Computers, Materials & Continua
DOI: 10.32604/cmc.2023.031641
Article

Tech Science Press

BotSward: Centrality Measures for Graph-Based Bot Detection Using Machine Learning

Khlood Shinan[1,2], Khalid Alsubhi[2] and M. Usman Ashraf[3,*]

[1]Department of Computer Science, College Computer Science in Al-Leith, Umm Al-Qura University, Mecca 21421, Saudi Arabia
[2]Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[3]Department of Computer Science, GC Women University Sialkot, Pakistan
*Corresponding Author: M. Usman Ashraf. Email: usman.ashraf@gcwus.edu.pk
Received: 23 April 2022; Accepted: 08 June 2022

Abstract: The number of botnet malware attacks on Internet devices has grown at an equivalent rate to the number of Internet devices that are connected to the Internet. Bot detection using machine learning (ML) with flow-based features has been extensively studied in the literature. Existing flow-based detection methods involve significant computational overhead that does not completely capture network communication patterns that might reveal other features of malicious hosts. Recently, Graph-Based Bot Detection methods using ML have gained attention to overcome these limitations, as graphs provide a real representation of network communications. The purpose of this study is to build a botnet malware detection system utilizing centrality measures for graph-based botnet detection and ML. We propose BotSward, a graph-based bot detection system that is based on ML. We apply the efficient centrality measures, which are Closeness Centrality (CC), Degree Centrality (CC), and PageRank (PR), and compare them with others used in the state-of-the-art. The efficiency of the proposed method is verified on the available Czech Technical University 13 dataset (CTU-13). The CTU-13 dataset contains 13 real botnet traffic scenarios that are connected to a command-and-control (C&C) channel and that cause malicious actions such as phishing, distributed denial-of-service (DDoS) attacks, spam attacks, etc. BotSward is robust to zero-day attacks, suitable for large-scale datasets, and is intended to produce better accuracy than state-of-the-art techniques. The proposed BotSward solution achieved 99% accuracy in botnet attack detection with a false positive rate as low as 0.0001%.

Keywords: Network security; botnet detection; graph-based features; machine learning; measure centrality

1 Introduction

The explosion of network applications and connected devices on the Internet has resulted in a high level of network complexity in terms of management and increased concerns over data security. Many institutions are continually exposed to a variety of security threats, which can result in financial and reputational harm. Malware such as viruses, spyware, trojans, worms, key-loggers, and botnets damages various systems and online businesses. A botnet is an overlay network formed by a large number of hosts (bots or zombies) that have been infected with bots and are controlled remotely by an attacker (botmaster). Botnets are one of the most dangerous types of serious cybersecurity threats, as they are a primary vector for large-scale attack campaigns including email spam, click fraud, financial theft, and DDoS attacks [1].

Botnet attacks will get worse because they are not typically created to infect just a single computer; they are designed to infect millions of them. Botherders frequently use a Trojan horse virus to install botnets on personal computers (PCs). Users often infect their own computers by downloading email attachments, clicking on malicious pop-up ads, or installing malicious software from a website. After infecting devices, botnets are then free to attack other computers, access and modify personal information, and commit other crimes. Botnets that are more advanced can even self-propagate, discovering and infecting devices on their own. Botnets take a long time to develop. Many will remain inactive on devices until the botmaster calls them for a DDoS attack or spam dissemination [2].

The first botnet attack to receive widespread attention, known as "EarthLink Spammer", was an email spammer created by Khan K. Smith in 2000. He made at least 3 million dollars from the botnet, which sent 1.25 million emails with phishing schemes [3]. Botnets have grown over the previous 20 years to become increasingly complex and destructive [4]. In 2016, Methbot, one of the first large and sophisticated ad fraud and takedown operations, appeared. It was the world's largest botnet for defrauding the advertising industry, using sophisticated bots that pretended to be premium publishers to watch 300 million video advertisements each day on spoofed websites. Over 6,000 premium domains have been spoofed [5]. The year 2019 also produced one of the major ransomware attacks, when the French cyber police freed over 850,000 Windows computers from a botnet named "Read up." When antivirus company Avast discovered a flaw in Retadup's C&C communications protocol, it alerted the French National Gendarmerie, who in turn seized the servers [6].

Therefore, detecting botnet propagation activities early and determining the expected size of the attack is critical. However, bots can avoid detection by mimicking normal traffic, modifying packet structures, and hiding their payload characteristics through encryption. Attackers began replacing internet relay chat (IRC) with hypertext transfer protocol (HTTP) in order to blend into normal HTTP traffic and avoid detection. With port-based filtering, botnets can easily get around intrusion detection systems (IDSs) and firewalls [7].

Botnet detection has received a lot of attention in wide-ranging studies, and various techniques have been proposed for this. Signature-based techniques [8,9] generally rely on identifying pre-computed hashes of existing malware binaries and a database of known threats that must be frequently updated. However, these approaches are unable to detect unknown botnets and variations because of polymorphous attacks, zero-day attacks, and related techniques. Anomaly-based detection algorithms [10,11] are based on the idea that botnets have communication patterns that are distinct from benign hosts in networks. These are commonly used for botnet detection since they can identify unknown botnets based on the number of anomalies in network traffic, large traffic volumes, traffic on unusual ports, high network latency, and other unusual system behavior. The main drawbacks of anomaly-based detection approaches are the high rate of false alarms and the restrictions of their training data.

Other detection approaches are based on honeypot technology [12], which is a network set up with intentional vulnerabilities used as a trap without exposing the real network, but which only detects existing bots and performs poorly in real-time. Community-based anomaly detection algorithms [13] can be defined as techniques that detect a group of highly correlated and highly interactive nodes that interact unusually frequently. These approaches cannot accurately determine botnets when entire communication graphs are unavailable. Detection methods that are based on certain structures and protocols [14] can't find botnets that have different structures or protocols.

Botnet detection techniques based on ML have been considered the most effective. Feature extraction is an important step in ML techniques that involves about 80% of the time before processing data, which helps to minimize data dimensionality and improve the accuracy of ML models [15]. Flow-based features are the features most often focused on in bot detection research. However, this approach suffers from limitations such as not completely capturing the communication patterns that can expose additional aspects of malicious hosts, involves significant computational cost, and can potentially be evaded by tweaking behavioral characteristics, such as changing packet structures [16]. In order to overcome these limitations in flow-based features, we used graph-based features derived from flow-level information to reflect the true behavior of hosts. Furthermore, these properties enable the models to combine inputs from several networks and enhance spatial stability [1]. We think that adding graph-based features to ML makes it more resistant to attacks from unknown sources and communication patterns that are hard to understand [16].

The contributions of this work are as follows:

1) By proposing BotSward, an effective graph-based bot detection system that converts NetFlow traffic features to graph features, we demonstrate that our system is resistant to zero-day attacks and suitable for large datasets.
2) Using Centrality Measures to extract valuable graph-based features that achieve high precision in bot detection while consuming less time.
3) Using the new graph-based features noted above, we designed an ML-based detection system that can detect botnets with an accuracy of 99%.
4) We train and validate our graph-based botnet detection approach on real CTU-13 botnet datasets.
5) We test and compare the efficacy of our proposed graph-based features with flow-based features from other studies using the same datasets and ML algorithms.
6) We test and compare the efficacy of our proposed ML algorithms with different ML algorithms from different studies using the same datasets and graph-based features.

Organization. The remainder of this work is organized in the following manner. In Section 2, we review the background and related work, while the methodology and system design of this paper, including the graph algorithmic definitions and ML analysis techniques, are outlined in Section 3. In Section 4, we introduce the evaluation metrics, including the datasets, tools and instruments, and performance metrics. In Section 5, we present the results of our proposed botnet detection for CTU-13 datasets, followed by a summary discussion. In Section 6, we discuss the comparison evaluation approach and state-of-the-art studies. In Section 7, we give our conclusions and final thoughts. Before moving further, Tab. 1 lists down the acronyms used in this study for a better understanding.

