IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2021, accepted June 26, 2021, date of publication July 2, 2021, date of current version July 19, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3094183*

# Botnet Detection Approach Using Graph-Based Machine Learning

**AFNAN ALHARBI AND KHALID ALSUBHI**
Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
Corresponding author: Afnan Alharbi (aalharbi1729@stu.kau.edu.sa)

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant D-285-611-1442.

**ABSTRACT** Detecting botnet threats has been an ongoing research endeavor. Machine Learning (ML) techniques have been widely used for botnet detection with flow-based features. The prime challenges with flow-based features are that they have high computational overhead and do not fully capture network communication patterns. Recently, graph-based ML has witnessed a dramatic increase in attention. In communication networks, graph data offers insights information about communication patterns between hosts. In this paper, we propose a graph-based ML model for botnet detection that first considers the significance of graph features before developing a generalized model for detecting botnets based on the selected important features. We explore different feature sets selected using five filter-based feature evaluation measures derived from various theories such as consistency, correlation, and information. Two heterogeneous botnet datasets, CTU-13 and IoT-23, were used to evaluate the effectiveness of the proposed graph-based botnet detection with several supervised ML algorithms. Experiment results show that using features reduces training time and model complexity and provides high bots detection rate. Our proposed detection model detects different types of botnet families and exhibits robustness to zero-day attacks. Compared to state-of-the-art techniques flow-, and graph-based, our approach achieves higher precision and shows competitive accuracy.

**INDEX TERMS** Botnet detection, cybersecurity, feature selection, machine learning.

## I. INTRODUCTION

In recent years, there has been an increase in demand for cybersecurity and defense against various forms of cyber-attacks. Cybersecurity has recently attracted a lot of attention due to the popularity of the Internet of Things (IoT), the exponential development of computer networks, and the large number of applications used by individuals or groups for personal or industrial purposes.

Malicious software (Malware) attacks are progressively increasing. According to a recent 2021 report [1], about 350,000 new malware and potentially unwanted software are registered daily by the AV-TEST Institute. The extensive availability of malware can create botnets, enabling cyber-criminals to exploit the collective bandwidth of thousands, if not millions, of infected devices to disrupt the day-to-day activities of governments and businesses. The number of newly discovered botnets increased by 24% in the first quarter of 2021 [2].

BOTNET is a network of infected computers (also known as bots or zombies) controlled by a "botmaster." Botnets are typically used to carry out intrusion attacks such as distributed denial-of-service (DDoS) attacks, click fraud (CF), flooding, and spamming. By combining the power of several infected bots, botmasters can exponentially increase the impact of malicious activities [3], [4]. Botmaster transmits commands to bots through either centralized command-and-control (C&C) structures, e.g., Mirai (2016) [5] or decentralized peer-to-peer (P2P) structures, e.g., Hajime (2016) [6], FritzFrog (2020) [7].

Botnet detection is a critical challenge due to the global scale of botnet-assisted attacks. Despite concerted efforts reported in the literature to detect the malicious activities of botnets, the diversity of botnet structures and protocols as attackers are continuously devising new intelligent ways to disrupt networks through botnet-assisted attacks makes botnet detection a major challenge in the cybersecurity

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

domain [8]. Although numerous solutions, such as encryption and firewalls, are intended to handle Internet-based cyber-attacks, an intrusion detection system (IDS) is more effective at protecting a computer network from external threats. As a result, the primary goal of an IDS is to detect and prevent various types of malicious network communications and computer device usage [9]. IDS detects and identifies malicious cyber-attacks through monitoring and analyzing the normal daily activities in a network. An IDS capable of identifying botnets in the network and various botnet-assisted attacks is essential for improving a system's security.

As botnets have progressed and become more complex, various botnet detection strategies have been proposed. Botnet detection using Machine Learning (ML) methods has gained a lot of attention in the last decade. Feature extraction is an important step before learning or training ML models. These features serve as discriminators in learning and inference. Some existing botnet detection techniques rely on traffic features or packet information; however, when traffic patterns are confidential or encrypted, these techniques become obsolete; additionally, traffic patterns can be intentionally altered to avoid detection [10]. Moreover, one of the major drawbacks of flow-based ML techniques to detect botnets is that they do not capture the dynamic topological structure of communication networks.

In recent years, many approaches have been proposed that leverage graph-based features to represent the true behavior of hosts [10]–[18]. Botnet detection using graph-based features takes advantage of the disparity in neighborhoods between anomalous and normal nodes in communication graphs. Detection of botnets using graph-based features has shown promising performance as it offers insight into the communications patterns between hosts in the network.

Numerous studies have recognized graph-based features for botnet detection [10]–[18]. However, a comprehensive analysis of the effect of features evaluation measurements on identifying the best graph-based features is yet to be explored. In general, ML algorithms can learn efficiently when the data contains a good set of relevant features and not too many irrelevant features. Several studies in the literature have highlighted the importance of feature selection on the performance of ML algorithms in botnet detection. To date, most of these studies have focused mainly on flow-based features. The primary challenge in this domain is finding the best graph-based features that reveal hidden network structures that can expose malicious hosts' communication patterns. Therefore, we believe that investigating the significance of feature selection for graph-based botnet detection would be a unique contribution.

Accordingly, the aim of this research is to explore new graph-based features and leverage feature evaluation measures to select the best features to build an efficient botnet detection system. Here, Naive Bayes, Decision Tree, Random Forests, AdaBoost, ExtraTrees, and K-Nearest Neighbors were used as classifiers to demonstrate the quality of the feature evaluation measure techniques. The main contributions of this paper are as follows:

- We present an effective and efficient graph-based bot detection system that is able to detect different types of botnets with different types of behavioral characteristics. The proposed approach is also suitable for a large-scale dataset and robust to zero-day attacks.
- We explore new graph-based features to train various ML techniques for botnet detection. In addition, we employ a comprehensive feature engineering step where five different feature evaluation measures derived from different theories such as information, correlation, and consistency are investigated to select the most discriminative set of features to improve ML algorithm performance and scalability.
- We train and validate our graph-based botnet detection approach on two real botnet datasets.
- We compare the performance of our graph-based botnet detection approach to state-of-the-art flow-based and graph-based botnet detection systems.

The rest of this paper is organized as follows: Section II provides a brief background and presents the current state of existing botnet detection systems, highlights limitations of the state-of-the-art and motivates the problem. The system design of our proposed model is described in Section III. In Section IV, we delineate our evaluation results in detecting botnet and benchmark the significance of our comprehensive features engineering step. Finally, in Section V, we conclude with a brief summary of this paper, highlights our contribution and instigates future research directions.

## II. BACKGROUND AND RELATED WORKS

### A. BOTNET DETECTION

A significant amount of research has been conducted in the field of cybersecurity to detect and prevent botnets and botnet-driven attacks. Botnet detection systems are designed to detect botnets in networks by analyzing and monitoring network behavior and flow. Numerous botnet detection techniques have been developed, which can be broadly categorized as signature-based and anomaly-based [3].

Signature-based techniques detect intrusions based on predefined attacks patterns (signatures). Signature-based IDSs are able to efficiently detect known threats and generally, they scale well. Over the years, many researchers implemented signature-based IDSs [19]–[21]. However, signature-based approaches become obsolete against unknown new or modified attacks. Furthermore, signature-based detection requires consistent database updates, as they mainly rely on known attacks signatures.

On the other hand, anomaly-based detection systems aim to establish normal behaviors of the network, and anything abnormal is considered an attack. Several studies [22]–[25] have applied anomaly-based botnet detection based on analytical features of packet payloads and network flows. Whereas anomaly-based detection overcomes

the limitations of the signature-based approaches for detecting unknown attacks, it may lead to a high rate of false alarms by classifying previously unseen normal system behaviors as malicious [26].

Several studies have attempted to overcome the aforementioned limitations by employing ML techniques to detect attacks. Studies like [27], [28] rely on deep packet inspection (DPI) to analyze the packet contents. Such techniques are computationally expensive since they search through the payload for known application signatures. In addition, due to rising security and privacy concerns, payload is often encrypted, making it non-trivial to inspect payload data [29].

Other studies, such as [30]–[33], have shown that host-based botnet detection techniques are capable of detecting unknown attacks with a low rate of false alarms. Host-level anomaly detection can be useful for early botnet detection [3]. However, despite the advantages of host-based monitoring, these techniques are generally not scalable because all devices must be adequately equipped with appropriate monitoring tools, such as spam detection software and antivirus software, resulting in poor host resource utilization. Furthermore, network-based attacks, such as DDoS, must be detected at the network level. Thus, host-based detection systems are not capable of detecting such attacks [3].

Flow-based techniques [34]–[41] use flow features as discriminators to detect anomalies in the network. Vinayakumar *et al.* [39] presented a botnet detection framework with two levels of deep learning. Researchers were able to distinguish between attacks and regular traffic using the domain generation algorithm. Their experimental results demonstrated that their proposed system resulted in significant improvements in terms of F1-score, detection time, and false alarm rate. Khan *et al.* [40] developed a multi-layered framework for detecting P2P botnets. A four-layer detection system is presented to address the shortcomings of single-stage botnet identification, such as class imbalance. Decision tree algorithm is employed to select the most important features. Experiment results highlight the impact of the multi-layer technique in effectively detecting P2P botnet traffic.

In [42], researchers proposed a hybrid deep learning scheme for botnet detection in IoT networks. A long short-term memory autoencoder is used first to minimize the dimensionality of network traffic features, followed by a deep bidirectional long short-term memory (BLSTM) to analyze long-term inter-related network traffic behavior. The finding shows that the proposed detection system obtained a high generalization capability, despite the substantial reduction in feature size.

