performance metrics.

Table 3
Summary of Datasets.

Datasets	Instances	Category
Alexa[1]	2000	Benign
DNSBL project[2]	110	FFSN
ZeusTracker monitor[3]	20	FFSN
DNSBH[4]	1500	FFSN

[1] http:// https://www.alexa.com.
[2] http:// htttp://dnsbl.abuse.ch/fastfluxtracker.php.
[3] http:// https://zeus.abuse.ch/monitor.php?filter = level 5.
[4] http:// https://www.malwaredomains.com.

4.3.1. Performance metrics

The performance of the developed Bot-FFX was measured based on standard metrics which are False Positive Rate (FPR), False Negative Rate (FNR), True Positive Rate (TPR), True Negative Rate (TNR) and Overall Accuracy (OA). To evaluate the Bot-FFX, the dataset was split into 50% training and 50% testing data, which was also adopted by Lin et al. [3] and Hsu et al. [17]. The training and testing data contained 1000 benign and 815 botnet domains, respectively. The results obtained in this study were compared with GRADE in Lin et al. [3], FFD in Hsu et al. [17], MLP in Ibrahim et al. [57], Logistic regression in Palaniappan et al. [76], Random forest in Sivaguru et al.[77], and Random forest in Patsakis & Casino [78].

4.3.2. Bot-FFX testing results

The testing dataset was tested with three machine learning algorithms namely: Genetic Algorithm and K-Nearest Neighbors (GA-*k-NN*), *k-NN* and Support Vector Machines (SVM). The justification for this evaluation is to determine the learning algorithm that best suits the detector. The SVM algorithm was implemented with the aid of LibSVM [79]. The performance of these algorithms was evaluated in terms of False Positive Rate (FPR), False Negative Rate (FNR), True Positive Rate (TPR), True Negative Rate (TNR) and Accuracy.

Table 4 shows the testing results. The results from Table 4 revealed that GA- *k-NN*, *k-NN* and SVM algorithms provided an overall accuracy of 99.178%, 96.362% and 98.741% respectively. These results informed the decision to adopt the GA-*k-NN* as the most suitable learning algorithm for the detector module. Table 5 and Table 6 show the performance comparison of the developed GA-*k-NN* and traditional *k-NN* on benign and botnet domains, respectively. Table 5 revealed that GA-*k-NN* provided OA of 96.858% on benign domain and OA of 99.178% on the botnet domain. Similarly, Table 6 revealed that *k-NN* provided OA of 98.706% on benign domain and OA of 96.362% on botnet domain. Table 7 revealed that *SVM* provided OA of 96.858% on benign domain and OA of 98.741% on botnet domain. These results showed that the developed GA-*k-NN* is better for botnet detection when compared to the traditional *SVM* and *k-NN* algorithms.

4.3.3. Analysis of feature set

The high performance of Bot-FFX can be attributed to the efficacy of the adopted attributes for differentiating between botnet and benign domains. To support the above statement, a plot of the IP address utilization, Standard Deviation of Round-Trip Time (SDRTT) and the Average Google Hits (AGH) for each domain category was carried out on the adopted dataset (Figs. 3 to 5). In Fig. 3, it is apparent that the distribution of IP address utilization varies for each domain category. The bar chart shows that about 93.2% of benign domains advertised less than 9 IP addresses while botnet domains advertised a minimum of 9 IP addresses during a total of 10 min of IP resolution duration. The justification for the 10 min time frame was due to the fact that many legitimate domains use short TTL values.

A critical look at Fig. 4 revealed that the SDRTT of botnet domain is much higher than that of benign domain. A large number of benign domain exhibited SDRTT values lower than 200 ms. In contrast, only a few botnet domains exhibit such behavior and this is due to the geographical dispersion of the set of IP address adopted by botnets. Similarly, Fig. 5 revealed that the Google footprint for botnet domains is much higher than for that of benign domains. A large number of benign domains exhibited AGH values less than 10,000. In contrast, most botnet domains exhibited AGH values above 10,000.

The contributions of this study to science are –.

