DETECTION OF PEER-TO-PEER BOTNETS USING GRAPH MINING

Dhruba Jyoti Borah and Abhijit Sarma

Department of Computer Science, Gauhati University
Jalukbari, Guwahati, Pin-781014, Assam India

ABSTRACT

Peer-to-Peer (P2P) botnets are significant threats to the Internet. The botnet traffic is increasing rapidly every year and impacts the entire Internet. A P2P botnet is responsible for launching various malicious activities such as DDoS attacks, click fraud attacks, stealing confidential information from bank and government websites, etc. It is challenging to detect P2P botnets because of their high resiliency against detection. This paper proposes a method that uses a network communication graph from network flow data to detect botnets. Three graph-mining techniques are used to detect bot nodes individually. The method's final result is obtained by applying an ensemble algorithm to the results of the three graph-mining techniques. A synthetic dataset from a testbed is used to assess the method's performance. In addition, the method is evaluated using a publicly available dataset. Experimental results show that the method performs with an accuracy of 99.99%, a precision of 94.29% ,and a recall of 98.02%, which is better than existing methods.

KEYWORDS

Botnet, P2P botnet, Communication graph, Cyber security, Graph-mining

1. INTRODUCTION

As the Internet becomes an integral part of our everyday life, protection against malicious activities on the Internet has become important. According to Bailey et al. [1], bad bots are software agents developed by attackers that are propagated through the Internet to gain entry into vulnerable computers, using various methodologies, including social engineering, exploiting operating systems vulnerabilities, etc. These bots communicate among themselves and establish their own network known as a botnet. After entering vulnerable computers, this software installs itself without informing the user of the computers. The attackers, also known as botmaster, sends commands to the bots to perform attacks on computers. The bots after receiving the commands from the botmaster, execute them to perform attacks like Distributed Denial of Service (DDoS) attacks, click fraud attacks, key-logging attacks, password cracking attacks ,and stealing credit card information, etc. Atefeh Zareh and Hamid Reza Shahriari in [2] says that, the botmaster can use a botnet to perform malicious mining activities in popular cryptocurrencies, like bitcoin. According to the report of IMPERVA, 2019 [3] concerning global bot traffic, the bad bot has generated 20.4% of all website traffic. On the other hand, human-generated traffic is 62.1%, and good bots have generated 17.5% of traffic.

Based on the command-and-control (C&C) topologies used by the botmasters to communicate with their bots, botnets can be classified into two categories:

Centralized topology: In this topology, botmasters use one dedicated command-and-control (C&C) server for communicating with their bots. Bot computers try to communicate with this centralized C&C server to get any instructions from the botmasters. Although these types of botnets are convenient to use because of their simplicity, it suffers from the fact that it has a single point of failure. If the central C&C server is brought down, the entire network will go down.

Peer-to-Peer or Multi-Server topology: This type of botnet uses a Peer-to-Peer(P2P) communication method to communicate among the bots. There is no central server, as such, there is no single point of failure. The botmaster can use any of them as C&C server to give commands to other bots. If the a C&C server is detected, the botmaster can switch the C&C server to another computer(s). Bringing down one bot does not affect the performance of the network. That is why this type of botnet is challenging to detect and bring down. So, the attackers prefer using Peer-to-Peer (P2P) botnets.

Based on the structure of the P2P botnets, they can be divided into three categories: leeching, parasite ,and bot-only botnet. In the parasite botnet, all the bot nodes are chosen from an existing P2P network to form the botnet. The botmaster does not worry about the communication structure of the network. On the other hand, in leeching P2P botnets, the botmaster selects bot nodes throughout the Internet, but the nodes use an existing P2P network to communicate among themselves, as reported in [4]. Finally, in bot-only botnets, botmasters develop their private networks. These types of botnets do not include benign peers. According to many researchers, MIRAI botnet, Nugache, Storm, SubSeven Bot, Bionet Bot, Attack Bot, GTBot, EvilBot, Slackbot, MTK botnet ,and Opfake are some examples of P2P botnets that cause significant loss in terms of money every year around the world [5,6,7].

The rest of the paper is organised as follows. In section 2, some previous P2P botnet detection methods are discussed. In the section 3, we discuss the proposed method to detect P2P botnets. In section 4, experimental results of the proposed method is discussed. Then we compare the result with other methods in section 5. In section 6, parameters which were tuned at the time of performing experiments are discussed. In section 7, the robustness of the proposed method is discussed. Finally, we conclude with the section 8.