To overcome the computational complexity of flow-based techniques, a recent study, BotFP [41], uses flow statistics to identify bots. However, instead of analyzing all flows, they proposed defining flows by the source IP address. BotFP distinguishes host activity using frequency distributions signatures of protocol attributes. Two BotFP variants presented to detect bots using distances to labeled clusters (BotFP-Clus) or a using supervised ML algorithm (BotFP-ML) to classify new bots. Experiment results revealed that both variants identified bots with high accuracy, while the clustering technique outperformed supervised ML in terms of recall. Nonetheless, their approach assumes that bots have to be a minimum active in order to be effective. Therefore, they proposed filtering out hosts with a low number of packets based on a threshold value, which could result in the omission of stealthy bots.

Flow-based botnet detection techniques have been extensively studied, but they have several shortcomings. These techniques rely on comparing each traffic flow to all others instead of monitoring overall network activities to detect malicious traffic, which leads to the omission of important interdependent relationships among bots that are special to a botnet. Moreover, per-flow or per-packet analysis may impose a significant computational overhead. In addition, these techniques are unreliable, as they can be evaded with encrypted flow data or with slightly altering flow characteristics (e.g., flow duration, source bytes, and destination bytes, *etc.*) [13], [14], [34], [43]. Further, adversaries may leverage distributed software, such as Zookeeper, to generate similar traffic behavior with legitimate C&C servers in order to avoid detection [13].

In recent years, many researchers [10]–[18] have attempted to analyze the impact of using communication graphs to represent hosts activates. The true structure of network communications, host interactions, and host behaviors are captured by graph-based features derived from high-level flow information. These methods are generally more efficient than flow-based approaches as they can recognize bot topological relationships and identify more informative behavioral features between bots [43]. BotGM [17] captures communication patterns between hosts by creating host-to-host communication graphs from observed network flows, then using the interquartile method to detect outliers. Their results have a low number of false positives (FPs) and a moderate level of accuracy. BotGM, on the other hand, is not scalable because it requires the creation of a new graph for each pair of unique IP addresses, resulting in high overhead.

Chowdhury *et al.* [11] proposed utilizing graph-based features to detect botnets in two steps. First, Self-Organizing Map (SOM) algorithm is applied to seven graph-based features to cluster nodes. They believe that larger, more dense clusters indicate normal connections, whereas smaller, more distant data points (or clusters of data points) imply malicious behavior. As a result, by eliminating the largest cluster, the detection overhead is reduced. They used statistical means and experts opinion to classify the remaining clusters as malicious or benign. While they presented a strong case for how removing the largest cluster reduced the detection time, relying on experts opinions is impractical for large networks and potentially error-prone, especially for previously unknown attacks.

Botchase [10] applies a hybrid supervised and unsupervised learning with graph-based features to detect botnets. According to the experiments performed by the

researchers, stand-alone classifiers are insufficient in terms of training time, precision, and overall accuracy performance. In their analysis, four supervised classification algorithms, namely logistic regression, support vector machine, feed-forward neural network, and decision tree, were evaluated. Researchers further demonstrated the effectiveness of applying a two-phase hybrid learning approach using supervised and unsupervised learning to detect botnet. Botchase first employs a clustering phase using SOM, followed by a classification phase using decision tree algorithm. Botchase showed impressive performance with 99.99% accuracy.

Some studies combined flow and graph-based features for botnet detection. Wang *et al.* [13] proposed BotMark, a botnet detection based on hybrid analysis of flow-based and graph-based traffic behaviors. A traffic pre-processing filtering out phase is first applied to filter out irrelevant traffic flows based on a whitelist of the top 1000 most popular websites obtained from Alexa.com in order to minimize traffic workload and enhance efficiency. BotMark employs fifteen statistical flow-based traffic features as well as three graph-based features. Similarity and stability of communication-flows (C-flows) are used for the flow-based detection. In the graph-based detector phase, the least-square technique and Local Outlier Factor (LOF) are used to calculate anomaly scores, which measure the differences between neighborhoods of anomalous nodes. The voting results generated by the three detectors: similarity-based detectors, stability-based detectors, and graph-based detectors are used to detect the botnet. According to the researchers' experimental results, the flow-based detector outperformed the graph-based detector. However, they only employed three graph-based features compared to fifteen flow-level features. The main limitation of BotMark is that bots that use legitimate C&C servers can evade detection because they will be filtered out in the first step. Advanced evasion techniques, such as changing the number of bytes or packets per flow, can also be used to evade the flow-based detector. BotMark achieved a high detection accuracy of 99.94%, but a low F1 score of 11.52%.

Rahal *et al.* [18] proposed a distributed architecture to detect botnets and early signs of possible DDoS attack in a two-tier detection system. First, SOM is employed to cluster hosts using flow and graph-based features. Correlations between nodes in each cluster are then used to detect bots using a deterministic method. They made a similar assumption to [11], that is, most benign hosts will be grouped in larger, more dense clusters, whereas smaller clusters of data points indicate malicious behavior as malicious behaviors are less often in the real world. Therefore, in order to improve detection speed, their method starts the detection process with the smallest cluster and works its way up until all clusters have been evaluated. Such distribution can greatly reduce detection time. Despite the fact that the majority of bots were successfully clustered in the same cluster using SOM, results showed that level two detection did not identify all bots.

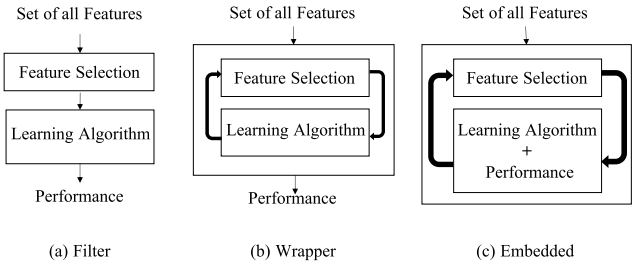


**FIGURE 1.** Feature Selection Techniques.

### B. FEATURE SELECTION FOR BOTNET DETECTION

Realizing the need for lightweight botnet detection, several studies [44]–[46] reported in literature illustrate the impact of feature selection on improving the accuracy of botnet detection while also reducing the complexity of ML models. To date, the majority of these studies are focused on flow-based features. Feature selection (also known as attribute selection or variable selection) is the process of automatically identifying and selecting the most relevant features in the feature space. This process helps to decrease the training time, reduces overfitting, and, in some cases, improves performance.

In general, feature selection techniques can be divided based on different selection strategies into three categories: **filter**, **wrapper** and **embedded** methods as shown in Fig. 1. The wrapper method selects the optimal subset of features through searching the feature space in a greedy search-based approach to evaluate all the possible combinations of features against the evaluation criterion. In the case of embedded methods, the feature selection process is built into the learning algorithm; these algorithms are embedded in the learning stage to guide the feature selection process and find the optimal feature subset. The main disadvantage of these methods is that they rely on an exhaustive search strategy to select the ideal subset among all possible combinations of feature subsets, resulting in high computational complexity. Moreover, these methods find the optimal subset features for the specific ML algorithm (i.e., model dependent) and may suffer from the risk of overfitting and decrease the generalizability of the model [47].

On the other hand, filter methods are model-independent methods that find the ideal subset of features based on predefined metrics. Therefore, the selected features have no dependence on the specific ML used, which helps in building a more accurate and generalizable model. Importantly, they are less computationally expensive than other methods and are therefore most preferable for large datasets. Alqahtani *et al.* [44] applied genetic-based extreme gradient boosting (GXGBoost) along with Fisher-score-based feature selection method to select the most relevant features for IoT botnet detection. According to the evaluation results, their approach achieved a high detection rate with only three features chosen using the Fisher-score method. In [45]

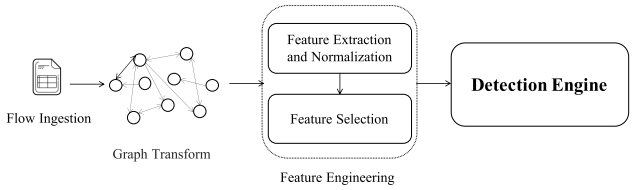


**FIGURE 2.** Architecture of Botnet Detection System.

Ismail *et al.* studied the effectiveness of information gain-based features selection on different classification algorithms for the detection of encrypted botnets. The results show that using selected features subset improved the detection rate at the expense of increased false-positive rate.

Alauthaman *et al.* [46] investigated features selection algorithms with multilayer feed-forward neural network for P2P Bot detection. Principal component analysis, Relief algorithm, and decision tree-based features selection were applied to select the top ten most relevant features extracted from TCP control packet headers to construct neural network model. Experiment results show that the highest accuracy and detection rate obtained with decision tree-based features set. Nonetheless, as the approach only tracks TCP traffic to detect botnets, botnets that do not communicate through TCP packets will not be detected.

Undoubtedly, graph ML lends itself to accuracy in botnet detection. However, the majority of current graph-based detection approaches rely on two layers of detection. This can increase the detection time and add to the approach's complexity because it requires training (and retraining) two models. Further, most of these studies have been limited to a few graph-based features. Meanwhile, several graph-based features have remained unexplored. With the rapid growth of network traffic, the need for lightweight graph-based detection has become evident. Unlike the above approaches, in this paper, we present a graph-, ML-based security model for botnet detection that first considers the importance of graph features and then builds a generalized model for detecting botnets based on the selected important features to address the aforementioned limitations.

## III. METHODOLOGY

The system architecture for our proposed botnet detection method is depicted in Fig. 2, which includes three main components: data bootstrap, feature extraction and selection, and detection engine. In the following sections, we discuss these components in detail.