Table 1: List of abbreviations used in this study

Acronyms	Used for	Acronyms	Used for
ML	Machine learning	GPUs	Graphics processing units
CTU-13	Czech technical university13 dataset	DNN	Deep neural network
BC	Betweenness centrality	SVM	Support vector machine
CC	Closeness centrality	DT	Decision tree
PR	PageRank	KNN	K-nearest neighbor
ID	In degree	RF	Random forest
OD	Out degree	SGD	Stochastic gradient descent
AC	Alpha centrality	ACC	Accuracy
EC	Eigenvector centrality	F	F-measure
LCC	Local clustering coefficient centrality	P	Precision
C&C	Command-and-control	R	Recall
DDoS	Distributed denial-of-service	SMOTE	Synthetic minority oversampling technique
IDS	Intrusion detection systems	T-links	Tomek links
TPUs	Tensor processing units	HTTP	Hypertext transfer protocol

2 Background and Related Work

In general, botnet detection techniques rely on deep packet inspection (DPI) and flow-based and graph-based botnet detection methods.

2.1 Deep Packet Inspection

DPI is an advanced technique for examining the full content of data packets as they pass through a monitored network checkpoint. It is highly effective in preventing overflow attacks, denial of service (DoS) attacks, buffer overflow attacks, and even some forms of malware. However, the existing botnet detection systems, which depend on DPI, suffer from high computational costs and can be exploited by crooks to facilitate similar attacks, as well as being inefficient at recognizing unknown payload signatures. Moreover, most new malware uses evasion techniques to avoid detection, such as protocol encapsulation, obfuscation, and payload encryption. As well, inspecting every packet on a high-speed network is a costly task because of the continuously increasing network speed and the daily increases in the amount of data transferred on the network [17,18].

Gadelrab et al. [19] presented BotCap, a botnet detection model based on DPI and ML techniques. The authors inspected network traffic packets in-depth in order to extract statistical features and then trained J48 decision tree and Support Vector Machine (SVM) algorithms to distinguish between benign traffic and malicious botnet traffic. The study's results showed an accuracy level of 80% for HTTP botnets and 95% for IRC botnets.

2.2 Flow-Based Detection

Flow-based detection is a network protocol that collects IP network traffic as it enters or exits an interface. Flow-based features are described as a collection of features based on information that exists in the headers of packets with similar parameters, such as source and destination IP, protocol type, source, and destination port, etc. [1]. Flow-based features are used to identify anomalies like botnets in large-volume data and high-speed networks. Extensive studies have made major contributions in the area of botnet detection using flow-based features; however, there are serious drawbacks to these approaches. Firstly, the approaches only capture the characteristics of the effects of bots on individual links, rather than using the topological structure of the communication graph as a whole, which can expose additional aspects of malicious hosts. Secondly, flow-based detection methods suffer from high computational overhead, since instead of holistically monitoring a given network's behaviors to identify malicious traffic, it requires the comparison of each particular flow of traffic to all the other flows. Finally, flow-based detection techniques can be evaded by attackers through data volume changes, using encrypted commands, or tweaking behavioral characteristics by changing the packet structure [20].

In 2014, Beigi et al. [21] discussed the effectiveness of flow-based features in combination with a C4.5 decision tree-supervised learning algorithm for botnet detection. The authors achieved a moderate detection rate, at 75%, and a low false-positive rate of only 2.3%. Gahelot et al. [22] proposed a J48 decision tree and naïve Bayes classifiers to classify flow-based P2P botnet traffic using a combined network flow PeerRush dataset and a botnet (2014) dataset. When using a J48 decision tree, their botnets detection accuracy was 99.94%, with a very low rate of false positives, and when using naïve Bayes classifiers, the accuracy was 99.46%. The main disadvantages of their approach, however, were the complexity of the model and the long processing runtime.

To overcome these limitations, using graph-based features to detect botnets has been the focus of another research path. Using graph-based features is more computationally efficient compared to using flow-based methods because it avoids the requirement to compare each particular traffic flow to all the other flows in the dataset [16].

2.3 Graph-Based Detection

Graph-based features are derived from flow-based features and reflect the real structure of host behavior, interactions, and communications, offering an option that overcomes flow-based limitations. This approach is robust and suitable for all botnet datasets because it attempts to create a graph from the captured traffic, which discards all features except IP address features to create nodes and packet features to represent directed edges. By using centrality-based graph measurements, data from packets is ignored in favor of focusing on the topological communications structure between hosts. Then, from each node, we can calculate centrality-based graph measures and extract features like DC, CC, and PR ... etc.

The authors of [16] proposed BotChase, a hybrid two-phase ML approach that leverages both unsupervised and supervised ML and graph-based bot detection systems. They modeled network communications as graphs, where hosts are vertices and communications between hosts are edges. However, the model incurs a high computational overhead when computing the features of a large communication graph, such as the betweenness centrality (BC) measures used by shortest-path algorithms computed for centrality measures.

By combining flow-based features with graph-based network traffic in [23], the study detects bots based on the hybrid analysis of these two types of features. This technique achieved a good performance, with a 96.62% F-score.

BotGM [24] suggested an unsupervised graph mining tool for detecting bots through abnormal communication patterns. For each pair of source and destination IP addresses, the authors first create a graph sequence of ports, then compare each graph using the Graph-Edit Distance (GED). They achieve a very good level of accuracy, ranging from 78% to 95%. However, the GED is computed only once for each pair of graphs, and its computation is known to be NP-complete. BotHunter [25] is designed to detect the infection and coordinate communication that takes place after a successful malware infection. CAMNEP [26] combines various state-of-the-art anomaly detection methods to improve accuracy. However, some communication patterns between hosts that are unique to a botnet are missed by these approaches. Furthermore, since the complexity of computing per-flow features is significantly high, BClus clusters the traffic sent by each IP address based on their behavioral similarity.

2.4 Research Gap

Botnets have evolved into one of the most significant cyber security concerns for networks because they impact numerous areas such as law enforcement, finance, cyber security, health care, and more. The majority of current botnet detection research relies on flow-based traffic analysis and mining communication patterns. However, these methods may not be capable of detecting bot activities in an efficient and effective manner. Existing flow-based methods have significant computational overhead and do not capture all network communication patterns, which might reveal additional aspects of malicious hosts. Moreover, flow-based features are often characteristic of specific protocol-based botnets and are not generalizable to newer botnet types. A graph-based approach is rather intuitive for overcoming flow-based approaches' limitations, as graphs are true representations of network communications. The drawback of graph-based detection systems compared to flow-based detection systems is the time required to extract features such as BC computation, which can sometimes be considerable. Although a big improvement was made over the initial BC computation algorithm, many researchers argued that the BC algorithm is still too costly for large graphs.

To solve this problem, some graph-based feature algorithms such as BC and CC are easily parallelizable, as using more cores will result in speed improvements. Also, we obtain superior results in terms of time reduction when we exclude BC and replace it with CC. The development of a quick and non-rule-based technique for detecting botnets is a significant step toward building a new graph-based detection methodology. Simultaneously, the approach must be validated on a real-world dataset with various botnet types. This detection scheme must also be sufficiently robust to detect any type of botnet present in the dataset. We show that incorporating graph-based features into ML yields robustness against complex communication patterns and unknown attacks. Furthermore, cross-network ML model training and inference are possible.

3 System Design

We illustrate our proposed system, BotSward, as shown in Fig. 1. Botsward is a classification model for botnet detection, where the current botnet anomaly detection approach based on NetFlow features can potentially be evaded by attackers through data volume changes, using encrypted commands, or tweaking behavioral characteristics by changing the packet structure. This problem is known as misleading. It works in four steps: traffic generation, data preprocessing, model building, and

bot identification. In the first step, we collect a very large amount of network traffic flow from the real-world public CTU-13 dataset discussed in Section 4. In the second step, we balance the distribution of the dataset classes and Select 3 features from 42 features, and then clean the data to remove duplicate or irrelevant data. In the third step, the system creates a graph G(V; E) from the features extracted, where V is a set of nodes and E is a set of directed edges. After that, we extract seven graph-based features from network traffic to characterize the behavior of the botnets and save them in a file to use in the model training phase. In the model training, we compared the results of seven different ML models and a Deep Neural Network (DNN), and in the last step, we identified bot traffic and normal traffic and then calculated the performance of the different models.