2. RELATED WORK

Researchers have developed different P2P botnet detection methods. Many researchers employ machine-learning algorithms to detect P2P botnets[8,9,10,11,12,13, 44]. One such algorithm is used in [8], where the J48 decision-tree model and K-means algorithm are used to detect P2P bot nodes. The method uses six steps to detect P2P bot nodes. Non-P2P traffic is filtered out in the first step. The computers that are running P2P applications are identified in the second step. The size of the P2P application's traffic flows is determined in the third step. In the fourth step, the J48 decision-tree model is used to classify traffic flows from various P2P applications. Four common P2P applications, BitTorrent, eDonkey, Foxy, and GoGoBox, as well as one known P2P botnet flow, Waledac flows, are used to train the decision tree model in this step. If the model gets any flows of Waledac in the testing phase, it will detect the computer relating to Waledac flow as bot node flow. In the next step, the model checks whether the remaining flows contain unknown botnet traffic flows. In this stage, the authors carry out this using the K-means algorithm. However, the experiment was performed using six computers running different P2P and non-P2P applications. On the other hand, the mechanism examines every packet's payload to identify the P2P traffic, which is not feasible in a large network environment. Therefore, this mechanism is unsuitable for botnet detection in a real network.

Matija Stevanovic and Jens Myrup Pedersen [9] use a supervised machine learning algorithm to detect botnets. The method extracts 39 statistical features from each flow record in this case. These features are fed into eight supervised machine learning algorithms: the Naive Bayesian classifier, the Bayesian Network classifier, the Logistic Regression classifier, the Artificial Neural Networks, the Support Vector Machines with the linear kernel, the C4.5 decision tree, the Random Tree classifier, and the Random Forest classifier. The ISOP dataset is used to test the model. The Random Forest classifier outperforms all other machine learning algorithms, achieving 96.20% precision, 95.73% recall, and 95.96% F1.

Some researchers use the graph-based approach to detect bot nodes[14,15,16]. One such approach is used by Jing Wang and Ioannis Ch. Paschalidis in [14]. Their method consists of two stages. A social interaction graph(SIG) is created in the first stage. They group the network packets into some time window W_k, where k is a timestamp ,and create a SIG graph for timestamp k. With the help of some reference models, they try to detect abnormal SIG graphs and store them in a pool. If the pool reaches a threshold p, then the pool is used for the second stage. In the second stage, the key nodes are identified. The key nodes are defined as the nodes having high interaction with other nodes. Then a social correlation graph (SCG) is created. The SCGs identify the correlation of other nodes with the key nodes. These SCGs are parsed to a refined modularity-based community detection algorithm to detect bot nodes' communities. The whole approach uses network packets as input. Inspecting every network packet passing through a network is not feasible. On the other hand, the authors have used a small dataset that contains 396 nodes, including 136 bot nodes and 260 normal nodes. In a realistic scenario, the percentage of bot nodes will be much smaller. Finally, the authors do not evaluate the method's performance using standard metrics like accuracy, precision, and recall.

Another graph-based approach is used in [15]. According to the authors, the entire bot communication graph is required to detect all the bot nodes in a botnet, which is unrealistic. Therefore they develop a model based on modularity-based community detection algorithms and try to detect part of the botnet when partial information about it is available. They create samples of bot communication graphs by incrementing edges in the graph from 20% to 100%, increasing every time by 20%. The model tries to detect bot nodes using Newman's Leading Eigenvector and Fast Modularity methods. The experiments have been performed with different botnet topologies, botnet sizes ,and background network sizes. The authors say that the Fast Modularity algorithm-based detection model performs better than Newman's leading eigenvector-based detection model in all sample proportions and botnet topologies. The FP and FN increase when the bot nodes decrease. Furthermore, the False Positive rate increases as the sampling proportion decrease. However, the FP and FN remain below 5%. At the end of the paper, the authors conclude that the FP is small when the botnet graph is small and the background graph is large. The FP and FN increase as the size of the botnet graph increases. However, the authors conducted all their experiments in a small environment with a total network size of 7500 nodes. Furthermore, no dataset containing real-world bot nodes has been used to test the method. Finally, the authors do not use standard metrics like accuracy, precision, and recall to assess performance.

Afnan Alharbi and Khalid Alsubhi also use the graph-based machine learning approach [16]. They generate a directed graph from the input flow dataset in the first step. For each node in the graph, various features including in-degree, out-degree, in-degree weight, out-degree weight, in-degree centrality, out-degree centrality, betweenness centrality, closeness centrality, Eigen centrality, Katz centrality, PageRank centrality, hub and authority, and local clustering coefficient are calculated. They use several feature selection algorithms including Information Measure, Gini Impurity, Correlation Measure, Pearson's Correlation, and Consistency Measure to extract the best features. The model employs six machine learning algorithms, Naive Bayes, Decision Tree, Random Forests, AdaBoost, ExtraTrees Classifier, and K-Nearest Neighbours to classify the bot nodes. They test the model's performance using the CTU-13 and IOT-23 datasets. According to the author, the ExtraTrees classifier with Pearson's correlation feature set produces the best results in both datasets. This model claims to achieve 100% accuracy, precision, and recall. However, the model was tested with CTU-13 scenario 9, which does not contain any P2P bot nodes.