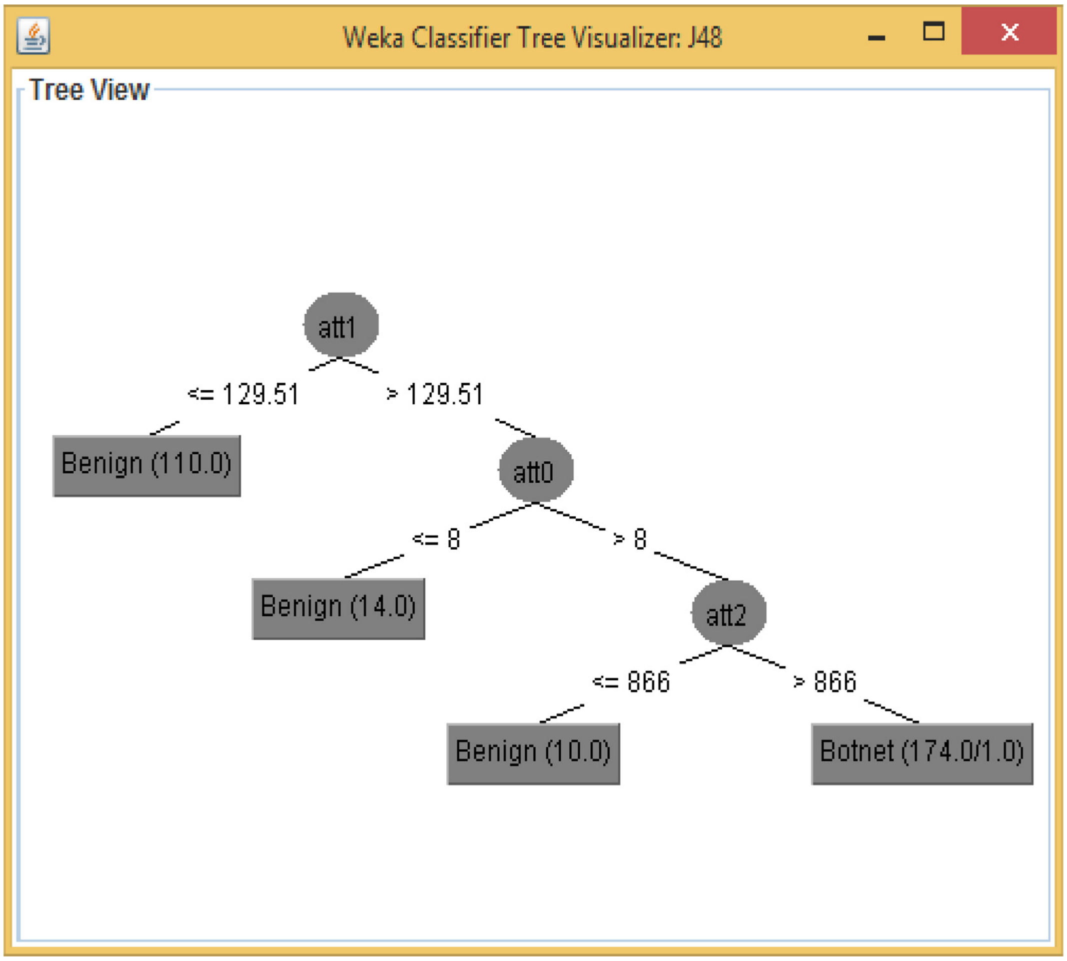


Fig. 2. C.45 decision tree based on the adopted feature set.

Table 4
Performance Comparison of GA-k-NN, k-NN and SVM.

Algorithm	TPR	TNR	FPR	FNR	OA (%)
GA-*k-NN*	808	793	7	7	99.178%
k-NN	784	769	31	31	96.362%
SVM	813	798	2	2	98.741%

Table 5
Performance Comparison of GA-k-NN on Benign and Botnet domains.

	BENIGN DOMAINS				BOTNET DOMAINS			
ItrNo	NoTI	TPR	FPR	OA (%)	NoTI	TPR	FPR	OA (%)
1	1000	982	18	96.673	815	795	20	97.653
2	1000	962	38	92.976	815	808	7	99.178
3	1000	983	17	96.858	815	800	15	98.239
4	1000	978	22	95.934	815	807	8	99.061
5	1000	983	17	96.858	815	799	16	98.122
6	1000	981	19	96.488	815	805	10	98.826
7	1000	982	18	96.673	815	802	13	98.474
8	1000	980	20	96.303	815	804	11	98.709
9	1000	983	17	96.858	815	801	14	98.357
10	1000	981	19	96.488	815	803	12	98.592

ItrNo – Iteration Number, NoTI **–** Number of Test Instance.

Table 6
Performance Comparison of k-NN on Benign and Botnet domains.

ItrNo	BENIGN DOMAINS				BOTNET DOMAINS			
	NoTI	TPR	FPR	OA (%)	NoTI	TPR	FPR	OA (%)
1	1000	968	32	94.085	815	772	43	94.953
2	1000	949	51	90.573	815	784	31	96.362
3	1000	980	20	96.303	815	769	46	94.601
4	1000	977	23	95.749	815	775	40	95.305
5	1000	985	15	97.227	815	767	48	94.366
6	1000	983	17	96.858	815	772	43	94.953
7	1000	993	7	98.706	815	760	55	93.545
8	1000	988	12	97.782	815	765	50	94.132
9	1000	992	8	98.521	815	755	60	92.958
10	1000	990	10	98.152	815	762	53	93.779

ItrNo – Iteration Number, NoTI – Number of Test Instance.