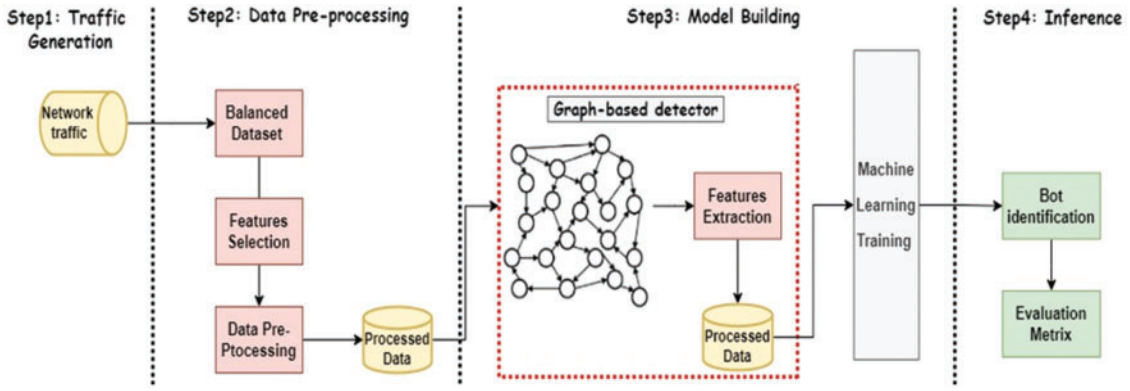


Figure 1: BotSward architecture

In this work, the software implementation is primarily based on Python. We used Jupyter Notebook, which is an open-source application that facilitates data visualization, data preprocessing, ML, statistical modeling, and much more, as well as Google Colab Notebook, which is hosted on Google cloud servers and provides access to the Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs) for tasks that can be done in a Jupyter notebook. Python 3.6 was used to implement the base classifiers and was used as the scripting language for writing most of the code. For this study, we used different important libraries such as Networkx, Scikit-learn, Keras, Tensorflow, Pandas, NumPy, Matplotlib, and Seaborn [27]. The main library in our work to compute the centrality of the graph features is NetworkX, which is a Python library for studying graphs and networks, which are mathematical structures used to model pairwise relations between objects. The experiment was carried out on a Windows 10 Pro workstation with an Intel(R) Core (TM) i9-8950HK CPU @ 2.90 GHz, a 64-bit operating system, and 32.0 GB of RAM.

3.1 Data Preprocessing

Preprocessing is crucial for developing a reliable and effective ML-based detection model since it enhances classification accuracy. This phase has three steps, which are: balanced dataset, feature selection, and data cleaning.

3.1.1 Balanced Dataset

The term "balanced dataset" refers to balancing the distribution of dataset classes, as in an unbalanced dataset class model is biased towards the majority class; in such a case, the accuracy percentage obtained may be illusory [28], Oversampling, undersampling, and hybrid approaches are the most common dataset class resampling techniques. The oversampling approach is used when

the quantity of data is insufficient [29]. It mainly involves replicating the rare samples in some classes in order to equalize the distribution of classes in an unbalanced dataset. The most commonly utilized form of oversampling is the synthetic minority oversampling technique (SMOTE) [30]. The undersampling approach [31], in contrast to the oversampling approach, is used to balance dataset classes when the quantity of data is sufficient. The aim of this technique is to reduce the size of the abundant class by randomly selecting an equal number of samples from that class in order to equalize the number of samples in the rare class. For further modeling, a new, balanced dataset is retrieved. Tomek links (T-links) are a common undersampling approach for balancing dataset classes. Hybrid methods use a combination of oversampling and undersampling methods to eliminate the lack of training data produced by undersampling and to prevent overfitting problems caused by oversampling. In this work, the T-links approach was utilized to balance the datasets in both training and validation since the quantity of data was sufficient.

3.1.2 Feature Selection

The process of automatically or manually selecting the features that contribute the most to the prediction variable or output in which you are interested is known as feature selection. Irrelevant features in a dataset may reduce model accuracy and lead the model to train on irrelevant features [32]. There are several advantages to selecting features before modeling data [33], including:

·Reducing training time: fewer data processing allows algorithms to train faster and reduces algorithm complexity.
·Reducing overfitting: fewer data processing means less opportunity to make decisions based on noise.
·Improving accuracy: fewer misleading data processing means that modeling accuracy improves. To reduce the dimensionality of the data, we selected three features, which are Source IP, Destination IP, and Total Packets.

3.1.3 Data Preprocessing

The first and most crucial step in creating a ML model is preprocessing the data [34]. This step is important for improving data quality for ML module training and more accurate decision-making. Bad data might lead to inaccurate results, and so to get the appropriate results from the beginning, the dataset is examined and formatting is done. In this study, data cleansing, normalization, and transformation are utilized to create a reliable dataset [35].

1) Data Cleansing: Data cleansing is the process of identifying incorrect, irrelevant, inaccurate, or incomplete parts of the data and then deleting, replacing, or modifying the dirty or coarse data. In our work, we drop the rows in the CTU-13 dataset that have null values by using the dropna () function of Pandas.
2) Normalization: Normalization is a scaling technique that provides a consistent scale in which values are shifted and re-scaled so that they end up ranging between 0 and 1. The following is the mathematical equation for min-max feature scaling:

$$x' = \frac{x - x_{\{\min\}}}{x_{\{\max\}} - x_{\{\min\}}}, \quad 0 \leq x' \leq 1 \tag{1}$$

We get $x_{\{min\}}$ and $x_{\{max\}}$ by using *.min* ( ) and *.max* ( ) functions of pandas.

3) Transformation: Transformation is the process of converting data from one format to another. Many categorical features in the CTU-13 dataset include non-numeric data that needed to

be translated to numeric format for the ML algorithms' analyses to deal with the algebraic format. Label Encoder is a technique that many data analysts use to do label encoding. We use the SciKit-learn library to import it.

3.2 Build Model

In the third step, we build the model, including a graph-based detector and an ML model.

3.2.1 Graph-Based Detector

The traffic flows that forward to our system BotSward are bidirectional network flows. These flows are transformed into a set T that includes 4-tuple flows $T_i = \{ScrAddr; DstAddr, TotPkts, Label\}$. Where *ScrAddr* is the source host *IP* address that uniquely identifies a source host known as *sip*, *DstAddr* is the destination host *IP* address that uniquely identifies a destination host known as *dip*. *TotPkts* quantifies the number of data packets sent by source host *i* to destination host *j*, *Label* is the output class "Bot" or "Normal". The system creates a graph $G = (V; E)$, where V is the set of nodes $V = \{v_1, v_2, v_3, \ldots, v_n\}$, whereas E is the set of directed edges $E = \{e_{ij}\}$, for all i; j such that $e_{i,j} \in V$. In this study, each node denotes a unique IP address and each edge denotes the connection between one IP address to another. Set A is a set of tuples that have exclusive source and destination hosts. Where $a_x \in A$, such that $a_x = \{sip; dip, TotPkts\}$. The set of nodes V is a union of source and destination hosts from set A such that

$$V = \bigcup_{a_X \in A} \{sip \cup dip\} \tag{2}$$

For every a_x in A, There exist directed edges $e_{i;j}$ and $e_{j;i}$ from v_i to v_j and v_j to v_i, respectively, such that $sip_x = v_i$ and $dip_x = v_j$. Therefore,

$$E = \bigcup_{a_X \in A} \{(sip_x, dip_x) \cup (dip_x, sip_x)\} \tag{3}$$

Graph Algorithmic Properties

Understanding networks requires the use of centrality measurements, also known as graphs. These algorithms use graph theory to calculate the importance of any given node in a network. It can capture all network communication patterns and disregard packet payloads, focusing on the topological communications structure between hosts. Every graph feature has a degree of importance, so you need to understand how they work to find the best one for your graph visualization applications. We present and discuss the following set of important Centrality Measures for Graph-Based features that are widely utilized which are DC, BC, CC, Alpha Centrality (AC), local clustering coefficient (LCC), and Eigenvector centrality (EC) Where BotSward replaced BC With CC.