Flow-based approaches are also popular in detecting bot nodes among researchers[17,18,19]. Jianbing Liang et al. detect bot nodes using the flow similarity metric [17]. The traffic flows are divided into time windows in the method. Then, a vector containing the packet length sequence is calculated for each flow. The Levenshtein algorithm is then used to calculate the similarity of these vectors. If the similarity score of two vectors exceeds a certain threshold, the flows associated with the vectors are considered bot flows. A port partitioning algorithm is then used to identify the bot nodes. In the ISCX testing dataset, the method claims 1.0 TPR and 0.00706 FPR. However, the method only employs TCP flows. According to the authors, bot node communication is based on TCP. In reality, the communication among bot nodes is not restricted to TCP protocol only. Because of its connectionless behavior, bot nodes sometimes prefer UDP over TCP. As a result, using only TCP for botnet detection leads to a high number of false negatives. On the other hand, the authors do not demonstrate the method's performance using standard metrics like accuracy, precision, and recall.

Some researchers use heuristic threshold-based algorithms, where some threshold values are used to differentiate various characteristics between bot nodes and benign nodes [20,21]. C. Dillon proposes a mechanism to detect Zeus P2P bot nodes in [20], where he uses live netflow data. The detection model tries to detect the Zeus P2P bot nodes' flow. Here the author generates Zeus P2P bot traffic in a testbed. The Zeus P2P software was collected from the public sandbox malwr.com. The bot traffic is collected in a Netflow collector and then merged with benign network traffic. The detection module starts with filtering out non-P2P traffic flow. In the resultant traffic flow, the Zeus P2P bot traffic is detected using either packet ratio anomaly or the unique traffic pattern of the Zeus bot. The packet ratio is the ratio of the outgoing packets to the incoming packets of P2P applications. According to the author, the packet ratio of standard P2P applications is between 1.4 and 1.8. If the packet ratio is less than 0.4, then the flow will be considered as Zeus bot flow. On the other hand, the model also detects the Zeus bot using its unique traffic pattern. He says the Zeus bot nodes communicate after every 20 minutes with other peers to get configuration updates of the latest bot software. However, the benign P2P application has no periodic communication properties. So this property is exploited to detect the Zeus bot. The author claims that the model performs with 100% true positive and 0% false positive. However, Zeus traffic is relatively low, and as such, it is not supposed to affect the incoming and outgoing traffic ratio significantly, especially in cases where benign traffic in a computer is high. Therefore, the packet ratio cannot be a good way to identify the bot nodes in a dataset. Furthermore, the author experimented with a small testbed containing only three Zeus bot nodes. One cannot say whether the model performs well in a real dataset.

Some researchers use the anomaly-based approach to detect bot nodes [5,22,23]. Himanshi Dhayal and Jitender Kumar in [5] develop a mechanism to detect botnets by classifying network traffic in the waiting stage of the bot nodes. To detect bots, they have considered three features of the bot traffic, the bot's lifetime, search request intensities ,and the time-correlated behavior of the bot nodes. According to the authors, the bot nodes remain active in a network for getting commands from the botmaster for a longer duration of time than the benign peers of normal P2P applications. Furthermore, to get commands from the botmaster, the bot nodes send abnormally higher search requests than the benign peers of the normal P2P applications. Finally, the bot nodes sending higher search requests show similar behavior. So, this temporal correlation between search request packets can be used to detect bot nodes in a network. The authors claim 99% accuracy of the model. However, the model inspects every network packet to detect bot nodes. Therefore the model is not feasible in a large network. Furthermore, the model was tested on a dataset with a higher proportion of malicious P2P packets than benign packets. So, it is unclear whether the model performs well in a real-world scenario where bot traffic is much lower than normal traffic.

Some researchers use deep neural networks to detect bot nodes [24,25,26]. In [24], the author extracts statistical flow features based on TCP, UDP, and ICMP from the input flow graph. These features are then fed into a deep neural network model as input. The author builds the deep neural model using embedding, convolution, LSTM, and fully connected network layers. The model is tested using the CTU-13 and ISOT datasets, claiming 99.2% and 99% accuracy for ISOT and CTU-13, respectively. The model also delivers 97.3% and 99.1% F1 for ISOT and CTU-13.

In [26], the authors try to detect botnets using Multi-Layer Perceptron (MLP). They combine the CTU-13 P2P bot traffic dataset with the HIKARI benign dataset to create a botnet dataset. The four best features relevant to botnet detection are then extracted using a combination of two feature engineering techniques: CFS subset evaluation and consistency subset evaluation. The input is then fed into MLP to detect bot nodes. According to the author, the method achieves 99.9% accuracy with 100% precision, recall, and F1 results. The method, however, is a packet-based inspection method that extracts 30 features from each packet in a dataset. As a result, the method requires a significant amount of memory and CPU time to detect bot nodes in a real-world dataset containing billions of packets.

Some of the above works are host-based, and as such, one such system can detect one node of a botnet only. Botnets consist of a significant number of nodes. It is not unusual to have a million nodes in a botnet. It is unrealistic to bring down a whole botnet by bringing down a single bot scattered globally. Many of the above works use a network-based approach where data is collected from across a large network to detect a large number of bots of a botnet. However, most of them use network packet information. The volume of such traffic data is large, and it is not practicable to store and process such a large volume of data. Many researchers used data where the ratio of bot nodes to non-bot nodes is not realistic. In real botnet data, the number of bot nodes is much less than that of non-bot nodes. Consolidation of communicated data among the hosts is available in the form of network flow data. This network flow data volume is much smaller. Some earlier works used network flow data. However, either they do not test their methods in a publicly available dataset, or their results are not satisfactory.