Table 7
Performance Comparison of SVM on Benign and Botnet domains.

ItrNo	BENIGN DOMAINS				BOTNET DOMAINS			
	NoTI	TPR	FPR	OA (%)	NoTI	TPR	FPR	OA (%)
1	1000	980	20	95.562	815	793	22	96.542
2	1000	959	41	91.754	815	813	2	98.741
3	1000	981	19	95.635	815	803	12	97.128
4	1000	976	24	94.723	815	810	5	98.152
5	1000	981	19	95.747	815	797	18	97.234
6	1000	979	21	95.379	815	802	13	97.715
7	1000	980	20	95.562	815	800	15	97.363
8	1000	978	22	95.212	815	802	13	97.617
9	1000	981	19	95.747	815	799	16	97.246
10	1000	979	21	95.379	815	801	14	97.481

ItrNo – Iteration Number, NoTI – Number of Test Instance.

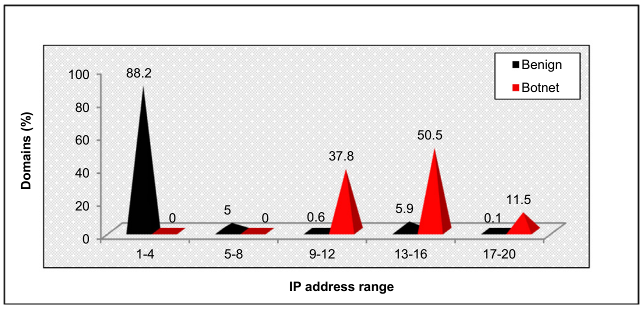


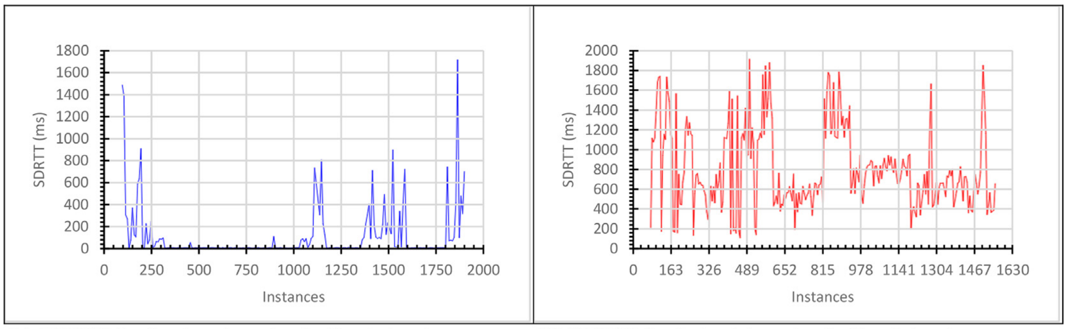
Fig. 3. IP Address utilization for benign and botnet domains.

- The development of a genomic *k-NN* Fast-Flux Botnet detection system for attacks classification.
- The use of reliable features to enhance the detection of fast flux botnets.
- The study also introduced a rule representation method for whitelisting domains.

4.4. Discussion

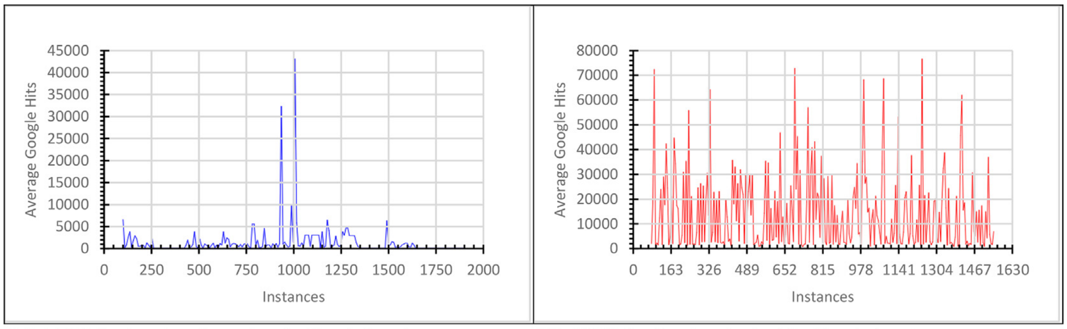
4.4.1. Summary of test results

Table 4 shows the performance comparison of GA-*k-NN*, *k-NN*, and SVM in terms of TPR, TNR, FPR, FNR, and OA. The OA column showed that GA-*k-NN* has the highest accuracy of 99.178% compared to the other well-known algorithms of *k-NN* and SVM with OA of 96.362% and 98.741% respectively. The increase in accuracy rate for GA-*k-NN* was due to the use of an optimization method in GA and a rule representation method that guide the selection of best solutions. The increase in accuracy of the GA-*k-NN* is also slightly dependent on the use of Standard deviation of Round-Trip-Time, Average Google Hits, and Number of IP address as the adopted features to enhance the detection of fast flux botnets.