Definition 1 (Indegree and outdegree):

In Degree (ID) can be described as the total number of head ends into a particular node coming from adjacent nodes in a directed graph (arrows pointing towards the node). A high ID value of node implies that the node is the crucial and pivotal node which could be C&C servers or bots, where the adjacent nodes' tendency to create additional connections, whereas a low value implies the opposite. In contrast, The out-degree (OD) is the total number of the tail of the edge ends going out of a particular node to adjacent nodes in a directed graph (arrows pointing outwards from the node). A high OD number for a node indicates that it makes more connections with other adjacent nodes, whereas a low value indicates the opposite. Bots tend to create additional connections with other possible victim machines in order to expand the botnet's reach or communicate with the C&C domain for transferring

information. As a result, on a graph, OD can be a good indicator of botnet activity. In degree weight (IDW) indicates the total amount of data packets received by a particular node from its adjacent linked nodes. Whereas the total amount of data packets sent by a particular node to its adjacent linked nodes is known as the out-degree weight (ODW) [16]. There are two types of graphs as directed and undirected graphs, Directed graphs have edges with direction whereas Undirected graphs have edges that do not have a direction. Let $G(V, E)$ be a graph where n is total number of nodes in the graph and v in V, The mathematical expression for node degree is:

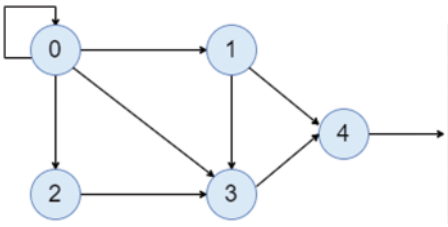
$$\deg(v) = \begin{cases} 2(n-1), & \text{In Direct graph} \\ (n-1), & \text{In undirect graph} \end{cases} \tag{4}$$

In direct graph G = (V, A) where V is set of vertices and A is a set of ordered pairs of vertices,

$deg^{+(v)}$ denotes to ID and $deg^{-(v)}$ denotes OD, G is balanced if and only if the in-and out-edges of each vertex have the same weight. Thus the graph is called a balanced directed graph when:

$$\sum_{v \in V} deg^{-(v)} = \sum_{v \in \mathbf{V}} deg^{+(v)} = |A| \tag{5}$$

Fig. 2 shows the example of ID and OD, ID of vertex $v1$ is number of edges termination from vertex v 0, whereas OD of vertex $v1$ is number of edges starting from vertex $v1$ [16].



Degree centrality of the graph

Nodes	Indegree	Outdegree
0	1	3
1	1	2
2	1	1
3	3	1
4	2	0

Figure 2: In-degree and OD example

Definition 5 (Betweenness centrality):

BC of a node $v_i \in V$ is a measure of centrality in graph theory based on the number of shortest paths from all nodes to all others that pass through that node. The mathematical expression for node BC is:

$$f(v) = \sum_{i \neq v \neq j} \frac{\sigma_{ij}(v)}{\sigma_{ij}} \tag{6}$$

where σ_{ij} is the total number of shortest paths from node pairs $v_j, v_k \in V$ and $\sigma_{ij}(v)$ is the number of shortest paths that pass-through node v_j, this feature has a high computational overhead with time complexity [16,23].

$$\mathbf{O}\left(|\mathbf{V}| \cdot |\mathbf{E}| + |V|^2 \cdot \log |\mathbf{V}|\right) \tag{7}$$

Node BC can be a valuable feature for detecting botnets, particularly in P2P botnets when bots are more interconnected and there is no central C&C structure. However, it has the potential to alienate bots when they make their first connections, when the bots' IDW and ODW are low. As a result, it would be more favorable for the network's shortest routes to pass via the host. There is an inverse relationship between node's DC and node's BC, when the IDW and ODW increase, the BC of a node decreases immensely, as it is less favored for being included in shortest paths.

Definition 6 (Local Clustering Coefficient):

LCC of a vertex (node) in a graph indicates how close its neighbors are to being a complete graph connected (clique). It has a lower computational overhead. The mathematical expression of the LCC for node a can be given by:

$$LCC_v = \frac{2.N_v}{k_v\,(k_v - 1)} \tag{8}$$

where node known as $v\,k_v$ indicates the number of node's neighbors v and N_v indicates the number of link connected pairs between all neighbors of node v where the output always between $0 \ll Lcc\,(v) \ll 1$. $Lcc\,(v)$ feature can play a significant impact to distinguish malicious hosts behavior especially in P2P botnets detection. As bots, successfully infected hosts have a greater Lcc [16].

Definition 7 (PageRank algorithm):

PR is a Google Search algorithm that ranks web pages in search engine results and way of measuring the importance of website pages. According to Google, PR calculates the importance of a website by measuring the quantity and quality of links that point to this page. Intuitively, A node that is often connected frequently is important, and nodes connected to important nodes are also considered important. This rule corresponds to bots and C&C servers in botnet detection. The definition for PR score of node u as

$$PR\,(u) = \frac{1-d}{N} + d \sum_{(v \in nb(u))} \frac{w\,(u,v)\,PR\,(v)}{d\,(v)} \tag{9}$$

where d is the damping factor, N is the total number of nodes in the network, u, v are web pages. $nb(u)$ represents all the neighbor nodes of node u [23].

Definition 8 (Eigenvector centrality):

EC is a measurement criterion of the influence of a node in a graph. The weight of a node in a graph is effective and important. The score of a node is influenced more by links to high-scoring nodes than by connections to low-scoring nodes, therefore each node is assigned a relative value. The EC of a node is determined by the total of the EC of all nodes connected to it. In other words, a node with a high EC is linked to other nodes with a high EC. Let, $A = \left(a_{v,w}\right)$ be the adjacency matrix where

$$a\,(v,w) = \begin{cases} 1, & \text{if node } (v) \text{ is linked to node } (w) \\ 0, & \text{if node } (v) \text{ is not linked with node } (w) \end{cases} \tag{10}$$

Then the centrality score can be given as:

$$x_v = \frac{1}{\lambda} + \sum_{w \in M(v)} a_{v,w} x_w \tag{11}$$

where $M\,(v)$ is the set of neighbors of node v and λ is a constant. Now Eq. (12) can be

rewritten as

$$Ax = \lambda\, x \tag{12}$$

Using the power technique and the Perron–Frobenius theorem, a positive solution λ with the last eigenvector exists. λ is also the largest eigenvalue related with the adjacency matrix's eigenvector. EC is an extension of DC, where DC provides equal scores for each node connected [16].

Definition 9: (Alpha centrality):

The concept of AC was inspired by social network research, EC refers to a node's relative weight in the network, with connections to high-scoring nodes contributing more to the v_i score [16]. Hence, AC is given as

$$x_i = \alpha A_{i,j}^T x_j + e_i \tag{13}$$

where α is influence factor that controls focus between external sources to internal influence, A_i is the adjacency matrix, and e_i is the external influence of node v_i [16].

Definition 10: (Closeness centrality)

CC is a method of detecting nodes that eligible efficiently transmit information throughout a graph. The concept of CC is based on the average farness (inverse distance) of a node to all other nodes. Nodes with a high closeness score have the shortest distances to all other nodes to spread information quickly, these nodes in the graph have important influence of the network. The mean distance between a vertex and other vertices is measured by CC [36].

For node v_i, the closeness is calculated as the average shortest path between that node and all other nodes in the graph G. This is, let $d\left(v_i, v_j\right)$ be the shortest path between v_i and v_j, the closeness is calculated as

$$c_c = \frac{(n-1)}{\sum_{(v=1)}^{(n-1)} d\left(v, u\right)} \tag{14}$$

where $d\left(v, u\right)$ is the shortest-path distance between v and u, and n is the number of nodes that can reach u.