3. Proposed Method to Detect P2P Botnets

To overcome the limitations of previous works, we propose a method for detecting a whole botnet or a significant portion of it. Only the source and destination IP addresses of each flow are retrained from network flow data collected from routers across a network. In earlier works, header data or other flow information were used to detect bot nodes in addition to IP addresses. Because we only use the source and destination IP addresses of flows in our method, the data volume is drastically reduced.

On the Internet, there exist publicly accessible both datasets. However, only one such publicly available bot dataset contains more than one P2P bot node where the bot nodes communicate among themselves, thereby preserving the P2P communication structure. This is the CTU-13 botnet dataset. Our method uses a P2P communication structure to detect bot nodes. We use this dataset to validate our method. However, this dataset also contains only three P2P bot nodes. So we develop a testbed simulating a notorious P2P botnet, namely Zeus GameOver. We mix the traffic obtained from this testbed with real - world non-bot traffic to create a dataset. We also validate our method using this synthetic dataset.

3.1. The Proposed Method

In the proposed method, network flow data is collected from multiple routers. From the network flow, it constructs a communication graph. The communication graph is fed into three graph-mining algorithms. Each algorithm is aimed at identifying bot nodes. Then the result of the three algorithms is consolidated using the ensemble technique to further improve the result.

3.1.1. Transformation of Input Dataset into Communication Graph

In the first stage, flow records are collected from different routers. From the flow records, only the source IP address and destination IP address fields are retained. Then using these, a communication sub-graph is created. For each IP address, a node is created in the graph if it does not already exist. An edge from the node corresponding to the source IP address to the node corresponding to the destination IP address is added if it does not already exist. This process is repeated for each flow. Algorithm 1 shows the steps to transform the network flow records into sub-graph.

```
Algorithm 1: Transformation of network flow dataset into host communication sub-graph
Input:   Flow information
Output:  Communication sub-graph G
 1:  while there exists a flow do
 2:      Read a flow into f
 3:      Extract the source IP address src_IP and the destination IP address dst_IP of the
         flow f
 4:      if n_src corresponding to src_IP is not in G then
 5:          Add a node n_src to G
 6:      end if
 7:      if n_dst corresponding to dst_IP is not in G then
 8:          Add a node n_dst to G
 9:      end if
10:      if an edge from n_src to n_dst does not already exist in G then
11:          Add edge from n_src to n_dst
12:      end if
13:  end while
```

All the sub-graphs are collected and merged in the second stage to form the final communication graph.

3.1.2. Detection of Botnets from the Graph

After creating the communication graph, the nodes of the communication graph are clustered using graph-mining algorithms into some clusters. These algorithms cluster the nodes so that all the bot nodes are included in one cluster. The cluster containing at least one bot node is considered a bot cluster, and all the nodes in that cluster are considered bot nodes. For this, we use prior information obtained from host-based intrusion detection systems, like the honeynet project.

Our detection method consists of three different approaches. In the first approach, we use the Markov Cluster algorithm (MCL)[27,28] to detect the bot nodes. The Infomap algorithm [29] is used in the second approach and the combination of the PageRank algorithm[30] and HDBSCAN algorithm[31] is used in the third approach. To the best of our knowledge, the first two approaches, namely MCL-based and info map-based approaches, had not previously been used for botnet detection. Though PageRank has been used in previous works for botnet detection, the combination of PageRank and HDBSCAN has never been used.

All the three approaches give us three sets of independent results. Then majority voting ensemble method is used on the results of these three approaches. The result of this voting ensemble will be our final result. Fig. 1 shows the diagram of the proposed method.