(a) SDRTT for benign domains in the dataset *(b) SDRTT for botnet domains in the dataset*

Fig. 4. SDRTT for benign and botnet domains in the dataset.



***(a)** AGH for benign domains in the dataset* ***(b)** AGH for benign botnet in the dataset*

Fig. 5. AGH for benign and botnet domains in the dataset.

In order to improve predictions and remove the problem of unbalanced data in classification, k-fold (k = 5) cross-validation was used. This study randomly divided the training data into 5 equal sized subsets. A single subset was applied to test the developed method and the remaining 4 subsets were used as the training data. The comparative test results were obtained on the same computational platform.

4.4.2. Comparison of GA-k-NN on benign and botnet domains

Table 5 shows the test experiments of GA-*k-NN* conducted on both the benign and botnet domains. The results of the test showed that GA-*k-NN* provided OA of 96.858% on the benign domain and OA of 99.178% on the botnet domain. The high performance of GA-*k-NN* can be attributed to the efficacy of the adopted attributes for differentiating between botnet and benign domains.

4.4.3. Comparison of k-NN and SVM on benign and botnet domains

Table 6 shows the test experiments of *k-NN* conducted on both the benign and botnet domains. The results of the test showed that *k-NN* provided OA of 98.706% on the benign domain and OA of 96.362% on the botnet domain. The results of the *k-NN* can be attributed to the algorithm's high detection accuracy. Table 7 shows the test experiment of SVM conducted on both the benign and botnet domains. The results revealed that *SVM* provided OA of 96.858% on benign domain and OA of 98.741% on botnet domain. The high performance of GA-*k-NN* compared to *k-NN and SVM* can be attributed to the adoption of the used features to tackle the problem of botnet master evasion as they exhibit different behaviours for both botnet and benign domains. Additionally, GA-*k-NN* was better than *k-NN and SVM* due to the introduction of the GA rooted in a rule representation method to reduce the time taken to differentiate between legitimate and botnet domains advertizing the same set of IP address over the time window.

4.4.4. Overall performance of Bot-FFX

The developed Bot-FFX showed overall performance with OA of 99.178%, FPR of 0.8%, and FNR of 0.8%. This result shows the positive contribution of Bot-FFX for botnet attack classification with reduced false alarm rate. The reduced false alarm rate was due to the ability of the Bot-FFX to clearly differentiate between botnet and benign domains using the adopted attributes.

4.4.5. IP address utilization for benign and botnet domains

Fig. 3 shows the plot of IP Address utilization for benign and botnet domains. Previous results has established the high performance of Bot-FFX. To further justify the high performance of Bot-FFX, a plot of the IP address utilization for each domain category was carried out on the adopted dataset. In Fig. 3, it is apparent that the distribution of IP address utilization varies for each domain category. The bar chart shows that about 93.2% of benign domains advertised less than 9 IP addresses while botnet domains advertised a minimum of 9 IP addresses during a total of 10 min of IP resolution duration. The justification for the 10 min time frame was due to the fact that many legitimate domains use short TTL values.

4.4.6. SDRTT for benign and botnet domains in the dataset

Fig. 4 shows the plot of SDRTT for benign and botnet domains in the dataset. In order to support the results for the high performance of the Bot-FFX, a plot of the Standard Deviation of Round-Trip Time (SDRTT) for each domain category was carried out on the adopted dataset. A critical look at Fig. 4 revealed that the SDRTT of botnet domain is much higher than that of benign domain. A large number of benign domain exhibited SDRTT values lower than 200 ms (Fig. 4a). In contrast, only a few botnet domains exhibit such behavior (Fig. 4b) and this is due to the geographical dispersion of the set of IP address adopted by botnets.

4.4.7. AGH for benign and botnet domains in the dataset

Fig. 5 shows the plot of AGH for benign and botnet domains in the dataset. Fig. 5 revealed that the Google footprint for botnet domains is much higher than for that of benign domains. A large number of benign domains exhibited AGH values less than 10,000 (Fig. 5a). In contrast, most botnet domains exhibited AGH values above 10,000 (Fig. 5b). This is because malicious domains are one of the key domains that attackers used to perpetrate malicious actions over the Internet. Hence, the botnet domains exhibiting AGH values above 10,000 compared to their benign counterpart.