### A. DATA BOOTSTRAP

#### 1) FLOW INGESTION

The flow ingestion step aims to convert bidirectional network flows into a list that is used to create the graph representation of the network flows. Each node in this study represents a unique IP address, and each edge represents a connection between two IP addresses. Network flows are transformed into a list $F$ that contains four tuples $f_i \in F$, such that $f_i = sip_i, dip_i, srcpkts_i, dstpkts_i$. Where $sip_i$ is the source host IP address, $dip_i$ is the destination host IP address, the number of data packets sent by $sip_i$ to $dip_i$ is $srcpkts_i$, and $dstpkts_i$ is the number of data packets sent by $dip_i$ to $sip_i$. As mentioned above, in our graph each node represents a unique IP address. Therefore, if multiple tuples have the same source and destination hosts, they are aggregated into one tuple in $F$, and the number of source packets and the number of destination packets are also aggregated. Furthermore, if there exists a reverse tuple $f_x, f_y \in F$ where, $sip_x = dip_y$ and $sip_y = dip_x$ then, $srcpkts_x = srcpkts_x + dstpkts_y$ and $srcpkts_y = srcpkts_y + dstpkts_y$ and $dstpkts_x = dstpkts_x + srcpkts_y$ and $dstpkts_y = dstpkts_y + srcpkts_x$.

#### 2) GRAPH TRANSFORM

Using the flow representation obtained from the flow ingestion step mentioned above, the system builds a directed graph $G(V, E)$, where $V$ is a set of vertices and $E$ is a set of weighted directed edges. Each edge $e \in E$ is associated with two vertices $u$ and $v$ from $V$, and has an associated weight $w(e)$. The set of vertices $V$ is a union of hosts from set $F$, such that,

$$V = (sip \cup dip). \tag{1}$$

And for each pair of connected vertices $v_i$ and $v_j$ in $V$ where $sip_x = v_i$ and $dip_x = v_j$, there exist directed edges $e_{i,j}$ and $e_{j,i}$ such that,

$$E = (sip_x, dip_x) \cup (dip_x, sip_x). \tag{2}$$

The weights $w(e_{i,j})$ and $w(e_{j,i})$ of edges $e_{i,j}$ and $e_{j,i}$ are $srcpkts_x$ and $dstpkts_x$, respectively.

### B. FEATURE ENGINEERING

#### 1) FEATURE EXTRACTION AND NORMALIZATION

To investigate and evaluate the effectiveness of graph-based features in detecting and distinguishing between benign and bot hosts. The graph representation of network flows obtained during the data bootstrap step is used to extract a set of graph-based features. We utilize a set of graph-based features, namely in-degree, out-degree, in-degree weight, out-degree weight, clustering coefficient and a set of centrality measures such as in-degree centrality, out-degree centrality, node betweenness, authority, hub, Katz, PageRank, closeness and eigenvector centrality. A brief introduction of these features and the rationale behind leveraging them is discussed below:

- **In-Degree (ID) and Out-Degree (OD):** Vertex degree is the measure of the total number of edges connected to a specific vertex. The in-degree of a vertex $v \in V$ in a graph $G(V, E)$ is the total number of incident edges to $v$. Where out-degree is the total number of incident edges from $v$. In C&C botnet topology, all bots communicate with the botmaster through a C&C server, which results in a relatively high ID for the C&C node. One of the primary characteristics of botnets is their fast spreading, as botnets are continuously increasing their footprint in

terms of the number of bots to gain higher privileges. Lateral Movement (LM) [48] is a technique used to spread through a network, and it is one of the first stages in any cyber-attack. As a result, having a high OD can aid in the detection of botnet LM.

- **In-Degree Weight (IDW) and Out-Degree Weight (OWD):** In a weighted graph, the in-degree weight and out-degree weight of $v$ is the total of all the weights of its incoming and outgoing edges, respectively. IDW and ODW features capture the total number of packets exchanged between hosts. As bots communicate with each other and with the botmaster, bots of the same botnet's family tend to have similar behavior in exchanging information or updating commands which can be captured using IDW and ODW.
  In graph theory and network analysis, centrality measures capture the importance of a node in the network in terms of node connections, node relationships, and node communications. Therefore, centrality measures can help distinguish between bots and hosts. The list of centrality measures used in this study to capture the behavior of hosts is explained below.
- **In-Degree Centrality (IDC) and Out-Degree Centrality (ODC):** Degree centrality is one of the most fundamental and easy to compute centrality metrics. Degree centrality assigns a score based on nodes' communication with other nodes in the network. IDC and ODC of $v$ are defined as the total number of incoming and outgoing connections divided by the total number of nodes in the network. These features add another layer of information, as it takes into consideration the entire network.
- **Betweenness Centrality (BC):** The betweenness centrality of a vertex is the number of the shortest paths through the vertex. In other words, BC is the number of times a vertex acts as a bridge along the shortest path between two other vertices. BC of vertex $v$ is defined as,

$$BC(v) = \sum_{s \neq v \neq d \in V, s \neq d} \frac{\sigma_{sd(v)}}{\sigma_{sd}}. \tag{3}$$

  where $\sigma_{sd}$ is the total number of shortest paths from $s$ to $d$ in $V$, and $(v)$ is the number of shortest paths from $s$ to $d$ that pass through a vertex $v$. BC is then normalized by dividing through the number of pairs of vertices not including $v$, which is $(n-1)(n-2)$.
  Despite the computational overhead of measuring BC with $O(|V| \cdot |E| + |V|^2 \cdot log|V|)$ time complexity and $O(V \cdot E)$ space complexity [49], BC can be a very useful feature to detect botnets. Bots in P2P botnets communicate directly with one another. If bots are not directly connected, they are more likely to communicate using the shortest paths in the network, which is captured by high BC.
- **Closeness Centrality (CC):** Closeness centrality is another intriguing and powerful centrality metric [50]. CC measures how close vertex $v$ to all other vertices in a graph, where closeness is defined in terms of the shortest path. The CC of $v$ is calculated as,

$$CC_v = \frac{1}{\sum_u d_{vu}}. \tag{4}$$

  where $d_{vu}$ is shortest path from $v$ to $u$. However, if the graph is disconnected and there is no path between two nodes, then the shortest path is set to Infinity. To avoid this issue, we use the harmonic CC, which is defined as,

$$CC_v = \sum_u \frac{1}{d_{vu}}. \tag{5}$$

  In case there is no path between the two vertices, $d_{vu}$ is set to Infinity such that $\frac{1}{d_{vu}} = 0$. A vertex's CC can reflect its influence; a vertex with a high CC will be more effective at spreading and updating commands. This feature is also valuable because it perfectly illustrates the graph's geometrical centrality and the network's compactness of organization.
- **Eigen Centrality (EC):** Eigen centrality [51], or eigenvector centrality, is a measure of the level of importance of a node in a graph, where the importance of a node depends on the importance of its neighbors. EC is an extension of degree centralities, with the addition that nodes are influenced not only by the number of their neighbors, but also by the importance of their neighbors. To compute EC, each $v_i$ is assigned an influence score $x_i$, that is iteratively exchanged with adjacent neighbors. Let, $A = (a_{v,u})$ be the adjacency matrix where,

$$a_{v,u} = \begin{cases} 1 & \text{if } v \text{ and } u \text{ are connected} \\ 0 & \text{if } v \text{ and } u \text{ are not connected.} \end{cases} \tag{6}$$

  Then, the centrality score can be defined as,

$$x_v = \frac{1}{\lambda} \sum_{u \in N(v)} a_{v,u} x_u. \tag{7}$$

  where, $N(v)$ is the set of neighbors of node $v$ and $\lambda$ is the largest eigenvalue associated with the eigenvector of the adjacency matrix [52]. Based on this definition, nodes with higher EC values are linked to nodes with higher EC values as well. As a result, EC eventually groups bots closer together and distinguishes bots from hosts.
- **Katz Centrality (KC):** Katz centrality [53] measures the relative importance of each node in a graph by considering both immediate and non-immediate neighboring nodes that are connected via immediate neighboring nodes. In contrast to EC, where all neighboring nodes have the same effect on the node's EC. KC uses attenuation factor $\beta$ where the centrality of the node is mostly affected by its first-degree neighboring nodes, and the effect on the KC diminishes as we expand to higher neighboring nodes. In other words, KC determines the importance of a node based on the importance of its neighboring nodes as well as the proximity of neighboring nodes to the node. Let $A^K$ be the adjacency matrix

where,

$$A^K_{(v,u)} = \begin{cases} 1 \text{ if there is an edge from } v \text{ to } u \text{ at time } t_k \\ 0 \text{ otherwise.} \end{cases} \tag{8}$$

Then, $KC$ of a node $v$ is computed as:

$$KC_v = \sum_k \beta^K A^K \alpha. \tag{9}$$

where $\alpha$ is a constant value that must be smaller than the inverse of the largest eigenvalue of the adjacency matrix. In a botnet, as bots communicate directly with each other and/or with the botmaster, KC can help discriminate between bots and benign hosts as interconnected neighboring nodes tend to have similar KC values.

- **PageRank Centrality (PR):** The PageRank algorithm [54] is a link analysis algorithm developed by Google to rank the relative importance of web pages on their web search engine. In PR, the importance of a node is related to the node's links from and to other nodes in a graph. PR is measured iteratively. The current rank score of each node is distributed to its adjacent nodes via its outgoing edges at each iteration. The new rank of each node at the end of each iteration is the sum of the scores received from its incoming edges. PR is given as,

$$PR_t(i) = (1-d)\sum_{k=1}^{n} W(k) + d \sum_{(i,j)\in E} \frac{PR_{t-1}(i)}{O_j}. \tag{10}$$

where $PR_t(i)$ denotes PR score of node $i$ at $t$ iteration, $(i, j)$ is the direct link from node $i$ to node $j$, $O_j$ denotes the number of outgoing links from node $j$, and $E$ represents the set of all edges in the graph. PR can be a key indicator of benign hosts because hosts with low connectivity tend to have low PR. Whereas botnets communicate with each other or with the C&C server to update commands and or transfer information, resulting in bots having relatively higher PR scores.