Graph Features Importance

Feature importance refers to a class of techniques for assigning scores to input features to a predictive model that indicates the relative importance of each feature when making a prediction. In BotSward, we applied all features discussed above and we found CC is the highest relative importance by 40%, followed by PR, which has a relative importance of nearly 30%. Whereas BC and LCC are the lowest features importance. Thus, BotSward discarded the BC and the LCC and applied the following set of important features relative which are DC, CC, and PR. Fig. 3 presents all the relative importance of each graph feature.

3.2.2 Machine Learning Analysis Techniques

The features generated previously are crucial and are used by ML and DL models for the classification of botnets. To train and evaluate these models, we used the CTU-13 dataset, large capture of real botnet traffic mixed with normal traffic and background traffic. The models that were trained in this study are Random Forest (RF), Gradient Boosting Classifier (GBC), Logistic Regression (LR), Stochastic Gradient descent (SGD), Decision Tree (DT), K-Nearest Neighbor (KNN), SVM, and DNN [37,38].

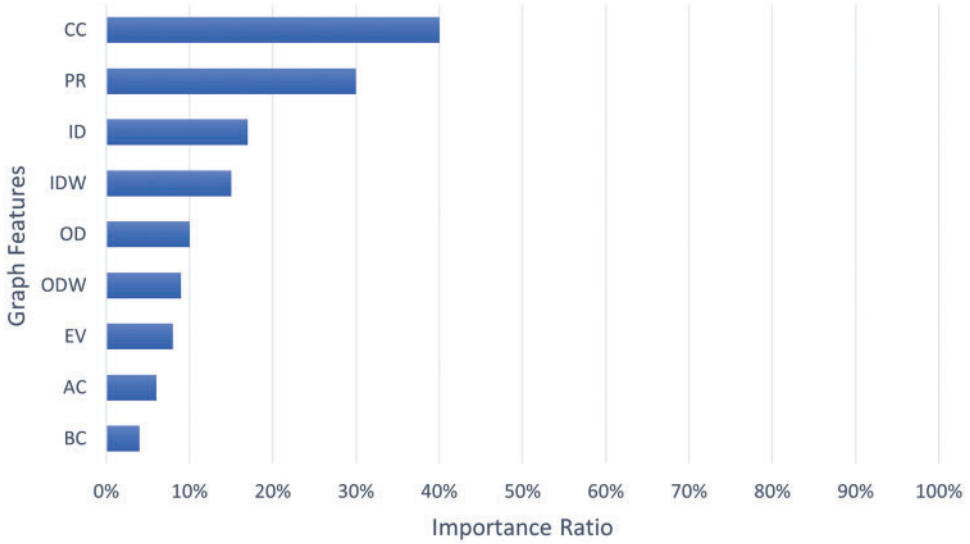


Figure 3: Graph features importance

4 Performance Evaluation

4.1 Dataset

We used the CTU-13 dataset, which is one of the largest publicly available labeled datasets and includes botnet traffic mixed with normal traffic and background traffic [39]. It was developed in 2011 at the CTU and distinguishes 13 scenarios for different botnet attacks. In CTU-13, there are various types of botnet samples, such as IRC, Port Scan, Click Fraud Spam traffic, Fast Flux, and DDoS attacks [1]. The distinctive characteristics of the CTU13 dataset are that there are real botnet attacks, real-world traffic, and multiple types of botnets. Firstly, to have a clear idea about our analysis, we concatenate all 13 scenarios' datasets that include benign and bot traffic. For these categories, 60% of the cases are chosen as training data, while the rest of the dataset, 40%, is used as validation data. However, the dataset is high-dimensional and largely class-imbalanced. The botnet traffic distribution in the CTU-13 dataset is just 1.5% of the entire network traffic. In this proposed work, the Tomeklinks under-sampling approach (T-links) is used for handling the data imbalance. Tab. 2 shows the distribution of normal and bot flows in each scenario and the botnet used.

4.2 Performance Metrics

A confusion matrix is a table depicting the performance measurement technique for the ML classification model by presetting and relating the botnet detection and mitigation scheme. A set of confusion matrix performance measures was used to evaluate the botnet detection models' robustness and effectiveness in classifying normal and attack traffic, which requires a high detection rate, high accuracy, and low false alarm rate. It contains four primary values that are utilized to generate the performance indicators listed below with the following short explanation for this technique:

- True Positive (TP): Total actual number of attack records correctly classified.
- False Negative (FN): Total actual number of attack records incorrectly classified.
- False-positive (FP): Total actual number of normal records incorrectly classified.
- True Negative (TN): Total actual number of normal records correctly classified.

Table 2: Scenarios from the CTU-13 dataset used for running experiments

Scenario	Total flows	Normal flows	Bot flows
1	2,824,636	39,933 (1.41%)	2,784,703 (98.58%)
1	1,808,122	1,787,181 (98.84%)	20,941 (1.16%)
2	4,710,638	4,683,816 (99.43%)	26,822 (0.57%)
3	1,121,076	1,118,496 (99.77%)	2,580 (0.23%)
4	129,832	128,931 (99.31%)	901 (0.69%)
5	558,919	554,289 (99.17%)	4,630 (0.83%)
6	114,077	114,014 (99.94%)	63 (0.06%)
8	2,954,230	2,948,103 (99.79%)	6,127 (0.21%)
9	2,087,508	1,902,521 (91.14%)	184,987 (8.86%)
10	1,309,791	1,203,439 (91.88%)	106,352 (8.12%)
11	107,251	99,087 (92.39%)	8,164 (7.61%)
12	325,471	323,303 (99.33%)	2,168 (0.67%)
13	1,925,149	1,885,146 (97.92%)	40,003 (2.08%)

Based on the defined confusion matrix values, the most widely adopted four metrics namely, the most widely adopted four metrics namely, Accuracy (ACC), Precision (P), Recall (R), F-measure (F), were used as in most previous botnet detection literature [40,41]. These metrics are defined as follows,

$$ACC = \frac{(TN + TP)}{(TP + FP + TN + FN)} \tag{15}$$

$$P = \frac{TP}{(TP + FP)} \tag{16}$$

$$R = \frac{TP}{(TP + FN)} = \text{sensitivity} = TPR \tag{17}$$

$$F1 - F = 2.\frac{P.R}{(P + R)} \tag{18}$$

4.3 Results

This section summarizes the findings and contributions made, and presents a performance evaluation of the results obtained from the experiment. The purpose of this evaluation was to see how well the centrality measurement algorithms and ML-based classifications performed. The results demonstrate the effectiveness of the proposed BotSward in mitigating botnet attacks.

4.3.1 Performance of Preprocessing Experiment on CTU-13 Datasets

One concern about the findings for the graph features was about the execution time of the centrality measures. Thus, we did two experiments to calculate the execution time of these measures. The first experiment yielded that the compute all graph features discussed in Section 3 for determining the most time-consuming. We found that this method can deliver good results with high accuracy (reaching 99%), but its calculation of BC was quite time-consuming and expensive. The second

experiment yielded that the compute all graph features discard BC. We found that this method can deliver good results with high accuracy (reaching 99%) and with low time consumption. Tab. 3 shows the comparison between the results of the two experiments. Fig. 4 shows the Experimenting time of BotSward when using BC and after excluded BC.