4.4.8. Benchmarking Bot-FFX with related works

Table 8 compared the performance of the developed Bot-FFX with GRADE in Lin et al. [3], FFD in Hsu et al. [17], MLP in Ibrahim et al. [57], Logistic regression in Palaniappan et al. [76], Random forest in Sivaguru et al.[77], and Random forest in Patsakis & Casino [78]. This study implemented and tested the methods under comparison on the same datasets and computational platform. The evaluation results obtained from the comparison, indicated that

Table 8
Comparison of Bot-FFX with related works.

Detection Approaches	OA (%)	FN rate (%)	FP rate (%)
Lin et al. [3]	96.5	1.6	1.9
Hsu et al. [17]	93.4	1.5	0.7
Ibrahim et al. [57]	98.7	0.9	0.8
Palaniappan et al. [76]	91.5	1.6	1.8
Sivaguru et al. [77]	98.4	1.5	1.7
Patsakis & Casino [78]	98.5	1.4	1.5
Bot-FFX	**99.2**	**0.8**	**0.8**

the developed Bot-FFX is better than Lin et al. [3], Hsu et al. [17], Ibrahim et al. [57], Palaniappan et al. [76], Sivaguru et al. [77], and Patsakis & Casino [78] respectively. The main advantage of Bot-FFX over the other related implemented systems is the requirement of a set of three (3) features depending on the filter module decision. This requirement reduces the time needed to train the GA-*k-NN* classifier to few minutes. Besides, the practical deployment of Bot-FFX will result to the detection of a botnet domain within 20 min of deployment. The developed Bot-FFX is also robust to dynamic environment since genetic algorithm can varies in accordance to the current situation. The results of the developed Bot-FFX also showed an optimized solution due to the fact that genetic algorithm always produce the best result.

5. Conclusion and future work

The evolution of the Internet and the network of anonymous users unaware of the need of Internet security has led to Fast-Flux Botnets as a means of exploitation by money-driven cyber-criminals. Fast-Flux Botnet is a prevalent security challenge as it provides botnetmasters the opportunity to remotely control the network of infected hosts. In the literature, a number of solutions have been developed to reduce this menace. However, these solutions are still limited in detection accuracy due to the ineffectiveness of the adopted feature set. In this study, Bot-FFX was developed with this limitation in mind, and this resulted in the utilization of a rule-based GA scheme and three effective features that are fed into a *k-NN* built on the decision tree and KD tree algorithms. Bot-FFX was tested on a public dataset and benchmarked with GRADE in Lin et al. [3], FFD in Hsu et al. [17], MLP in Ibrahim et al. [57], Logistic regression in Palaniappan et al. [76], Random forest in Sivaguru et al. [77], and Random forest in Patsakis & Casino [78]. The evaluation results showed that the developed Bot-FFX is better in detection accuracy and achieved the best false negative rate of 0.8% compared to other related implemented methods. In the future, a machine learning classifier in combination with genetic algorithm will be deployed in the extractor module to produce the GTV and another machine learning classifier based on the GTV will be deployed in the detector module.

CRediT authorship contribution statement

Femi Emmanuel Ayo: Conceptualization, Methodology. **Joseph Bamidele Awotunde:** Data curation. **Sakinat Oluwabukonla Folorunso:** Visualization, Investigation. **Matthew O. Adigun:** Supervision. **Sunday Adeola Ajagbe:** Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Zhang, L., Shui, Y., Di, W. & Paul, W. 2011. A Survey on Latest Botnet Attack and Defense. In: Proceedings of International Joint Conference of IEEE Trustcom-11/IEEE ICESS-11/FCST-11. Changsha China pp.53-60.

[2] Butt UJ, Richardson W, Nouman A, Agbo HM, Eghan C, Hashmi F. Cloud and Its Security Impacts on Managing a Workforce Remotely: A Reflection to Cover Remote Working Challenges. In: Cybersecurity, Privacy and Freedom Protection in the Connected World. Cham: Springer; 2021. p. 285–311.

[3] Lin H-T, Lin Y-Y, Chiang J-W. Genetic-based Real-time Fast-Flux Service Networks Detection. J. Comput. Networks: Elsevier 2013;57(2):501–13.

[4] Holz, T., Gorecki, C., Rieck, K. & Freiling F.C. 2008. Detection and mitigation of fast-flux service networks. In: Proceedings of the 15th Network and Distributed System Security Symposium. San Diego USA.

[5] Lallie HS, Shepherd LA, Nurse JR, Erola A, Epiphaniou G, Maple C, et al. Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. Comput Secur 2021;105:102248.

[6] Stalmans, E. & Irwin, B. 2011. A framework for DNS based detection and mitigation of malware infections on a network. In: Proceedings of the 10th IEEE International Conference on Information Security. Johannesburg South Africa pp.1-8.

[7] Khari M, Dalal R, Rohilla P. Extended paradigms for botnets with WoT applications: a review. Smart Innovation of Web of Things 2020:105–22.