- **Hub and Authority:** Hyperlink-Induced Topic Search (HITS) defines two centralities measures, authority and hub. Similar to PR, HITS is a link analysis approach developed to rank pages according to their authority and hub ranking. More specifically, a node's authority is measured by the number of incoming edges; an authoritative node has a high number of incoming edges. On the other hand, the hubness of a node is computed by the number of outgoing edges. A node's hub and authority scores are the sums of the hub scores of the node's in-neighbors and out-neighbors, and the total authority score of the node's out-neighbors and in-neighbors, respectively. Let $A$ be the adjacency matrix and $A^T$ be transpose of matrix $A$, then hub $h$ and authority $a$ centrality to the vertices can be given as:

$$\begin{aligned} a &= \alpha A h \\ h &= \beta A^T a. \end{aligned} \tag{11}$$

Bots in centralized botnets receive commands and updates from a central node (i.e., botmaster), causing this node to have a high hubness. Furthermore, bots scan the network in the early stages of the intrusion kill-chain to infect more hosts. As a result, authorities and hubs scores can assist in distinguishing bot behaviors in the network.

- **Local Clustering Coefficient (LCC):** The local clustering coefficient [55] measures the neighborhood connectivity of a node. LCC is a metric to determine how close a node's neighbors are to each other. LCC is defined as,

$$LCC_i = \frac{\left|\{e_{jk}\}\right|}{k_i(k_i-1)} : v_j, v_k \in N_i, \quad e_{jk} \in E. \tag{12}$$

where is $k_i$ the out-degree of vertex $i$, and $N_i$ the set of out-neighbors of vertex such that,

$$N_i = \{v_j : e_{jk} \in E\}. \tag{13}$$

LCC has a low computational overhead with $O(|V|\langle K^2\rangle)$ time, where $\langle K^2\rangle$ is the second moment of the degree distribution. Interconnectedness can play a major role in detecting malicious bots' activities in P2P botnets, as bots connect and communicate with each other, resulting in high LCC.

### 2) FEATURE SELECTION TECHNIQUES

Feature evaluation measures are crucial in guiding the search for the best discriminative features in the feature space to improve accuracy while reducing computational costs. Filter methods, as previously described, find and select discriminative features based on predefined metrics without considering the ML model, making them extremely computationally efficient.

Filter methods use various measures based on the intrinsic characteristics of data to determine the significance of features or feature subsets; these measures can be classified as univariate or multivariate. Univariate methods assign a score to each feature and measure the relative importance of each feature individually; the score of each feature is unaffected by the scores of other features in the features space. Multivariate methods, on the other hand, assess the relative significance of the entire set of selected features. In this study, we investigated various univariate and multivariate evaluation measures, which are explained below.

- **Information Measure:** Information measure [56] is a univariate filter measure that evaluates the information significance of each feature in the context of the target class variable. It is capable of exposing the relationship of each feature with the target class. To this end, the information measure evaluates the information gain (IG) or mutual information for each feature in the context of the target class [56], [57]. IG is calculated as,

$$\begin{aligned} IG &= H(C) - H(C|F), \\ &= H(F) - H(F|C), \\ &= H(F) + H(C) - H(F, C). \end{aligned} \tag{14}$$

where $H(C)$ is the entropy of class $C$ and is calculated as,

$$H(C) = -\sum_{c \in C} p(c) log_2 p(c). \quad (15)$$

And $H(C|F)$ is the conditional entropy of class $C$ given the feature $F$ and is calculated as,

$$H(C|F) = -\sum_{f \in F} p(f) \sum_{c \in C} p(c|f) log_2 p(c|f). \quad (16)$$

IG algorithm has low computational complexity with a time of $O(N \cdot T)$, where $N$ is the number of features in the dataset and $T$ is the time required to calculate the IG. Features with high IG values are important because they have high relevance in detecting the target class $C$.

- **Gini Impurity (GI)**: Gini impurity or Gini index [58] is a well-known measure of the impurity of a node in statistics and data mining. GI measures the impurity of a node based on the probability of wrongly classifying a randomly chosen instance. For a feature $f$, GI [58] can be expressed as,

$$GI(f) = 1 - \sum_{i=1}^{c} p_i^2. \quad (17)$$

where $p_i$ represents the probability of an element being classified for a particular class. GI is a multivariate measure that evaluates the feature merit based on the impurity of the features. Features are ranked in order of decreasing impurity using the above equation; the lower the impurity, the more significant the feature.

- **Correlation Measure (CM):** Correlation measure [59] is one of the most powerful feature evaluation measures. CM can be implemented in a multivariate or bivariate manner. The multivariate CM approach is used to determine the importance of the candidate feature subset, which considers the degree of correlation between each feature in the subset and the target class, as well as the correlation between features within the subset.
  CM can be extremely useful in identifying a discriminative feature set with a strong association to the target class. Furthermore, they are capable of eliminating irrelevant and redundant features that have a weak correlation with the target class. CM applies a heuristic search technique to investigate the feature space and measure the relevance of the candidate feature subset. Correlations-based features selection (CFS) [59] algorithm starts with an empty feature set and evaluates each new feature as follows:

$$M_s = \frac{k\overline{r_{cf}}}{k + k(k-1)\overline{r_{ff}}}. \quad (18)$$

where $M_s$ is the heuristic merit evaluation of a feature subset $S$ containing $k$ features, $\overline{r_{cf}}$ is the mean correlation value between features and class labels $f \in S$, and $\overline{r_{ff}}$ is the average intercorrelation between two features in $S$. This algorithm has computational time complexity of $O(NI \cdot M^2)$ where $NI$ is the total number of instances in the dataset, and $M$ is the number of selected features.

- **Pearson's Correlation:** Pearson's correlation [59], [60] is a bivariate CM, where the correlation between any two features is computed as,

$$r = \frac{(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{(x_i - \overline{x})^2 (y_i - \overline{y})^2}}. \quad (19)$$

$r$ denotes Pearson's correlation coefficient between two feature $x$, $y$. $x_i$ and $y_i$ are the values of $x$ variable and $y$ variable in a sample, respectively. And $\overline{x}, \overline{y}$ are the mean value of the $x$ and $y$ variable in the data. Pearson's correlation can be implemented to select the most correlated features with the class label. Pearson's correlation is powerful in selecting the most predictive features in the feature space. However, unlike CFS, the correlation between the selected features is not considered. Therefore, the selected subset might have redundant features. Nonetheless, Pearson's correlation is more computationally effective in removing irrelevant features, as the time complexity of this algorithm is $O(n)$.

- **Consistency Measure:** Consistency measure is a multivariate measure that assesses the relevance of features subsets based on their consistency rate over the dataset [61]. Starting with the entire original feature set, feature selection based on consistency measure (CBF) uses heuristic search technique to search the given feature space through randomly generating subsets of features until it reaches a minimum size feature subset with a consistency rate equal to the consistency rate of the full set of features. In CBF, if two instances match except for their class label, then they are considered inconsistent. Consistency measure is advantageous in discovering and selecting features that improve the performance of a learning algorithm. The consistency rate of the features set is computed as,

$$CR_S = \frac{\sum_{i=0}^{j} |A_i| - |B_i|}{NI}. \quad (20)$$

where $S$ represents feature subset and $j$ is the number of possible combinations of feature values for $S_i$. $|A_i|$ and $|B_i|$ are the number of occurrences and the cardinality of the majority class for the *ith* feature value in the combination, and $NI$ is the total number of instances in the dataset. The complexity time of this algorithm is $O(NI \cdot M^2)$.

### C. DETECTION ENGINE

To assess the efficiency of graph-based features in identifying botnet behaviors, we investigate several powerful ML classification algorithms, including Naive Bayes, Decision Tree, Random Forests, AdaBoost, ExtraTrees Classifier, and K-Nearest Neighbors. The chosen classifiers cover various ML families since they approach the problem of supervised classification differently. After the ML model has been trained, it is deployed in the system to perform bot detection.

**TABLE 1.** CTU-13 Dataset Description [63].

| ID | Duration (hrs) | # Flows | Bot | #Bots | Activity |
|---|---|---|---|---|---|
| 1 | 6.15 | 2,824,637 | Neris | 1 | IRC, SPAM, CF |
| 2 | 4.21 | 1,808,123 | Neris | 1 | IRC, SPAM, CF |
| 3 | 66.85 | 4,710,639 | Rbot | 1 | IRC, PS |
| 4 | 4.21 | 1,121,077 | Rbot | 1 | IRC, DDoS |
| 5 | 11.63 | 129,833 | Virut | 1 | SPAM, PS |
| 6 | 2.18 | 558,920 | Menti | 1 | PS |
| 7 | 0.38 | 114,078 | Sogou | 1 | HTTP |
| 8 | 19.5 | 2,954,231 | Murlo | 1 | PS |
| 9 | 5.18 | 2,753,885 | Neris | 10 | IRC, SPAM, CF, PS |
| 10 | 4.75 | 1,309,792 | Rbot | 10 | IRC, DdoS |
| 11 | 0.26 | 107,252 | Rbot | 3 | IRC, DdoS |
| 12 | 1.21 | 325,472 | NSIS.ay | 3 | IRC, P2P |
| 13 | 16.36 | 1,925,150 | Virut | 1 | HTTP, SPAM, PS |

The detection engine is the core component of our botnet detection approach. It is responsible for analyzing the network to detect botnet activities.

## IV. EXPERIMENT

### A. ENVIRONMENT SETUP

#### 1) HARDWARE

Data analysis, pre-processing, visualization, features engineering, model training and validation are performed on Aziz supercomputer, each experiment performed on a single node with 24 cores and 96 GB memory for each node.

#### 2) SOFTWARE

The software implementation of our approach is based on Python. In our experiment, we utilize several powerful python libraries (Pandas, NumPy, scikit-learn, and Matplotlib). Graph-tool [62] is used to transform network flows into graphs and extract graph-based features. python-weka-wrapper library is used to implement different features evaluation measures.

### B. DATASET

The malicious behavior of infected devices by malware forming botnets is typically associated with cybersecurity breaches. We aim to build a ML model that can distinguish the malicious behaviors of different botnet families. For this purpose, two datasets, CTU-13 and IoT-23, with real botnet traffic are used in this study.