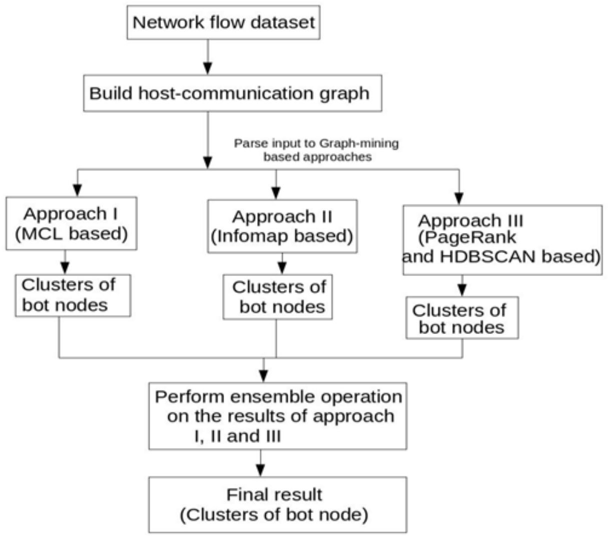
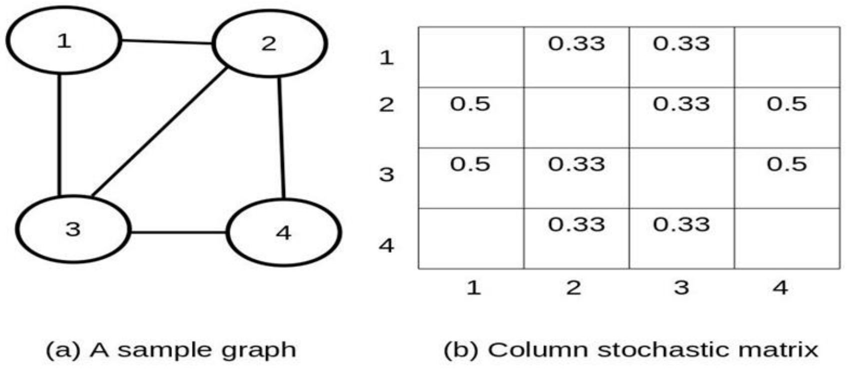

Network flow dataset
Build host-communication graph
Parse input to Graph-mining based approaches
Approach I (MCL based)
Approach II (Infomap based)
Approach III (PageRank and HDBSCAN based)
Clusters of bot nodes
Clusters of bot nodes
Clusters of bot nodes
Perform ensemble operation on the results of approach I, II and III
Final result (Clusters of bot node)

Figure 1. Diagram of the proposed method

3.1.2.1. Approach I: MCL based P2P Botnet Detection Approach

In the first approach, we use the MCL algorithm [27,28], which is mainly used in biological networks[32,33,34]. It is an unsupervised clustering algorithm for graphs. Here, a column stochastic matrix is created from the input graph, where the sum of each column is 1. Fig. 2 shows a column stochastic matrix for a sample input graph.



	1	2	3	4
1		0.33	0.33	
2	0.5		0.33	0.5
3	0.5	0.33		0.5
4		0.33	0.33	

Figure 2. A sample graph and its corresponding column stochastic matrix

The algorithm simulates random walks to get the graph's cluster structure by altering two algebraic operations: expansion and inflation. The expansion operator uses matrix multiplication (matrix squaring) to expand the matrix. The inflation operator is responsible for strengthening and weakening the current value of each entry in the graph. This operation is performed by raising the column values to non-negative power and then re-normalizing them. These two operations are repeatedly applied until a stable state of the column stochastic matrix is reached. The stable state is the state from which the column stochastic matrix does not change its state. Then retrieve the clusters of nodes from the resultant matrix so that nodes flowing into the same sink nodes are assigned to the same cluster. Finally, by using prior knowledge of some bot nodes, we identify the clusters of bot nodes present in the input graph.

3.1.2.2. Approach II: Infomap Algorithm based P2P Botnet Detection Approach

In the second approach, we use the info map algorithm [29] to detect bot nodes. This algorithm is mainly used for community detection in a network [35,36,37].

The info map algorithm is based on the map equation[38]. The map equation is used to represent the structures and their relationships of directed, weighted complex networks in a simplified fashion. Here the local interactions among groups of nodes are emphasized, which causes system-wide information flow of the entire system. Therefore, a network's modules or clusters can be identified by finding the coarse-grained description of the network's information flow. So finding the information flow and communities is performed by using coding or compression. A Random Walker (RW) is used to describe the information flow in the network. Therefore for a given modular partition of a network, there is an associated information cost of the RW. The partition with the shortest description length of the RW is regarded as the best capture of the network's community structure.

The info map runs some sub-routines iteratively and some recursively for identifying the module partition with the shortest description length of the map equation. The clusters of the nodes are identified from this shortest module partition.

3.1.2.3. Approach III: Botnet Detection using PageRank and HDBScan Algorithm

We use the PageRank algorithm[30] in the communication graph in the third approach. PageRank is a link analysis algorithm. The Google search engine first used the PageRank algorithm to rank the website in their search engine result. The PageRank algorithm is based on the premise that a web page is valuable if it has several incoming web pages or an important web page points to it. For example, if many web pages point to a webpage, then the page rank value of this page is good. On the other hand, if a web page is pointed by only one web page, which is Facebook, then the PageRank value of this page is also good.

A suspected bot node cluster includes nodes with higher PageRank values than a threshold value. However, this cluster contains many false positives. On the other hand, threshold values may also be different for other datasets. Therefore, another communication graph is created among the nodes of the suspected bot-node cluster. It will reduce the size of the input dataset. In the resultant communication graph, we use the hierarchical density-based spatial clustering of applications with noise (HDBSCAN) to isolate the bot nodes.

HDBSCAN[31] is a hierarchical density-based clustering algorithm. It extends the DBSCAN algorithm by performing DBSCAN on different epsilons values and delivering the clusters that persist over the epsilon. This allows us to find the cluster with different densities.