Table 3: Comparing of experimenting time with BC and without

DS	Nodes	All features with BC (Seconds)	Result	All features without BC (F)	Result
1	8283	85.8262	99%	23.226	100%
2	3628	22.5601	98%	11.6547	98%
3	6345	77.1845	83%	34.8127	83%
4	2235	8.0149	94%	5.5411	93%
5	357	0.1292	93%	0.0602	90%
6	1471	1.1609	99%	0.3392	99%
7	280	0.0939	33%	0.055	33%
8	6385	79.3971	100%	27.6409	100%
9	5950	96.5348	98%	60.2349	99%
10	2342	39.3802	75%	33.6711	75%
12	735	0.2778	100%	0.1117	89%
13	4728	61.5901	99%	18.749	99%

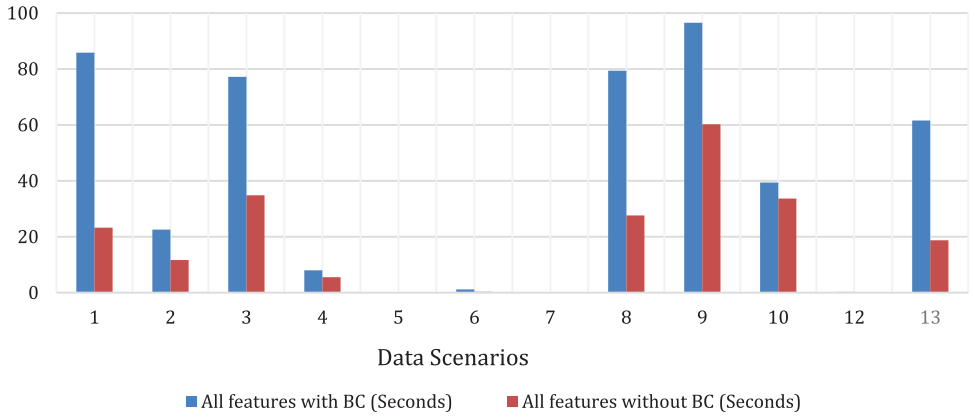


Figure 4: Experimenting time

4.3.2 Performance of Classification Algorithms Experiment on CTU-13 Datasets

In order to ensure the best results from the classifier, grid searching can be applied across ML to discover the optimal hyperparameters for a model that provide the most "accurate" predictions. Grid-search will create a model based on each possible parameter combination. It iterates through every parameter combination and stores a model for each one. Several state-of-the-art binary classification methods are experimentally assessed in the context of botnet detection. Eight widely used classification algorithms are discussed, where the predictive powers of RF, GBC, LR, SGD, SVM, DT, and

KNN are contrasted. For most of these algorithms, we report on several models using alternative hyperparameter optimization techniques to obtain the best parameters for a given model. Tab. 4 shows the results of the grid search along with the range of parameters used, and the best parameter values are highlighted [42].

Table 4: Hyperparameter search comparison (Grid *vs.* Random)

Classifier	Random		Grid search	
	Parameter	ACC	Optimal	ACC
RF	criterion = 'gini, n_estimators = 700, max_depth = 6	97%	Criterion = 'entropy' n_estimators = 300, max_depth = 8	99%
GBC	n_estimators = 100, max_depth = 3	98%	n_estimators = 300 max_depth = 4	99%
LR	penalty = l2, solver = 'sag, C = 1.0, random_state = 33	81%	Penalty = 'none', Solver = 'newton-cg', C = 0.01	82%
SGD	penalty = l2, loss = 'squaredloss, learning_rate = 'optimal' random_state = 33	23%	penalty = l2, eta0 = 0.1, learning_rate = 'constant'	50%
SVM	kernel = 'rbf', max_iter = 100, C = 1.0, Gamma = 'auto'	72%	max_iter = 300, C = 0.1, Gamma = 'auto'	76%
DT	criterion = 'gini', max_depth = 3, random_state = 33	97%	Criterion = 'gini', max_depth = 8	99%
KNN	n_neighbors = 5, weights = 'uniform', algorithm = 'auto	92%	criterion = 'gini', max_depth = 8	94%

4.3.3 Discussion

The results demonstrated in this study match state-of-the-art methods and go beyond previous reports, showing a better result than in other studies. Graph-based features in previous phases extract full communication patterns to enhance traffic classification for the learning algorithms. Our findings in preprocessing suggest that excluding the BC feature and calculating PR, EC, and CC saves around 69.5% of time while maintaining the same high accuracy 99% in the classification models. RF, GBC, LR, SGD, SVC, DT, KNN, and DNN classification success is attributed to their operational procedures and ability to select the optimal parameters using the grid-search technique. As a result, the SGD algorithm had the lowest values in all performance metrics, providing an accuracy of 50% when compared to other algorithms due to SGD evaluating the error for each training example within the dataset. This means that the parameters for each training example are updated one by one. Thus, the frequency of updates causes noise in the gradients, which has an impact on convergence and affects the

classification accuracy. The SVM algorithm had moderate results in all performance metrics, providing an accuracy of 76% when compared to other algorithms due to the classifiers not being efficient with very large datasets. KNN scored higher than the LR by approximately 14.64% in classification accuracy, and even outperformed SVM by 23.69% for the CTU-13 dataset. The results in Tab. 5 prove that KNN performs better than LR SVM for classification precision in complex situations [43].

Table 5: Performance comparison of ML algorithms

Classifier	Precision (P)	Recall (R)	F-score (F)	Accuracy (ACC)
RF	99%	99%	99%	99%
GBC	99%	99%	99%	99%
LR	86%	82%	81%	82%
SGD	75%	50%	34%	50%
SVM	79%	76%	75%	76%
DT	98%	98%	98%	98%
KNN	94%	94%	94%	94%
DNN	95%	95%	95%	95%

The RF and DT models scored almost the same results, up to 99% and 98%, respectively, since they execute the classification based on building decision trees to make quick data-driven decisions, as a DT is easier to interpret and faster to train on large datasets, particularly linear ones. The RF model needs rigorous training and consumes more training time, up to eight times more than the DT training time recorded in our experiment. Due to the additive learning technique, which uses gradient descent to identify challenges in the learners' predictions and improve weak learners' predictions, GBC attained a superior performance of up to 99% as well as RF compared to other algorithms. Also, GBC overcomes the overfitting problem and attains full computational resource utilization.

Our approach is robust even when the dataset includes missing data because we are just collecting source IP, destination IP, and total packets, which are usually available in each flow of traffic, and then from these features, we extract 7 graph features to feed the ML model for bot detection.

5 Evaluation Approach and Discussion

The evaluation of our proposed classification model is based on two aspects: performance of preprocessing to state-of-the-art and performance of the classification model to state-of-the-art.

5.1 Performance Comparison of Preprocessing to State-of-the-Art

To validate the effectiveness of our preprocessing steps, we evaluated the performance of the same ML algorithms and same datasets described earlier with different methods for extracting the features. Here we compare the results of the proposed method with BotSward with those of the traditional methods in [44] and [45]. Our proposed work on BotSward and these studies all used the same CTU-13 datasets and ML classifiers, which were RF and DT, but the methodologies for extracting the features were different. [44] and [45] used conversation-based features, while we used graph-based features. The framework in [44] employed the RF model to extract conversation-based features, and in their study, the RF had the greatest detection rate of all the classification methods, reaching up to 93.6%, with only a 0.3% false alarm rate, which is 10 times lower than detection based on traffic flow features

in [46]. Also, [45] proposed an effective two-stage traffic classification method to detect botnet-based on DT on conversation features. The DT algorithm's success rate was as high as 94.4%. From these results, it is clear that BotSward obtained the most robust results, with 99% with both RF and DT and only a 0.001% false alarm rate. The comparison of these results is summarized in Tabs. 6 and 7.

Table 6: Performance of [44] compared to BotSward

Algorithm	Features	Dataset	Classifier	Recall (R)	F-score (F)	Accuracy (ACC)
[44]	Conversation-based	CTU-13	RF	93.6%	93.6%	93.6%
BotSward	Graph-based	CTU-13	RF	99%	99%	99%

Table 7: Performance of [45] compared to BotSward

Algorithm	Features	Dataset	Classifier	Recall (R)	F-score (F)	Accuracy (ACC)
[45]	Conversation-based	CTU-13	DT	94.4%	94.4%	94.4%
BotSward	Graph-based	CTU-13	DT	98%	98%	99%

5.2 Performance Comparison of Classification Algorithm to State-of-the-Art

To validate the effectiveness of our classification model in BotSward, we evaluate the performance of the different ML classifier algorithms using the same feature extraction methods and dataset. We used the CTU-13 dataset to compare our model to the state-of-the-art in graph-based botnet detection methods, namely BClus, CAMNEP, BotHunter, BotGM, and Botchase, which are described in Section 2. Tab. 8 reports the results for each solution and all scenarios from the test set, which includes Scenarios 1, 2, 6, 8, and 9, as recommended in [26]. Our results are very competitive as we reached an accuracy of between 97% and 100%, while other studies with BClus, CAMNEP, and BotHunter only provide an accuracy of between 30% and 50%. Only Botchase achieved up to 99% accuracy for Scenario 9, but it achieved moderate accuracy in the remaining scenarios. To compare BotSward, Botchase, and BotGM for graph-based botnet detection, we used the performance metrics for Scenarios 1, 2, 8, 6, and 9 to measure how well they worked.