#### 1) CTU-13 DATASET

CTU-13 [63] is a publicly available dataset that contains thirteen scenarios with different numbers of infected computers and seven real botnet families performing different malicious activities such as DDoS, CF, port scanning (PS), spamming, *etc.* Table 1 summarizes the dataset duration, number of flows, number of bots and the type of bot in each subset. We used scenario #9 for testing and the remaining scenarios were used for training and validation. In scenario #9, there are 10 unique hosts infected with the Neris botnet and $\approx 367K$ benign hosts.

**TABLE 2.** IoT-23 Dataset Description [65].

| Dataset Name | Duration (hrs) | #Packets | #Flows | Name |
|---|---|---|---|---|
| IoT-Malware-34-1 | 24 | 233,000 | 23,146 | Mirai |
| IoT-Malware-44-1 | 2 | 1,309,000 | 238 | Mirai |
| IoT-Malware-20-1 | 24 | 50,000 | 3,210 | Torii |
| IoT-Malware-21-1 | 24 | 50,000 | 3,287 | Torii |
| IoT-Malware-42-1 | 8 | 24,000 | 4,427 | Trojan |
| IoT-Malware-17-1 | 24 | 109,000,000 | 54,659,864 | Kenjiro |
| IoT-Malware-8-1 | 24 | 23,000 | 10,404 | Hakai |
| IoT-Malware-35-1 | 24 | 46,000,000 | 10,447,796 | Mirai |
| IoT-Malware-3-1 | 36 | 496,000 | 156,104 | Muhstik |
| IoT-Malware-1-1 | 112 | 1,686,000 | 1,008,749 | Hide & Seek |
| Honeypot-7-1 | 1.4 | 8,276 | 139 | benign |
| Honeypot-4-1 | 24 | 21,000 | 461 | benign |
| Honeypot-5-1 | 5.4 | 398,000 | 1,383 | benign |

In the CTU-13 dataset, the distribution of botnet traffic is only 1.5% of the entire network traffic, which demonstrates how extremely imbalanced this dataset is. Moreover, as our approach focuses on detecting infected bots in the network rather than infected flows, the ratio of benign hosts to bots is 3M:35. To overcome the class imbalance in the dataset, we applied ADASYN [64] to over-sample the minority class to have a 6:4 benign to bot ratio.

#### 2) IoT-23 DATASET

IoT-23 [65] is a network traffic dataset that includes 20 malicious subsets and 3 benign subsets of real IoT devices. We leveraged 10 of the malicious subsets that contain infamous IoT botnet families as well as the three benign subsets described in Table 2. We used a stratified split to use 80% of the dataset for training and validation and 20% for testing. In addition, since the dataset is also imbalanced, we applied ADASYN [64] to obtain a 6:4 distribution on the training set.

### C. METRICS

Several experiments were conducted to investigate and analyze the effectiveness of different ML algorithms for botnet detection. It is important to define performance measures that are relevant to the task of botnet detection. For this purpose, the most widely adopted four metrics namely, Accuracy, Recall (or detection rate), Precision, and F1-Score, were adopted as in most previous literature on botnet detection. These metrics are defined as follows,

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{21}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{22}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{23}$$

$$F1 - Score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)} \tag{24}$$

### D. PERFORMANCE AND RESULTS

#### 1) GRAPH TRANSFORM

To extract graph-based features from graph representations of network flows, we first generate the graph by ingesting

**TABLE 3.** Features Selected by Five Feature Evaluation Measures.

| Feature Evaluation Techniques | Dataset | Feature Subset |
|---|---|---|
| IG | CTU-13 | {BC, OD, ODC, IDC, ID, PR} |
| | IoT-23 | {ODW, OD, CC, BC, ID, Hub} |
| GI | CTU-13 | {ID, OD, ODW, IDC, ODC, Hub} |
| | IoT-23 | {CC, ODW, ID, Hub} |
| CFS | CTU-13 | {OD, BC, ID, IDC, ODC, IDW} |
| | IoT-23 | {ID, OD, IDW, ODW, Hub, CC} |
| Pearson's Correlation | CTU-13 | {ID, OD, IDW, ODW, BC, PR} |
| | IoT-23 | {ID, OD, IDW, ODW, Hub, Katz} |
| CBF | CTU-13 | {ID, OD, IDW, ODW, IDC, ODC} |
| | IoT-23 | {ID, OD, IDW, ODW, IDC, ODC} |

network flows, then extract and normalize all features for each subset of the datasets.

#### 2) FEATURE SELECTION ANALYSIS

We investigate the impact of feature selection techniques on the performance of ML algorithms by examining the five feature evaluation measures listed above to select the most discriminative set of features. For univariate measures, we ranked features based on their merit score and selected the top 40% of our feature space. While for multivariate measures, we leveraged a greedy step-wise strategy as our search strategy where the minimum percentage of selected features is also set to 40%. Features selected by each feature evaluation measure on the training datasets are reported in Table 3. 10-folds stratified cross-validation strategy was applied with random seed for feature selection to avoid selection bias. The CBF measure selected the same set of features for both datasets. Whereas using the GI measure, only four features on the IoT-23 dataset have merit.

The performance evaluation procedures and the running time were repeated ten times for each evaluation criteria, and the numbers reported in the tables are the arithmetic mean of these ten results. We used stratified 10-fold cross-validation to train the ML algorithms.

### *E. EXPERIMENTAL EVALUATION*

#### 1) EXPERIMENTAL EVALUATION ON THE CTU-13 DATASET

To ensure the best results from classifiers, the best parameter values for each classifier were chosen using a stratified grid-search on the training set. The reported results of grid-search with the parameter range used and the optimal parameter values are highlighted in Table 4.

Performance analysis for all classifiers with *all* baseline features are presented in Fig 3, where NB, KNN, AdaBoost and ExtraTrees classifiers were able to accurately detect all bots with 100% recall. ExtraTrees shows the best performance, it successfully detects all bots in the test dataset, with 100% accuracy and no misclassification of any of the benign hosts. Comparatively, with superiority in precision and F1 score, KNN outperformed NB and AdaBoost.

**TABLE 4.** Parameter Settings for Classifiers.

| Classifier | Parameters | Value Range | Optimal Value |
|---|---|---|---|
| **NB** | alpha | [0.5,0.7,0.9,1.1,1.3,1.5] | 0.5 |
| | fit_prior | [True, False] | False |
| **DT** | criterion | [Gini, Entropy] | Entropy |
| | max_depth | Range (1,40) | 13 |
| **RF** | criterion | [Gini, Entropy] | Entropy |
| | n_estimators | [10, 100, 1000] | 1000 |
| | max_features | [sqrt, log2] | log2 |
| | learning_rate | [0.0001, 0.001, 0.01, 0.1, 1.0] | 1.0 |
| **Ada -Boost** | n_estimators | [50, 100, 500, 600, 800, 1000] | 600 |
| | learning_rate | [0.0001, 0.001, 0.01, 0.1, 1.0] | 1.0 |
| **KNN** | n_neighbors | [3,5,7,9,11,13] | 13 |
| | distance metric | [manhattan, euclidean, minkowski] | Manhattan |
| | Weights | [uniform, distance] | Uniform |
| **Extra-Tress** | criterion | [Gini, Entropy] | Entropy |
| | n_estimators | [50, 100, 400, 500, 600, 800, 1000] | 400 |

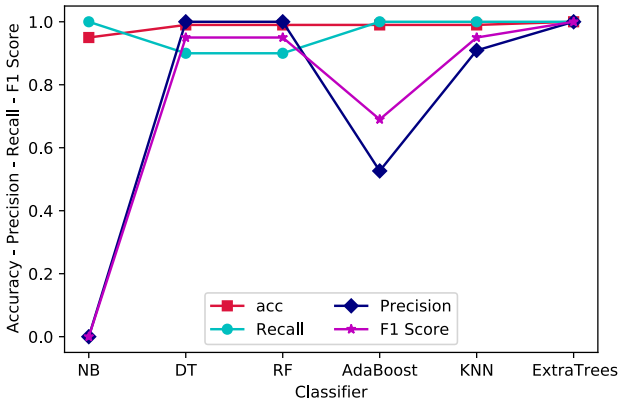


**FIGURE 3.** Classification Performance Analysis on CTU-13 with Baseline Features.

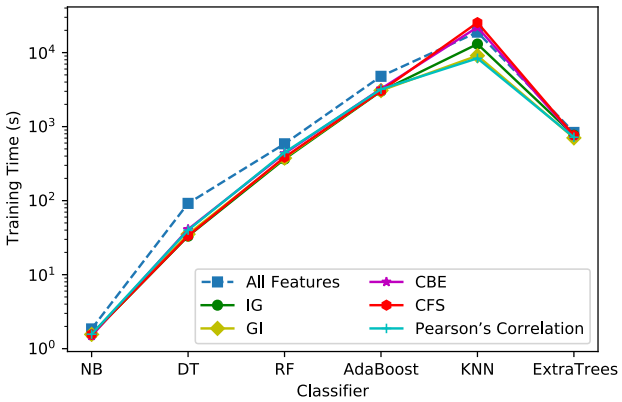


**FIGURE 4.** Classifiers Training Time on CTU-13.

DT and RF achieved similar performance outcomes, as both obtained a 90% recall with one undetected bot. From the above results, it's apparent that KNN, AdaBoost, and

ExtraTrees showed efficient performance, while NB, DT, and RF still have room for improvement. When evaluating training times of baseline classifiers, as shown in Fig. 4 NB needed the least training time of 1.85 seconds. AdaBoost and KNN, on the other hand, required the most time to train. Even though NB performed well in terms of recall and training time, the high number of FPs indicates that the model is over-fitting and not able to correctly identify benign hosts. However, feature selection can help remove irrelevant features that cause such bias.

Based on the above performance evaluations, to remedy the problem of high FPs, reduce computational overhead, and improve the accuracy of ML models. The second set of experiments was carried out to investigate the impact of the feature selection step using feature sets obtained from the previously discussed feature evaluation measure.