3.2. Simple Majority Voting Ensemble Technique

An ensemble of methods consists of independent classification methods whose results are combined using some rules to enhance more accurate results. Carlos Orrite et al. [39] state that the ensemble technique outperforms the best result produced by any individual classifiers. There are various types of ensemble techniques available. Among them, we have used the simple majority voting technique to perform an ensemble on the results of the above three approaches. We have chosen this technique because of its simplicity and high level of accuracy. According to A.F.R. Rahman et al.[40], the simple majority voting technique can be defined as follows:

If there are n classification methods with their independent solutions, then the final decision for a sample can be made by assigning the sample to a class if at least k methods agree to vote. The value of k is calculated as follows:

$$K= \begin{cases} (n/2)+ 1, \text{ if n is even,} \\ (n+1)/2, \text{ if n is odd,} \end{cases} \qquad (1)$$

Since we use three approaches to perform ensemble operation, the value of n is 3, so K is 2.

4. Experiments and Results

To evaluate the performance of the proposed method, we have used two datasets. We generated a synthetic dataset in a testbed. We also tested the performance of the method by using CTU-13[41] dataset.

4.1. Details of the Testbed Setup and Data Generation

The details of the testbed dataset is described below:

4.1.1. Motivation of the Testbed Dataset

The communication structure of a P2P botnet can be used to detect multiple bots. Researchers use lots of botnet datasets to validate their methods. However, the datasets have some deficiencies. Some datasets contain traffic of a single P2P bot node. Some datasets contain a few P2P bot nodes. In other cases, communications among most of the bots are unavailable and as such, the P2P communication structure of the botnet is not available. There are other datasets that are not publicly available. Therefore, we generate a synthetic dataset to remove these limitations. The CTU-13 dataset contains only three P2P bot nodes, though they contain other non-P2P bot nodes. We us e this dataset to compare our synthesized result with this real-world dataset.

4.1.2. Generation of the Testbed Dataset

To create a synthetic dataset, we have developed a bot software simulating the behavior of the Zeus P2P bot also known as Zeus GameOver. This bot is different from the original Zeus bot which was a non-p2p bot. To simulate Zeus GameOver, we use two technical reports, one by CERT Polska[42] and the other by Dennis Andriesse et al. [43]. The behaviour of this P2P botnet has been studied thoroughly from these technical reports. Then we developed the bot and its control software by implementing the core functionality of this botnet to obtain the same communication graph from this simulation as would have been obtained from the actual Zeus P2P botnet. Other details, like encryption of data, are not considered.

4.1.2.1. Simulation of the Zeus Bot Behaviour

In this section, we discuss the properties of the Zeus P2P botnet for ready reference. The Zeus network has a three-layered architecture consisting of a C2 layer, a C2 proxy layer, and a P2P layer. The P2P layer includes all bot nodes. The Zeus P2P botnet establishes a p2p communication network in its P2P layer. For that, each node maintains a peer list. This peer list is updated periodically. The bots in the P2P layer are called harvester nodes. They harvest data to communicate with the botmaster. These nodes take commands from the botmaster to perform tasks such as stealing confidential information from their hosts or performing DDOS attacks on a specific computer.

Initially, a peer list is hardcoded in each bot. The bots periodically send a version request message to the nodes in their peer list. If it does not receive a reply after sending such a message up to five times, it deletes that node from its peer list. On the other hand, if a node receives a version request message from a bot that is not in its peer list, it adds that node to the peer list. When the number of nodes in the peer list falls below a threshold, then the bots send a peer list request to all the bots in its peer list. The threshold below which a peer list request is sent is 25. The bots receiving these messages send a list of 10 peers in reply. These ten peers are selected based on how close the ID of the peers is to the ID of the requesting peer. The size of the peer list is limited to 50. The version request message is used to keep track of the bot's binary and configuration version number and update itself to a newer version if necessary.

Each bot communicates to the upper layer through a bot in the P2P layer, which is designated as a proxy bot. The botmaster designates one or more P2P bots as proxy bots by sending commands. The bots receiving such a command announce that information by sending a proxy announcement message to the bots in its peer list. The receiving bots send these messages to their peers and so on. The TTL field limits the life of this message. Apart from the above, bots communicate with the C2 proxy layer through the proxy bots to receive commands and send data.

The bots use different messages to perform all of the above tasks. Each bot is given an identification number(ID). Each packet contains a packet header and payload. The format of the packet header is given in fig. 3.