Table 8: Accuracy of different algorithms evaluated in [26] and compared to BotSward

Classifier	1	2	6	8	9
BClus	50%	50%	40%	30%	40%
CAMNEP	50%	40%	40%	50%	50%
BotHunter	40%	30%	38%	42%	40%
BotGM	91%	78%	95%	89%	83%
BotChase	98%	97%	94%	84%	99%

(Continued)

Table 8: Continued

Classifier	1	2	6	8	9
BotSward	99%	97%	99%	100%	99%

BotChase *vs.* BotSward, according to the BotChase experiment, they applied ID, OD, ODW, BC, LCC, and AC graph-based algorithms, and therefore they employed a two-layer detection approach based on supervised and unsupervised learning. However, the BC feature used in Botchase has a high computational overhead, with a time complexity of (O|V|.|E| + |V|2. log |V|). In contrast, BotSward focuses on the measures of centrality used in Botchase and replaces BC with PR, EC, and CC, followed by a one-layer detection technique based on GBC ensemble ML algorithms. In Scenario 8, Botchase achieved 84%, whereas BotSward achieved outstanding results with 100%. Overall, the accuracy in BotSward overcame that obtained in Botchase by the ratio of 2% to 16% and reduced the time complexity by the ratio of 69.55%, as shown in Tab. 8.

Reference [23] *vs.* BotSward, [23] proposed an anomaly-based botnet detection method by hybrid analysis of flow-based features and graph-based features of network traffic. They extracted 7 graph features from network traffic, which are ID, OD, IDW, ODW, LCC, BC, and PR. Their results achieved a good performance with a 96.62% F-score. In contrast, BotSward focuses on the bot classification by measuring the centrality used in [23] and replacing BC with CC [23] achieved a moderate performance, with a 96.62% F-score compared to a high performance of more than 99% in BotSward.

BotGM *vs.* BotSward, while both of these systems use graph methodologies in attempting to identify behavior that is malicious or outlying, the BotGM model only achieved accuracy of up to 95% in Scenario 6 of the test datasets, in contrast to 99% for BotSward. In addition, the lower accuracy of the BotGM model was shown in Scenario 2 with 78%, and in contrast, BotSward achieved 97%, as shown in Tab. 8. However, BotGM generates multiple graphs for every single host, which entails a high overhead. According to our results, it can be concluded that BotSward outperformed BotGm in terms of lower overhead and higher accuracy, as shown in Fig. 5.

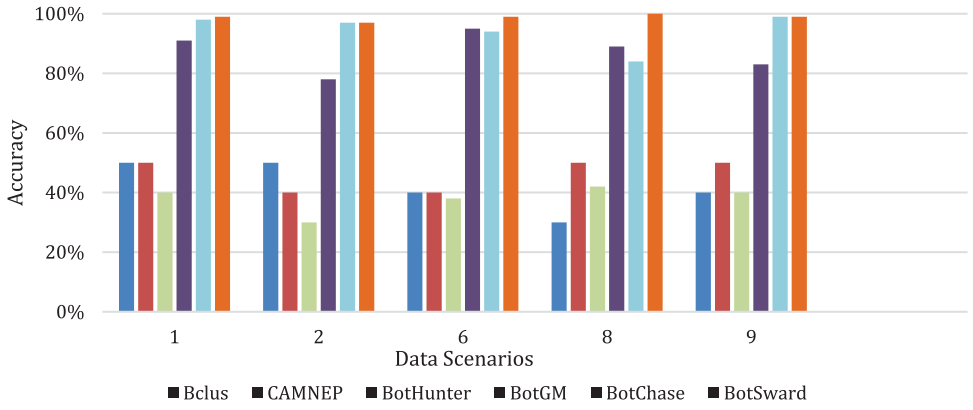


Figure 5: Accuracy of different algorithms evaluated in [26] and compared to BotSward

In summary, using graph-based features in the ML approach provides robustness in defending against unknown attacks and complicated communication patterns, and provides enhanced resource utilization, a low error rate, a high detection rate, and detection and mitigation of botnet attacks. Put

simply, BotSward is suitable to work with a variety of datasets because it only requires the extraction of source IP, destination IP, and total packets from the dataset, and then we can calculate measures of centrality for graph-based botnets, which include DC, CC, EC, and AC, and then use the ML classification to detect unknown botnets by using these features.

6 Conclusion

This paper proposes BotSward, a botnet attack defense approach based on centrality measures for graph-based and ML algorithms. BotSward utilizes centrality measures for graph-based, which employs a set of important graph algorithm features that are widely utilized. The graph-based feature extraction in all the previous studies suffered from high computational overhead because of the time complexity of the BC, and so we suggest excluding BC and LCC, and replacing them with CC, as we found the same high accuracy with a time reduction of up to 69.5%. Our proposed solution then employs a detection model based on the ML algorithm to distinguish different botnet attacks from normal traffic. We compared various types of ML such as RF, GBC, LR, SGD, SVM, DT, KNN, and DNN and evaluated them using the CTU-13 dataset, and GBC showed the top accuracy, up to 99% in all dataset scenarios and 100% in some scenarios compared with other ML algorithms with a false positive rate as low as 0.0001%. BotSward is also able to detect bots that rely on different protocols and proves robust against unknown attacks. In future work, we intend to deploy our BotSward approach in the Software-Defined Network (SDN) environment, which is an emerging network architecture that provides centralized administration through a single controller that decouples the control and data planes, addressing the limitations of traditional networks. However, this evolution has made the controller a critical target for malicious users to launch attacks such as DDoS attacks, where botnets provide platforms for DDOS. Several ways to stop botnet attacks in SDNs have been discussed, but the problems still exist.

Funding Statement: The authors received no funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. Shinan, K. Alsubhi, A. Alzahrani and M. U. Ashraf, "Machine learning-based botnet detection in software-defined network: A systematic review," *Symmetry*, vol. 13, no. 5, pp. 866, 2021.

[2] S. Heron, "Working the botnet: How dynamic DNS is revitalising the zombie army," *Network Security*, vol. 15, no. 1, pp. 9–11, 2007.

[3] M. A. Kamal, L. M. Ibrahim and A. A. Al-alusi,"Dolphin and elephant herding optimization swarm intelligence algorithms used to detect neris botnet," *Journal of Engineering Science and Technology*, vol. 15, no. 5, pp. 2906–2923, 2020.

[4] A. Izzillo and A. Pellegrini, "Graph and flow-based distributed detection and mitigation of botnet attacks," M.S.thesis, Dept. Engineering in Computer Science,University of Rome, Roma, Italy, 2021.

[5] T. Lange and H. Kettani, "On security threats of botnets to cyber systems," in *Proc. IEEE 6th Int. Conf. on Signal Processing and Integrated Networks (SPIN)*, Noida, India, pp. 176–183, 2019.

[6] A. Blaise, M. Bouet, V. Conan and S. Secci, "Botfp: Fingerprints clustering for bot detection," in *IEEE/IFIP Network Operations and Management Symp. (NOMS)*, Budapest, Hungary, pp. 1–7, 2020.

[7] W. N. H. Ibrahim, S. Anuar, A. Selamat, O. Krejcar, R. G. Crespo *et al.,* "Multilayer framework for botnet detection using machine learning algorithms," *IEEE Access*, vol. 9, pp. 48753–48768, 2021.