Performance evaluation obtained for classifiers with each feature evaluation measures over baseline performance of these classifiers with all features shown in Tables 5 - 7. As highlighted in Table 6, all classifiers, except RF, were able to successfully achieve 100% recall using different selected features subsets, whereas RF achieves 100% recall with IG and GI features. NB classifier performed impressively in identifying all bots across all feature subsets. Moreover, in terms of precision and F1 score, NB exhibits performance improvements up to 60% and 50% using Pearson's correlation-based feature set. NB performed the best with Pearson's correlation-based feature set with only 1.57s training time.

DT exhibits perfect results with Pearson's correlation-based feature set achieving 100% accuracy. Another most noteworthy result is that the DT classifier was able to boost the recall to 100% with all selected features subsets. On the other hand, the precision and F1 score of the DT classifier deteriorated with IG, CBF and CFS based feature subset, with 9, 6 and 5 benign hosts were misclassified out of the $\approx 367K$ hosts. Overall, it can be noted that the DT classifier with Pearson's correlation-based feature set and GI feature set outperformed the baseline classifier with only 43% and 38% of baseline classifier's training time, respectively.

With Pearson's correlation-based feature set, the RF classifier was able to maintain the same performance as the baseline classifier while requiring 34% less training time. Although RF with IG and GI outperforms the baseline classifier in terms of recall, precision deteriorated as the number of FPs increased.

When assessing the performance of KNN using selected features, it is noteworthy that KNN benefited greatly in terms of training time, requiring the least training time with Pearson's correlation feature set with approximately 3 hours less than the KNN baseline classifier. However, KNN exhibits substandard performance across all features subsets as the number of misclassified benign hosts increases. Generally, the performance of KNN is significantly affected and confined by the number of neighbors (K). By optimizing the number of K with Pearson's correlation features set as shown in Fig. 5, we were able to increase the F1 score to 65%, with only 11 benign hosts out of $\approx 367K$ wrongly flagged.

Similarly, based on the AdaBoost classifier experimental results, AdaBoost benefited significantly in terms of the training time, requiring only 60% of baseline total time with Pearson's correlation-based feature set. Recall that both AdaBoost and KNN classifies showed competitive results

**TABLE 5.** Accuracy Analysis of Features Evaluation Measures on CTU-13.

| Classifier | Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | All Features | IG | GI | CBE | CFS | Pearson's Correlation |
| NB | 0.95 | 0.99 | 0.81 | 0.99 | 0.99 | 0.99 |
| DT | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | **1.0** |
| RF | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| AdaBoost | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| KNN | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| ExtraTrees | **1.0** | 0.99 | **1.0** | 0.99 | 0.99 | **1.0** |

**TABLE 6.** Recall Analysis of Features Evaluation Measures on CTU-13.

| Classifier | Recall | | | | | |
|---|---|---|---|---|---|---|
| | All Features | IG | GI | CBE | CFS | Pearson's Correlation |
| NB | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| DT | 0.90 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| RF | 0.90 | 1.0 | 1.0 | 0.90 | 0.90 | 0.90 |
| AdaBoost | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| KNN | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| ExtraTrees | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |

**TABLE 7.** Precision Analysis of Features Evaluation Measures on CTU-13.

| Classifier | Precision | | | | | |
|---|---|---|---|---|---|---|
| | All Features | IG | GI | CBE | CFS | Pearson's Correlation |
| NB | 0.0 | 0.32 | 0.0 | 0.42 | 0.32 | **0.50** |
| DT | **1.0** | 0.53 | 0.91 | 0.63 | 0.67 | **1.0** |
| RF | **1.0** | 0.39 | 0.59 | 0.82 | 0.64 | 0.90 |
| AdaBoost | **0.53** | 0.09 | 0.13 | 0.08 | 0.09 | 0.08 |
| KNN | **0.90** | 0.17 | 0.19 | 0.20 | 0.09 | 0.25 |
| ExtraTrees | **1.0** | 0.77 | **1.0** | 0.77 | 0.77 | **1.0** |

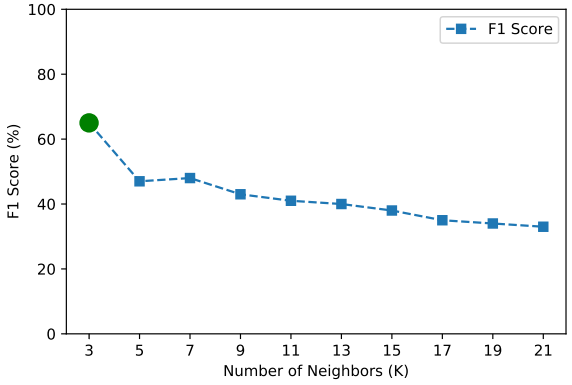


**FIGURE 5.** F1 Score VS Number of Neighbors for KNN with Pearson's Correlation Features.

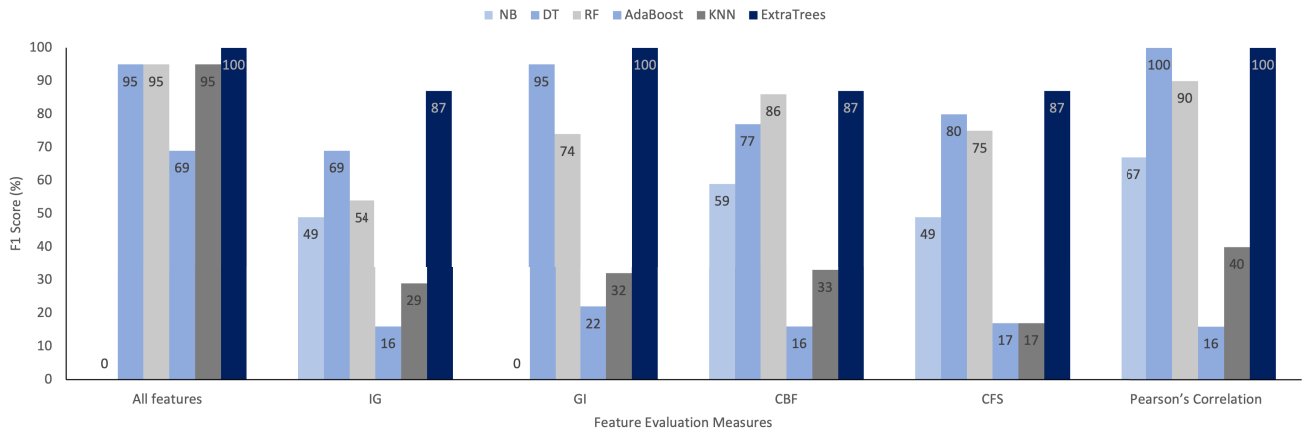


**FIGURE 6.** F1 Score Analysis of Features Evaluation Measures on CTU-13.

**TABLE 8.** Classification Performance Analysis on IoT botnet.

| Classifier | Accuracy | Recall | FP | Precision | F1 Score |
|---|---|---|---|---|---|
| NB | 0.99 | **1.0** | 1310 | 0.0 | 0.0 |
| DT | 0.99 | **1.0** | 1 | 0.67 | 0.8 |
| RF | **1.0** | **1.0** | **0** | **1.0** | **1.0** |
| AdaBoost | **1.0** | **1.0** | **0** | **1.0** | **1.0** |
| KNN | **1.0** | **1.0** | **0** | **1.0** | **1.0** |
| ExtraTrees | **1.0** | **1.0** | **0** | **1.0** | **1.0** |

**TABLE 9.** Accuracy Analysis of Features Evaluation Measures on IoT-23.

| Classifier | Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | All Features | IG | GI | CBE | CFS | Pearson's Correlation |
| NB | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| DT | 0.99 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| RF | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AdaBoost | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| KNN | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| ExtraTrees | 1.0 | 1.0 | 0.99 | 0.99 | 0.99 | 1.0 |

with all baseline features, yet both classifiers lacked their counterparts in terms of training time.

The most remarkable result is that the ExtraTrees classifier shows significant performance across all feature evaluation measures. ExtraTrees maintained a perfect score as the baseline classifier with the GI feature subset while requiring only 26% of the training time. Overall, the ExtraTrees classifier achieves higher detection accuracy across all feature evaluation measures in comparison to other classifiers.

In general, as highlighted in Fig. 6, the most significant performance of NB, DT, and RF classifiers was achieved with Pearson's correlation-based feature set. ExtraTrees demonstrated meaningful performance with all feature subsets, but it performed best with GI-based and Pearson's correlation-based feature subsets. These findings show that using the feature selection step reduces the computational overhead of model training while also improving detection accuracy.

### 2) EXPERIMENTAL EVALUATION ON THE IoT-23 DATASET

The IoT-23 dataset is used to assess the efficiency of the classifiers on IoT botnets. Table 8 presents the overall accuracy using *all* baseline features. All classifiers, except for NB, performed well on the baseline features. Classifiers performance evaluations with the selected feature subsets are highlighted in Tables 9, 10. Results show that with IG and Pearson's correlation-based feature sets, the RF, AdaBoost, KNN, and ExtraTrees classifiers achieved perfect scores with less training time in comparison to baseline classifiers as depicted in Fig 7. More significantly, the DT classifier improved the F1 score with all selected features subsets.