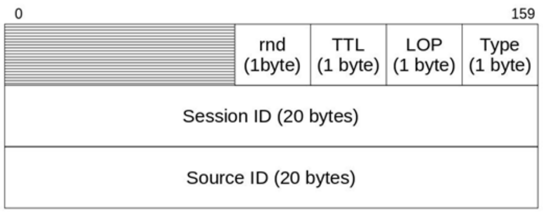


Figure 3. Zeus P2P packet header

The red field in the packet header is used for encryption. The TTL field is used to limit the number of hopes a message can be re-transmitted. LOP is the padding's length. The type field indicates the message type. The session ID identifies the specific session. The source ID field contains the ID of the sending bot. Not all messages require a payload field. The messages not having a payload field are identified by their type field. The format of the different payload field is given below:

The value 0x00 in the type field indicates the version request message. Generally, this message contains no payload. However, if the requesting node's proxy list is too short, it sends a little-endian integer 1 followed by four random bytes as payload. This is an indication to the receiving bot to send its proxy list. The receiving bot sends a version reply message. The binary version number, configuration file version number, and TCP port number are followed by 12 random bytes in the payload of the version reply message. The TCP port is used by the receiver of the message to communicate back the new version of the bot. The payload format of the version reply format is given in fig. 4.

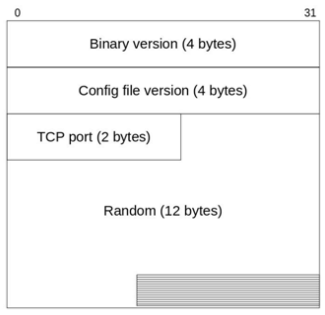


Figure 4. Format of Zeus P2P version reply message payload

If the version request message includes a proxy list request indication, the receiving bot will additionally send a separate proxy reply message. Fig. 5 depicts the structure of each proxy node in the proxy reply message payload. The proxy reply message contains up to 4 proxies, each of which is RSA-2048 signed. The size of each proxy entry is 304 bytes.

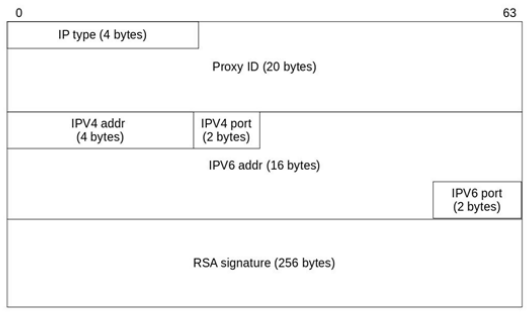


Figure 5. Format of proxy reply payload for one proxy node

Zeus bot uses UDP data request messages to request binary or configuration updates via UDP. The payload of a UDP data request message includes three fields; type, offset , and size. The size of the type field is 1 byte. It is set to 1 for configuration file download and 2 for binary file download. The offset field is 2 bytes, indicating the word from which the responding peer starts transmitting data. The size field is 1360 bytes long, indicating the number of bytes that should be transmitted in response. The data request message via TCP can be identified using the type field in the message header. Type 0x68 is used for binary requests and 0x6A for configuration requests.

UDP data reply messages are sent as a reply to UDP data request messages. The data reply includes a 4 bytes file identifier field and then the data block. An RSA-2048 signature of the MD5 hash of the plaintext data is appended to the transmitted files. The maximum payload size of a UDP data reply message is 1360 bytes.

Data transmission via TCP starts with a message header, where the type field takes 0x64 for a binary update and 0x66 for a configuration update. A little-endian integer with value 1 terminates the data transmission via TCP, where no header is used.

The payload of a peer list request message contains the identifier(20 bytes) of the sending bot followed by eight random bytes. The receiver bot sends a peer list reply message containing ten peers from its peer list. The size of the peer list reply message is 450 bytes. The peer list reply message adds six fields for each returned peer; IP type, peer ID, IPv4 address, IPv4 port, IPv6 address ,and IPv6 port. Fig. 6 depicts the structure of each peer in the peer list reply message payload.

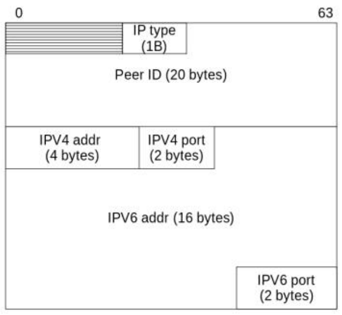


Figure 6. Format of peer-list reply message payload for one peer node.

Fig. 7 depicts the proxy announcement message payload format. This message utilizes the TTL field in the message header. Initially, the value of the TTL field is 4. When a bot node receives a proxy announcement message, it updates its proxy list by adding the new proxy node. Then decrements the value of TTL and sends this message to its neighbor nodes. The message will be broadcasted until the value of the TTL field reaches 0.

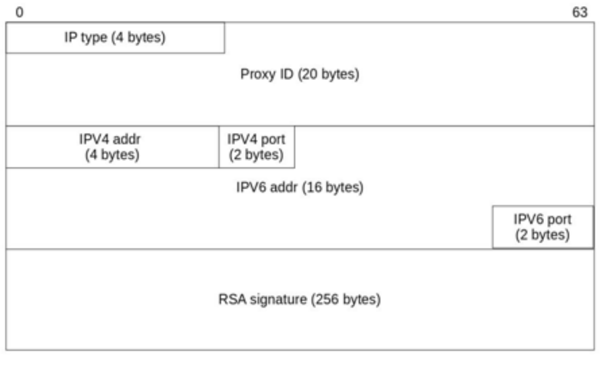


Figure 7. Format of the proxy announcement message payload

Finally, the Zeus bot nodes use another type of message called the C2 message. This message is exchanged over TCP and wraps HTTP messages. This HTTP-based message is used to instruct the bot nodes to perform various malicious activities such as executing a file at a URL, stealing crypto certificates, stealing cookies ,and performing a DDoS attack at a given URL. The C2 message is also used for transferring the stolen data from the bot nodes to the upper layer nodes. Since the C2 message contains an HTTP message, the authors of the technical reports suspect that the communication between the proxy bots and the C2 layer is HTTP based.