[8] Aruna J, Shyry SP. Survey on Artificial Intelligence Based Resilient Recovery of Botnet Attack. In: In *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE; 2021. p. 1–8.

[9] Firat I. Inevitable Battle Against Botnets. In: Management Association IR, editor. *Research Anthology on Combating Denial-of-Service Attacks*:. IGI Global; 2021. p. 1–19.

[10] Hsu, C-H., Huang, C-Y. & Chen, K-T. 2010. Fast-flux bot detection in real time. In: Proceedings of the 13th International Conference on Recent Advances in Intrusion Detection (RAID). Springer Berlin Heidelberg pp.464–483.

[11] Passerini E, Roberto P, Lorenzo M, Danilo B. FluXOR: Detecting and Monitoring Fast-Flux Service Networks. Berlin: Detection of Intrusions and Malware, and Vulnerability Assessment, Springer; 2008. p. 186–206.

[12] Ahmad R, Alsmadi I. Machine learning approaches to IoT security: A systematic literature review. Internet of Things 2021;100365.

[13] Kumar P, Gupta GP, Tripathi R. Toward design of an intelligent cyber attack detection system using hybrid feature reduced approach for iot networks. Arab J Sci Eng 2021;46(4):3749–78.

[14] Celik, Z.B. & Oktug, S. 2013. Detection of Fast-Flux Networks Using Various DNS Feature Sets. In: Proceedings of IEEE Symposium on Computers and Communications (ISCC). Split Croatia pp.000868 – 000873.

[15] Ashraf J, Keshk M, Moustafa N, Abdel-Basset M, Khurshid H, Bakhshi AD, et al. IoTBoT-IDS: A Novel Statistical Learning-enabled Botnet Detection Framework for Protecting Networks of Smart Cities. Sustain Cities Soc 2021;103041.

[16] Zhang J, Ling Y, Fu X, Yang X, Xiong G, Zhang R. Model of the intrusion detection system based on the integration of spatial-temporal features. Comput Secur 2020;89:101681.

[17] Hsu F-H, Wang C-SfC-H, Tso C-K, Chen L-H, Lin S-H. Detect Fast-Flux Domains Through Response Time Differences. IEEE J Sel Areas Commun 2014;32 (10):1947–56.

[18] Knysz, M., Hu, X. & Shin, K. 2011. Good guys vs. bot guise: Disguise attacks against fast-flux detection systems. In: Proceedings of 2011 IEEE INFOCOM. Shanghai China pp.1844-1852.

[19] Zhu Z, Lu G, Chen Y, Fu ZJ, Roberts P, Han K. Botnet research survey. In: In *2008 32nd Annual IEEE International Computer Software and Applications Conference*. IEEE; 2008. p. 967–72.

[20] Provos, N. 2004. A Virtual Honeypot Framework. In USENIX Security Symposium (Vol. 173, No. 2004, pp. 1-14).

[21] Choo KKR. Zombies and botnets. Trends Issues Crime Crim Justice 2007;333:1–6.

[22] Dagon, D., Zou, C. C., & Lee, W. 2006. Modeling Botnet Propagation Using Time Zones. In NDSS (Vol. 6, pp. 2-13).

[23] Zeidanloo HR, Shooshtari MJZ, Amoli PV, Safari M, Zamani M. A taxonomy of botnet detection techniques. In *2010 3rd International Conference on Computer Science and Information Technology*, Vol. 2. IEEE; 2010. p. 158–62.

[24] Wang TZ, Wang HM, LIU B, Shi PC. Some critical problems of botnets. Chinese J Comput 2012;35(6):1192–208.

[25] Alparslan E, Karahoca A, Karahoca D. BotNet detection: Enhancing analysis by using data mining techniques. Advances in Data Mining Knowledge Discovery and Applications 2012;Vol. 349.

[26] Vrable M, Ma J, Chen J, Moore D, Vandekieft E, Snoeren AC, et al. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. SIGOPS Oper Syst Rev 2005;39(5):148–62.

[27] Bajtoš, T., Sokol, P., & Mézešová, T. 2018. Virtual honeypots and detection of telnet botnets. In *Proceedings of the Central European Cybersecurity Conference 2018* (pp. 1-6).

[28] Kumar P, Gupta GP, Tripathi R. Design of anomaly-based intrusion detection system using fog computing for IoT network. Autom Control Comput Sci 2021;55(2):137–47.

[29] Kumar, P., Tripathi, R., & P. Gupta, G. 2021d. P2IDF: a privacy-preserving based intrusion detection framework for software defined Internet of Things-fog (SDIoT-Fog). In Adjunct Proceedings of the 2021 International Conference on Distributed Computing and Networking (pp. 37-42).

[30] Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., & Lee, W. 2007. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *USENIX Security Symposium* (Vol. 7, pp. 1-16).