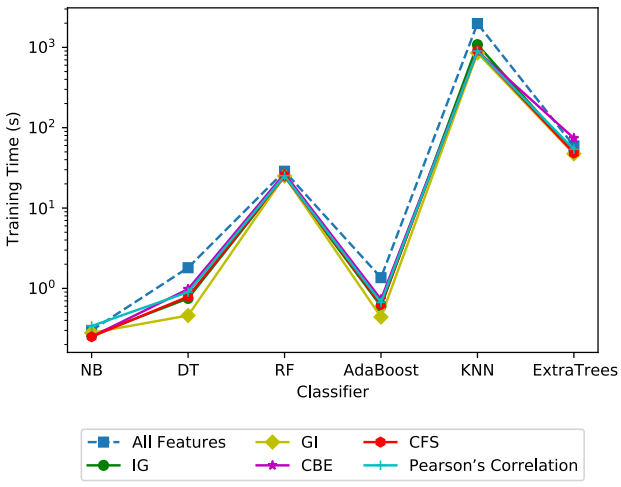


**FIGURE 7.** Classifiers Training Time on IoT-23.

Although the NB classifier was able to detect all bots using all feature subsets, when employing the GI features set, it outperformed the baseline classifier with 80% fewer FPs. However, the number of FPs remains high. We believe

**TABLE 10.** F1 Score Analysis of Features Evaluation Measures on IoT-23.

| F1 Score | | | | | | |
|---|---|---|---|---|---|---|
| **Classifier** | **All Features** | **IG** | **GI** | **CBE** | **CFS** | **Pearson's Correlation** |
| NB | 0.003 | 0.003 | **0.015** | 0.003 | 0.003 | 0.003 |
| DT | 0.8 | 1.0 | 1.0 | 1.0 | 1.0 | **1.0** |
| RF | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| AdaBoost | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| KNN | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| ExtraTrees | 1.0 | 1.0 | 0.8 | 0.67 | 0.8 | 1.0 |

**TABLE 11.** Performance Analysis Against Zero-day Attacks.

| Classifier | Accuracy | Recall | FP | Precision | F1 Score |
|---|---|---|---|---|---|
| Our model | 0.99 | **1.0** | 2 | 0.34 | 0.5 |

**TABLE 12.** Performance Evaluation of Our Model vs Botchase [10].

| Algorithm | Accuracy | Precision | Recall | FP |
|---|---|---|---|---|
| Botchase | 0.99 | 0.91 | 1 | 1 |
| **Our Model** | 1 | 1 | 1 | 0 |
| Botchase' stand-alone DT | 0.99 | 0.91 | 1 | 1 |
| **Our stand-alone DT** | 1 | 1 | 1 | 0 |

this is reflective of the fact that the NB classifier is not optimized for imbalanced data. Overall, the effectiveness of the features selection step is evident from the evaluation results as RF, KNN, and AdaBoost classifiers presented a substantial performance among all features set with improved training time. Whereas ExtraTrees performed the best with IG and Pearson's correlation-based features sets.

Based on the above performance evaluation, results clearly confirm our initial discussion that feature selection plays an important role in selecting the most discriminative features. In addition, feature selection eliminates irrelevant features, resulting in improved classification performance. Evidently, among all the experiment results, the ExtraTrees classifier demonstrated superior performance on both datasets. Therefore, we choose the stand-alone ExtraTrees classifier with Pearson's correlation-features set as ***Our Model*** for comparison to the state-of-the-art.

To further evaluate our model, we perform a robustness test against zero-day threats in addition to early attack detection. To this end, we used scenario #6 of CTU-13 for testing, which contains a unique bot that is not present in the training dataset performing anomalous Remote Desktop Protocol (RDP) session. RDP is an authentication method used in LM for an unauthorized host. As discussed before, LM is one of the earliest stages of any cyber-attack. Results presented in Table 11 asserts our model's robustness to detect threats from new unseen attack sources and early detection ability of bots activity with only two benign hosts misclassified out of $\approx 107K$ hosts.

### 3) COMPARISON TO STATE-OF-THE-ART DETECTION TECHNIQUES

We used the CTU-13 dataset to compare our model to state-of-the-art flow-based and graph-based botnet detection methods, namely Botchase [10] and BotFP-Clus [41] described in Section II.

For graph-based botnet detection, we compare our model with Botchase [10]. According to the researchers' experiments, stand-alone classifiers perform inadequately in terms of training time, precision, and overall accuracy efficiency. Therefore, they employed a two-layer detection approach based on supervised and unsupervised learning. Botchase first applies a clustering phase using SOM, followed by a classification phase using DT. We first compared our model to their two-layer detection system (i.e. Botchase). In addition, we also compared our implementation of stand-alone DT to their stand-alone DT. Results highlighted in Table 12 show that our stand-alone DT with Pearson's correlation-based features outperformed Botchase's stand-alone DT with fewer features, confirming the effectiveness of the feature selection step. Furthermore, our model outperformed Botchase in terms of precision, as we correctly identified all benign hosts.

We also evaluate our model against state-of-the-art flow-based botnet detection. We compute performance metrics for scenarios 1, 2, 8, 6, and 9 to compare our model to BotFP-Clus [41]. Table 13 reports the results for each scenario from the test set. In terms of precision, our model outperformed BotFP-Clus with only one benign host misclassified in scenarios 1 and 6, and two benign hosts misclassified in scenario 2. For scenario 8, despite being able to achieve good recall, BotFP-Clus has a very poor precision of 20%. Further, our model obtained high recall in all scenarios, with only one bot remains undetected in scenario 8. Overall, our model performance is very competitive, as we maintain a high accuracy between 99% and 100%.

## V. CONCLUSION

In this paper, we proposed a graph-, ML-based security model for botnet detection. The detection results are very promising since all algorithms were able to successfully detect all bots with 100% recall on both datasets. All classifiers achieved competitive results, with ExtraTrees providing the best detection accuracy between 99% and 100% across all feature evaluation measures. After evaluating several classification algorithms with different features subsets, we chose ExtraTrees classifier with Pearson's correlation features subset as the best model with respect to accuracy, recall and precision in botnet detection. ExtraTrees shows promising results and outperforms flow-based and graph-based

**TABLE 13. Performance Evaluation of Our Model vs BotFP-Clus [41].**

| Metrics | Recall | | | | | Precision | | | | | Accuracy | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Scenario** | **1** | **2** | **6** | **8** | **9** | **1** | **2** | **6** | **8** | **9** | **1** | **2** | **6** | **8** | **9** |
| BotFP-Clus | 1 | 1 | 1 | 1 | 1 | 0.25 | 1 | 1 | 0.2 | 0.91 | 0.98 | 1 | 1 | 0.97 | 0.99 |
| **Our Model** | 1 | 1 | 1 | 0 | 1 | 0.5 | 0.34 | 0.5 | 1 | 1 | 0.99 | 0.99 | 0.99 | 0.99 | 1 |

detection approaches with better accuracy. Overall, the comprehensive feature evaluation findings will be beneficial in the development of an efficient, lightweight botnet detection system for advanced emerging technologies such as IoT and Fog Clouds. In our work, we focused on the structural features of the network. Attributed networks include both attribute and structural data. In future work, we will extend our model to evaluate node attribute features as well.

## ACKNOWLEDGMENT

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant D-285-611-1442. The authors, therefore, gratefully acknowledge DSR technical and financial support. Computation for the work described in this paper was supported by King Abdulaziz University's High Performance Computing Center (Aziz Supercomputer) (http://hpc.kau.edu.sa).

## REFERENCES

[1] AV-TEST Institute. (Apr. 2021). *Malware Statistics & Trends Report*. Accessed: Apr. 21, 2021. [Online]. Available: https://www.av-test.org/en/statistics/malware

[2] The Spamhaus Project. (Apr. 15, 2021). *Spamhaus BotNet Threat Update: Q1-2021*. Accessed Apr. 25, 2021. [Online]. Available: https://www.spamhaus.org/news/article/809/spamhaus-botnet-threat-update-q1-2021

[3] A. Nazir and R. A. Khan, "Network intrusion detection: Taxonomy and machine learning applications," in *Machine Intelligence and Big Data Analytics for Cybersecurity Applications* (Studies in Computational Intelligence), vol. 919. Springer, 2021, pp. 3–28.

[4] S. Dange and M. Chatterjee, "IoT BotNet: The largest threat to the IoT network," in *Data Communication and Networks*. Singapore: Springer, 2020, pp. 137–157.

[5] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," in *Proc. USENIX Secur. Symp.*, Vancouver, BC, Canada, 2017, pp. 1093–1110.

[6] S. Edwards and I. Profetis, "Hajime: Analysis of a decentralized internet worm for IoT devices," *Rapidity Netw.*, vol. 16, pp. 1–18, Oct. 2016.

[7] C. Osborne. (Aug. 19, 2020). *New Fritzfrog P2P Botnet Has Breached at Least 500 Enterprise, Government Servers*. Accessed Apr. 25, 2021. [Online]. Available: https://www.zdnet.com/article/new-fritzfrog-p2p-botnet-has-breached-at-least-500-enterprise-government-servers/

[8] IBM. (Mar. 2021). *IBM X-Force Threat Intelligence Index 2021 Report*. [Online]. Available: https://doi.org/10.1016/S1353-4858(21)00026-X

[9] I. H. Sarker, Y. B. Abushark, F. Alsolami, and A. I. Khan, "IntruDTree: A machine learning based cyber security intrusion detection model," *Symmetry*, vol. 12, no. 5, pp. 754–769, 2020.

[10] A. A. Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "BotChase: Graph-based bot detection using machine learning," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 1, pp. 15–29, Mar. 2020.

[11] S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman, and L. Bian, "Botnet detection using graph-based feature clustering," *J. Big Data*, vol. 4, no. 1, pp. 14–36, Dec. 2017.

[12] P. Kalmbach, A. Blenk, W. Kellerer, and S. Schmid, "Themis: A data-driven approach to bot detection," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Honolulu, HI, USA, 2018, pp. 1–2, doi: 10.1109/INFCOMW.2018.8406870.

[13] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, "BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors," *Inf. Sci.*, vol. 511, pp. 284–296, Feb. 2020.

[14] B. Venkatesh, S. H. Choudhury, S. Nagaraja, and N. Balakrishnan, "BotSpot: Fast graph based identification of structured P2P bots," *J. Comput. Virol. Hacking Techn.*, vol. 11, no. 4, pp. 247–261, Nov. 2015.

[15] J. François, S. Wang, T. Engel, and R. State, "Bottrack: Tracking botnets using netflow and pagerank," in *Proc. Netw.*, Valencia, Spain, 2011, pp. 1–14.