[8] I. Ghafir, J. Svoboda and V. Prenosil, "A survey on botnet command and control traffic detection," *Int. J. Adv. Comput. Netw. Secur.*, vol. 5, no. 2, pp. 7580, 2015.
[9] M. Abualkibash, "Machine learning in network security using knime analytics," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 11, no. 5, pp. 564–578, 2019.
[10] H. R. Zeidanloo, M. J. Z. Shooshtari, P. V. Amoli, M. Safari and M. Zamani, "A taxonomy of botnet detection techniques," in *Proc. IEEE 3rd Int. Conf. on Computer Science and Information Technology (ICCSIT)*, Chengdu, China, pp. 158–162, 2010.
[11] J. Vania, A. Meniya and H. Jethva, "A review on botnet and detection technique," *International Journal of Computer Trends and Technology*, vol. 4, no. 1, pp. 23–29, 2013.
[12] R. Limarunothai and M. A. Munlin, "Trends and challenges of botnet architectures and detection techniques," *Journal of Information Science and Technology*, vol. 5, no. 1, pp. 51–57, 2015.
[13] J. Wang and I. C. Paschalidis, "Botnet detection based on anomaly and community detection," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 392–404, 2016.
[14] A. Karim, R. B. Salleh, M. Shiraz, S. A. A. Shah, I. Awan *et al.,* "Botnet detection techniques: Review, future trends, and issues," *Journal of Zhejiang University Science C*, vol. 15, no. 11, pp. 943–983, 2014.
[15] R. C. Fernandez, D. Deng, E. Mansour, A. A. Qahtan, W. Tao *et al.,* "A demo of the data civilizer system," in *Proc. ACM Int. Conf. on Management of Data (MOD)*, Chicago, USA, pp. 1639–1642, 2017.
[16] A. Abou Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "Botchase: Graph-based bot detection using machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 15–29, 2020.
[17] B. Venkatesh, S. H. Choudhury, S. Nagaraja and N. Balakrishnan, "Botspot: Fast graph based identification of structured p2p bots," *Journal of Computer Virology and Hacking Techniques*, vol. 11, no. 4, pp. 247–261, 2015.
[18] R. Biswas and S. Roy, "Botnet traffic identification using neural networks," *Multimedia Tools and Applications*, vol. 80, no. 16, pp. 24147–24171, 2021.
[19] M. S. Gadelrab, M. ElSheikh, M. A. Ghoneim and M. Rashwan, "Botcap: Machine learning approach for botnet detection based on statistical features,"*International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 10, no. 3, pp. 563, 2018.
[20] S. Miller and C. Busby-Earle, "The role of machine learning in botnet detection," in *Proc. IEEE 11th Int. Conf. for Internet Technology and Secured Transactions (ICITST)*, Barcelona, Spain, pp. 359–364, 2016.
[21] E. B. Beigi, H. H. Jazi, N. Stakhanova and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proc. IEEE Conf. on Communications and Network Security (CNS)*, San Francisco, CA, USA, pp. 247–255, 2014.
[22] P. Gahelot and N. Dayal, "Flow based botnet traffic detection using machine learning," in *Proc. ICETIT*, Cham, Switzerland, Springer, pp. 418–426, 2020.
[23] Y. Shang, S. Yang and W. Wang, "Botnet detection with hybrid analysis on flow based and graph based features of network traffic," in *Proc. ICCCS*, Switzerland, Springer, pp. 612–621, 2018.
[24] S. Lagraa, J. François, A. Lahmadi, M. Miner, C. Hammerschmidt *et al.,* "Botgm: Unsupervised graph mining to detect botnets in traffic flows," in *Proc. 1st Cyber Security in Networking Conf. (CSNet)*, Rio de Janeiro, Brazil, IEEE, pp. 1–8, 2017.
[25] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *USENIX Security Symp.*, Vancouver, BC, Canada, vol. 7, pp. 1–16, 2007.
[26] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014.
[27] A. R. Vishwakarma, "Network traffic based botnet detection using machine learning," M.S. thesis, Dept. Computer Science, San Jose State University, Washington, United States, 2020.
[28] D. Gonzalez-Cuautle, A. Hernandez-Suarez, G. Sanchez-Perez, L. K. Toscano-Medina, J. Portillo-Portillo *et al.,* "Synthetic minority oversampling technique for optimizing classification tasks in botnet and intrusiondetection-system datasets," *Applied Sciences*, vol. 10, no. 3, pp. 794, 2020.

[29] D. Zhang, J. Hu, F. Li, X. Ding, A. K. Sangaiah *et al.,* "Small object detection via precise region-based fully convolutional networks," *Computers, Materials and Continua*, vol. 69, no. 2, pp. 1503–1517, 2021.
[30] S. Pokhrel, R. Abbas and B. Aryal, "Iot security: Botnet detection in iot using machine learning," arXiv preprint arXiv:2104.02231, 2021.
[31] M. M. Rahman and D. N. Davis, "Addressing the class imbalance problem in medical datasets," *International Journal of Machine Learning and Computing*, vol. 3, no. 2, pp. 224, 2013.
[32] Z. M. Algelal, E. A. G. Aldhaher, D. N. Abdul-Wadood and R. H. A. AlSagheer, "Botnet detection using ensemble classifiers of network flow," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 3, pp. 2543, 2020.
[33] C. Hung and H. Sun, "A botnet detection system based on machine-learning using flow-based features," in *Proc. of the SECURWARE 2018: The Twelfth Int. Conf. on Emerging Security Information*, Italy, 2018.
[34] R. Mishra and S. K. Jha, "Survey on botnet detection techniques," in *Internet of Things and Its Applications, Lecture Notes in Electrical Engineering*, vol. 825. Singapore: Springer, pp. 441–449, 2022.
[35] D. Dagon, C. C. Zou, and W. Lee, "Modeling botnet propagation using time zones.," *NDSS*, vol. 6, pp. 2–13, 2006.
[36] A. Sanatinia and G. Noubir, "Onionbots: Subverting privacy infrastructure for cyber attacks," in *Proc. IEEE 45th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN)*, Janeiro, Brazil, pp. 69–80, 2015.
[37] J. Wang, Y. Zou, P. Lei, R. S. Sherratt and L. Wang, "Research on recurrent neural network based crack opening prediction of concrete dam," *Journal of Internet Technology*, vol. 21, no. 4, pp. 1161–1169, 2020.
[38] S. He, Z. Li, Y. Tang, Z. Liao, F. Li *et al.,* "Parameters compressing in deep learning," *Computers Materials & Continua*, vol. 62, no. 1, pp. 321–336, 2020.
[39] S. Ryu and B. Yang, "A comparative study of machine learning algorithms and their ensembles for botnet detection," *Journal of Computer and Communications*, vol. 6, no. 5, pp. 119, 2018.
[40] W. Wang, H. Liu, J. Li, H. Nie and X. Wang, "Using CFW-net deep learning models for X-ray images to detect COVID-19 patients," *International Journal of Computational Intelligence Systems*, vol. 14, no. 1, pp. 199–207, 2021.
[41] W. Wang, Y. Yang, J. Li, Y. Hu, Y. Luo *et al.,* "Woodland labeling in chenzhou, China, via deep learning approach,"*International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 1393–1403, 2020.
[42] D. Comaneci and C. Dobre, "Securing networks using sdn and machine learning," in *Proc. IEEE Int. Conf. on Computational Science and Engineering (CSE)*, Bucharest, Romania, pp. 194–200, 2018.
[43] J. Zhao, X. Liu, Q. Yan, B. Li, M. Shao *et al.,* "Multi-attributed heterogeneous graph convolutional network for bot detection," *Information Sciences*, vol. 537, pp. 380–393, 2020.
[44] R. Chen, W. Niu, X. Zhang, Z. Zhuo, and F. Lv,"An effective conversation-based botnet detection method," *Mathematical Problems in Engineering*, vol. 2017, pp. 334–344, 2017.
[45] R. U. Khan, R. Kumar, M. Alazab and X. Zhang, "A hybrid technique to detect botnets, based on P2P traffic similarity," in *Proc. Cybersecurity and Cyberforensics Conf. (CCC)*, Melbourne, Australia, pp. 136–142, 2019.
[46] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics," *Computers and Electrical Engineering*, vol. 50, pp. 91–101, 2016.