[31] Xie Y, Yu F, Achan K, Panigrahy R, Hulten G, Osipkov I. Spamming botnets: signatures and characteristics. ACM SIGCOMM Computer Communication Review 2008;38(4):171–82.

[32] Behal S, Brar AS, Kumar K. Signature-based botnet detection and prevention. In: In *Proceedings of International Symposium on Computer Engineering and Technology*. p. 127–32.

[33] Chen T, Zhou G, Liu Z, Jing T. A novel ensemble anomaly based approach for command and control channel detection. In: In *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*. p. 74–8.

[34] Martinez-Bea, S., Castillo-Perez, S., & Garcia-Alfaro, J. 2013. Real-time malicious fast-flux detection using DNS and bot related features. In 2013 Eleventh Annual Conference on Privacy, Security and Trust (pp. 369-372). IEEE.

[35] McGrath DK, Kalafut A, Gupta M. Phishing infrastructure fluxes all the way. IEEE Secur Priv 2009;7(5):21–8.
[36] Zhao, D., & Traore, I. 2012. P2P botnet detection through malicious fast flux network identification. In 2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (pp. 170-175). IEEE.
[37] Vranken H, Alizadeh H. Detection of DGA-Generated Domain Names with TF-IDF. Electronics 2022;11(3):414.
[38] Cucchiarelli A, Morbidoni C, Spalazzi L, Baldi M. Algorithmically generated malicious domain names detection based on n-grams features. Expert Syst Appl 2021;170:114551.
[39] Muhammad, A., Asad, M., & Javed, A. R. 2020. Robust early stage botnet detection using machine learning. In *2020 International Conference on Cyber Warfare and Security (ICCWS)* (pp. 1-6). IEEE.
[40] Haq, S., & Singh, Y. 2018. Botnet detection using machine learning. In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) (pp. 240-245). IEEE.
[41] Randhawa RH, Aslam N, Alauthman M, Rafiq H, Comeau F. Security hardening of botnet detectors using generative adversarial networks. IEEE Access 2021;9:78276–92.
[42] Stiawan, D., Arifin, M. A. S., Rejito, J., Idris, M. Y., & Budiarto, R. 2021. A Dimensionality Reduction Approach for Machine Learning Based IoT Botnet Detection. In 2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) (pp. 26-30). IEEE.
[43] Hosseini S, Nezhad AE, Seilani H. Botnet detection using negative selection algorithm, convolution neural network and classification methods. Evol Syst 2022;13(1):101–15.
[44] Lefoane M, Ghafir I, Kabir S, Awan IU. Machine Learning for Botnet Detection: An Optimized Feature Selection Approach. In: In *The 5th International Conference on Future Networks & Distributed Systems*. p. 195–200.
[45] Kolpe P, Kshirsagar D. Botnet Detection Using Bayes Classifier. In: Applied Information Processing Systems. Singapore: Springer; 2022. p. 321–30.
[46] Hoang XD, Nguyen QC. Botnet detection based on machine learning techniques using DNS query data. Future Internet 2018;10(5):43.
[47] Nõmm, S., & Bahşi, H. 2018. Unsupervised anomaly based botnet detection in IoT networks. In 2018 17th IEEE international conference on machine learning and applications (ICMLA) (pp. 1048-1053). IEEE.
[48] Shang, Y., Yang, S., & Wang, W. 2018. Botnet detection with hybrid analysis on flow based and graph based features of network traffic. In International Conference on Cloud Computing and Security (pp. 612-621). Springer, Cham.
[49] Maeda, S., Kanai, A., Tanimoto, S., Hatashima, T., & Ohkubo, K. 2019. A botnet detection method on SDN using deep learning. In 2019 IEEE International Conference on Consumer Electronics (ICCE) (pp. 1-6). IEEE.
[50] Ayo FE, Folorunso SO, Abayomi-Alli AA, Adekunle AO, Awotunde JB. Network intrusion detection based on deep learning model optimized with rule-based hybrid feature selection. Informat Secur J Global Perspect 2020;29(6):267–83.
[51] Kumar P, Gupta GP, Tripathi R. PEFL: Deep Privacy-Encoding-Based Federated Learning Framework for Smart Agriculture. IEEE Micro 2021;42(1):33–40.
[52] Alieyan K, Almomani A, Anbar M, Alauthman M, Abdullah R, Gupta BB. DNS rule-based schema to botnet detection. Enterprise Informat Syst 2021;15 (4):545–64.
[53] Kwon J, Lee J, Lee H, Perrig A. PsyBoG: A scalable botnet detection method for large-scale DNS traffic. Comput Netw 2016;97:48–73.
[54] Pomorova, O., Savenko, O., Lysenko, S., Kryshchuk, A., & Bobrovnikova, K. 2016. Anti-evasion technique for the botnets detection based on the passive DNS monitoring and active DNS probing. In International Conference on Computer Networks (pp. 83-95). Springer, Cham.