[16] M. Iliofotou, H.-C. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese, "Graption: A graph-based P2P traffic classification framework for the internet backbone," *Comput. Netw.*, vol. 55, no. 8, pp. 1909–1920, Jun. 2011.

[17] S. Lagraa, J. François, A. Lahmadi, M. Miner, C. Hammerschmidt, and R. State, "Botgm: Unsupervised graph mining to detect botnets in traffic flows," in *Proc. 1st Cyber Secur. Netw. Conf. (CSNet)*, Rio de Janeiro, Brazil, 2017, pp. 1–8, doi: 10.1109/CSNET.2017.8241990.

[18] B. M. Rahal, A. Santos, and M. Nogueira, "A distributed architecture for DDoS prediction and bot detection," *IEEE Access*, vol. 8, pp. 159756–159772, Aug. 2020.

[19] G. Kaur, A. Bansal, and A. Agarwal, "Wavelets based anomaly-based detection system or J48 and Naïve Bayes based signature-based detection system: A comparison," in *Ambient Communications and Computer Systems*. Singapore: Springer, 2018, pp. 213–224.

[20] F. Erlacher and F. Dressler, "On high-speed flow-based intrusion detection using snort-compatible signatures," *IEEE Trans. Dependable Secure Comput.*, early access, Feb. 14, 2020, doi: 10.1109/TDSC.2020.2973992.

[21] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," *Appl. Soft Comput.*, vol. 92, pp. 106301–106320, Jul. 2020.

[22] S. Raheja, G. Munjal, J. Jangra, and R. Garg, "Rule-based approach for botnet behavior analysis," in *Intelligent Data Analytics for Terror Threat Prediction: Architectures, Methodologies, Techniques and Applications*. Wiley, 2021, pp. 161–179, doi: 10.1002/9781119711629.ch8.

[23] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting P2P botnets through network behavior analysis and machine learning," in *Proc. PST*, Montreal, QC, Canada, Jul. 2011, pp. 174–180.

[24] 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, 2013.

[25] W. Chen, X. Luo, and A. N. Zincir-Heywood, "Exploring a service-based normal behaviour profiling system for botnet detection," in *Proc. IM*, Lisbon, Portugal, May 2017, pp. 947–952.

[26] G. Vormayr, T. Zseby, and J. Fabini, "Botnet communication patterns," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2768–2796, Sep. 2017.

[27] W. Song, M. Beshley, K. Przystupa, H. Beshley, O. Kochan, A. Pryslupskyi, D. Pieniak, and J. Su, "A software deep packet inspection system for network traffic analysis and anomaly detection," *Sensors*, vol. 20, no. 6, p. 1637, Mar. 2020.

[28] G. A. P. Rodrigues, G. Arquelau, R. de Oliveira Albuquerque, F. E. G. de Deus, G. A. De Oliveira Júnior, L. J. G. Villalba, and T.-H. Kim, "Cybersecurity and network forensics: Analysis of malicious traffic towards a honeynet with deep packet inspection," *Appl. Sci.*, vol. 7, no. 10, pp. 1082–1111, 2017.

[29] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 1, pp. 1–99, Dec. 2018.

[30] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 807–819, Apr. 2014.
[31] G. Rosenthal, O. E. Kdosha, K. Cohen, A. Freund, A. Bartik, and A. Ron, "ARBA: Anomaly and reputation based approach for detecting infected IoT devices," *IEEE Access*, vol. 8, pp. 145751–145767, Aug. 2020.
[32] P. Deshpande, S. C. Sharma, S. K. Peddoju, and S. Junaid, "HIDS: A host based intrusion detection system for cloud computing environment," *Int. J. Syst. Assurance Eng. Manage.*, vol. 9, no. 3, pp. 567–576, Jun. 2018.
[33] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined CNN/RNN model," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Dublin, Ireland: Springer, Feb. 2018, pp. 149–158.
[34] G. Apruzzese and M. Colajanni, "Evading botnet detectors based on flows and random forest with adversarial samples," in *Proc. NCA*, Cambridge, MA, USA, 2018, pp. 1–8.
[35] S. Zavrak and M. Iskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108346–108358, Jun. 2020.
[36] S. H. Mousavi, M. Khansari, and R. Rahmani, "A fully scalable big data framework for botnet detection based on network traffic analysis," *Inf. Sci.*, vol. 512, pp. 629–640, Feb. 2020.
[37] A. A. Ahmed, W. A. Jabbar, A. S. Sadiq, and H. Patel, "Deep learning-based classification model for botnet attack detection," *J. Ambient Intell. Humanized Comput.*, vol. 11, pp. 1–10, Mar. 2020, doi: 10.1007/s12652-020-01848-9.
[38] L. Mathur, M. Raheja, and P. Ahlawat, "Botnet detection Via Mining of network traffic flow," *Procedia Comput. Sci.*, vol. 132, pp. 1668–1677, Dec. 2018.
[39] R. Vinayakumar, M. Alazab, S. Srinivasan, Q.-V. Pham, S. K. Padannayil, and K. Simran, "A visualized botnet detection system based deep learning for the Internet of Things networks of smart cities," *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 4436–4456, Jul. 2020.
[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, pp. 2375–2397, 2019.
[41] A. Blaise, M. Bouet, V. Conan, and S. Secci, "Botnet fingerprinting: A frequency distributions scheme for lightweight bot detection," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 3, pp. 1701–1714, Sep. 2020.
[42] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, and H. Gacanin, "Hybrid deep learning for botnet attack detection in the Internet-of-Things networks," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4944–4956, Mar. 2021.
[43] J. Zhao, X. Liu, Q. Yan, B. Li, M. Shao, and H. Peng, "Multi-attributed heterogeneous graph convolutional network for bot detection," *Inf. Sci.*, vol. 537, pp. 380–393, Oct. 2020.
[44] M. Alqahtani, H. Mathkour, and M. M. Ben Ismail, "IoT botnet attack detection based on optimized extreme gradient boosting and feature selection," *Sensors*, vol. 20, no. 21, pp. 6336–6357, 2020.
[45] Z. Ismail, A. Jantan, M. N. Yusoff, and M. U. Kiru, "The effects of feature selection on the classification of encrypted botnet," *J. Comput. Virol. Hacking Techn.*, vol. 17, no. 1, pp. 61–74, Mar. 2021.
[46] M. Alauthaman, N. Aslam, L. Zhang, R. Alasem, and M. A. Hossain, "A P2P botnet detection scheme based on decision tree and adaptive multilayer neural networks," *Neural Comput. Appl.*, vol. 29, no. 11, pp. 991–1004, Jun. 2018.
[47] R. Zhang, F. Nie, X. Li, and X. Wei, "Feature selection with multi-view data: A survey," *Inf. Fusion*, vol. 50, pp. 158–167, Oct. 2019.
[48] T. Bai, H. Bian, M. A. Salahuddin, A. A. Daya, N. Limam, and R. Boutaba, "RDP-based lateral movement detection using machine learning," *Comput. Commun.*, vol. 165, pp. 9–19, Jan. 2021.
[49] U. Brandes, "A faster algorithm for betweenness centrality," *J. Math. Sociol.*, vol. 25, no. 2, pp. 163–177, 2001.
[50] T. Opsahl, F. Agneessens, and J. Skvoretz, "Node centrality in weighted networks: Generalizing degree and shortest paths," *Social Netw.*, vol. 32, no. 3, pp. 245–251, Jul. 2010.
[51] A. N. Langville and C. D. Meyer, "A survey of eigenvector methods for Web information retrieval," *SIAM Rev.*, vol. 47, no. 1, pp. 135–161, Jan. 2005.
[52] M. E. Newman, "The mathematics of networks," *New Palgrave Encyclopedia Econ.*, vol. 2, pp. 1–12, Sep. 2008.
[53] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, Mar. 1953.
[54] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford InfoLab, Tech. Rep., 1999.
[55] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
[56] F. Zhao, J. Zhao, X. Niu, S. Luo, and Y. Xin, "A filter feature selection algorithm based on mutual information for intrusion detection," *Appl. Sci.*, vol. 8, no. 9, pp. 1535–15372, 2018.
[57] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.
[58] J. Han, M. Kamber, and J. Pei, "Data mining concepts and techniques third edition," *Morgan Kaufmann Ser. Data Manage. Syst.*, vol. 5, no. 4, pp. 83–124, 2011.
[59] M. A. Hall, "Correlation-based feature selection of discrete and numeric class machine learning," Ph.D. dissertation, Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, 2000.
[60] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing* (Springer Topics in Signal Processing), vol. 2. Springer, 2009, pp. 1–4.
[61] H. Liu and R. Setiono, "A probabilistic approach to feature selection-a filter solution," in *Proc. ICML*, Bari, Italy, 1996, pp. 319–327.
[62] T. de Paula Peixoto. *Graph-Tool*. Accessed Feb. 10, 2020. [Online]. Available: graph-tool.skewed.de
[63] 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.
[64] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IJCNN*, Hong Kong, 2008, pp. 1322–1328.
[65] A. Parmisano, S. Garcia, and M. Jose Erquiaga. (Jan. 22, 2020). *Aposemat IoT-23, Stratosphere Laboratory. A Labeled Dataset With Malicious and Benign IoT Network Traffic*. Accessed: Dec. 10, 2020. [Online]. Available: https://www.stratosphereips.org/datasets-iot23

**AFNAN ALHARBI** received the B.S. degree in computer science from Umm Al-Qura University, Makkah, Saudi Arabia, in 2016. She is currently pursuing the M.Sc. degree in computer science with King Abdulaziz University, Jeddah, Saudi Arabia. She is currently a Research Assistant with Umm Al-Qura University. Her research interests include machine learning, cybersecurity, and the Internet of Things.

**KHALID ALSUBHI** received the M.S. and Ph.D. degrees in computer science from the University of Waterloo, Waterloo, Canada, in 2009 and 2016, respectively. His research interests include intrusion detection systems, privacy of healthcare systems, big data, and resource management in distributed systems.