[55] Wang TS, Lin HT, Cheng WT, Chen CY. DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis. Comput Secur 2017;64:1–15.
[56] Dwyer, O. P., Marnerides, A. K., Giotsas, V., & Mursch, T. 2019. Profiling IoT-based Botnet Traffic using DNS. In 2019 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.
[57] Ibrahim WNH, Anuar S, Selamat A, Krejcar O, Crespo RG, Herrera-Viedma E, et al. Multilayer framework for botnet detection using machine learning algorithms. IEEE Access 2021;9:48753–68.
[58] Masud, M. M., Al-Khateeb, T., Khan, L., Thuraisingham, B., & Hamlen, K. W. 2008. Flow-based identification of botnet traffic by mining multiple log files. In 2008 first international conference on distributed framework and applications (pp. 200-206). IEEE.
[59] Shahrestani, A., Feily, M., Ahmad, R., & Ramadass, S. 2009. Architecture for applying data mining and visualization on network flow for botnet traffic detection. In 2009 International Conference on Computer Technology and Development (Vol. 1, pp. 33-37). IEEE.
[60] Liao, W. H., & Chang, C. C. 2010. Peer to peer botnet detection using data mining scheme. In 2010 international conference on internet technology and applications (pp. 1-4). IEEE.
[61] Folorunso O, Ayo FE, Babalola YE. Ca-NIDS: A network intrusion detection system using combinatorial algorithm approach. J Informat Priv Secur 2016;12 (4):181–96.
[62] Dora V, Lakshmi VN. Optimal feature selection with CNN-feature learning for DDoS attack detection using meta-heuristic-based LSTM. Int J Intellig Robot Appl 2022:1–27.
[63] Ramachandran A, Feamster N, Dagon D. Revealing botnet membership using dnsbl counter-intelligence. Sruti 2006;6:49–54.
[64] Koza JR. Genetic programming: On the programming of computers by means of natural selection. Massachusetts: MIT; 1992.
[65] Goldberg, D. E., & Holland, J. H. 1988. Genetic algorithms and machine learning. Machine Learning, 3(2): 95–99 Springer, USA.
[66] Alcalá R, Gacto MJ, Herrera F, Alcalá-Fdez J. A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems. Int J Uncertainty, Fuzzin Knowledge-Based Syst, World Scientific: Singapore 2007;15(05):539–57.
[67] Fernández A, López V, del Jesus MJ, Herrera F. Revisiting Evolutionary Fuzzy Systems: Taxonomy, applications, new trends and challenges. Knowl-Based Syst 2015;80:109–21.
[68] Bishop CM. Neural networks for pattern recognition. England: Oxford University; 1995.
[69] Manocha S, Girolami MA. An empirical analysis of the probabilistic Knearest neighbour classifier. Pattern Recogn Lett 2007;28:1818–24.
[70] Chaudhari P, Agarwal H, Bhateja V. Data augmentation for cancer classification in oncogenomics: an improved KNN based approach. Evol Intel 2021;14 (2):489–98.
[71] Mitchell T. Machine learning. New york: McGraw Hill; 1997.
[72] Navada A, Ansari AN, Patil S, Sonkamble BA. Overview of use of decision tree algorithms in machine learning. In: In *2011 IEEE control and system graduate research colloquium*. IEEE; 2011. p. 37–42.
[73] Yan X, He J, Zhang C, Liu Z, Qiao B, Zhang H. Single-vehicle crash severity outcome prediction and determinant extraction using tree-based and other non-parametric models. Accid Anal Prev 2021;153:106034.
[74] Rathore SS, Kumar S. A decision tree logic based recommendation system to select software fault prediction techniques. Computing 2017;99(3):255–85.
[75] Muñoz V, Vallejo M, Aedo JE. Machine learning models for predicting crime hotspots in medellin city. In: In *2021 2nd Sustainable Cities Latin America Conference (SCLA)*. IEEE; 2021. p. 1–6.
[76] Palaniappan G, Sangeetha S, Rajendran B, Goyal S, Bindhumadhava BS. Malicious domain detection using machine learning on domain name features, host-based features and web-based features. Procedia Comput Sci 2020;171:654–61.
[77] Sivaguru R, Peck J, Olumofin F, Nascimento A, De Cock M. Inline detection of DGA domains using side information. IEEE Access 2020;8:141910–22.
[78] Patsakis C, Casino F. Exploiting statistical and structural features for the detection of Domain Generation Algorithms. J Informat Secur Appl 2021;58:102725.
[79] Chang C-H, Lin C-J. LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2011;2(27):27.