4.1.2.2. Generation of Zeus Botnet Traffic in the Testbed

We use a system with an 8th-generation Intel i7 processor and 32 GB RAM for the testbed. Debian Linux (buster) operating system is installed in the system. We use the QEMU virtualization to create 101 virtual hosts. The Debian Linux (stretch) operating system has been installed in the virtual hosts to reduce memory footprint. The network setup is done to interconnect the virtual hosts. A copy of the developed bot has been run in each virtual host. These bots create a P2P network among themselves and communicate periodically (30 minutes) with every 15 others, as is done by the Zeus botnet. A random bot node is selected as a proxy bot node. This proxy bot node collects information from the bot nodes and sends them to the upper layer. Furthermore, this proxy bot node distributes commands or any other information gotten from the upper layer to the bot nodes. In the testbed, the bots steal terminal history information and send it periodically to the proxy bot nodes.

The experiment was run for 24 hours to get a clear communication structure of the bot nodes. We captured the header of all the packets passing through the router using the tcpdump tool. Then we converted the traffic into the NetFlow data using the probe tool. This NetFlow data contains flow information of the bot nodes. Table 1 shows the flow information of the Zeus botnet captured in the testbed. Out of all the flow fields, only the source IP addresses and the destination IP addresses are retained for further analysis.

Table 1. Flow information of Zeus botnet captured in the testbed

Sl. no.	Flow information	Description
1	IPV4_SRC_ ADD	Source IP address(IPV4)
2	IPV4_DST_ ADDR	Destination IP address(IPV4)
3	IN_PKTS	Number of input packets
4	IN_BYTES	Size of input packets in bytes
5	FIRST_SWIT CHED	System uptime at which the first packet of this flow was switched
6	LAST_SWIT CHED	System uptime at which the last packet of this flow was switched
7	SRC_TOS	Type of Service byte setting when entering incoming interface

These flow data contain bot traffic only. To get a realistic dataset, we mix this data with the flow data captured from a real network. For that, we use the CAIDA OC48 Peering Point Traces dataset (2002-2003) which is publicly available. Their newer dataset is no longer publicly available. This publicly available dataset was collected in 2002 and 2003. It contains anonymized passive traffic traces captured at a large ISP's west coast OC48 peering link. The data in the CAIDA dataset are converted to network flow before mixing. As with the testbed data, a list of source and destination IP-address pairs are extracted from these network flows. We mix these two types of traffic flow as follows:

Each bot IP address is merged with an IP address of the CAIDA dataset chosen randomly. The corresponding flows are also merged. One IP address represents both these IP addresses in the final dataset. The details of the final dataset are depicted in table 2.

Table 2. Details of the testbed dataset

Number of nodes	Number of bot nodes	Number of botnet flows	Number of non-bot nodes	Number of non-botnet flows
194044	101	1001986	193943	5954245

The dataset contains no null values as it was generated in a testbed. Our goal is to locate an entire botnet. However, some of the data may be missing as botnets can potentially span the whole Internet, and all such data cannot be captured in reality. To check how our method performs under missing data, we experimented by dropping communications from some bots from our dataset. The result seems satisfying and is presented in figure 12.

The dataset's ratio of bot traffic to regular traffic in the real-worldis significantly low. It is kept low in our dataset as well, to make it imbalanced. To deal with the problem, we reported precision and recall as the measure which represents performance well in an imbalanced dataset.

4.2. Details of CTU-13 Dataset

To evaluate the performance of the proposed method, we have also used the CTU-13dataset[41]. The dataset has 13 scenarios, and each scenario has different malware traffic. Since only scenario 12 contains P2P botnet traffic, we have used this scenario to evaluate the proposed method. A brief description of scenario 12 of the CTU-13 dataset is given in table 3.

Table 3. Details of scenario 12 of CTU-13 dataset

Bot	Number of Bot	Characteristics of botnets	Duration (hrs)	Number of non-bot nodes	Number of flows
NSIS.ay	3	P2P	1.21	94393	325471

4.3. Results

We use the following performance metrics to evaluate the performance of our method

$$ACCURACY = (TP+TN)/(TP+TN+FP+FN) \quad (2)$$
$$PRECISION = TP/(TP+FP) \quad (3)$$
$$RECALL = TP/(TP+FN) \quad (4)$$
$$FPR = FP/(FP+TN) \quad (5)$$

Where TP stands for True Positive and represents the number of bot nodes that were correctly identified, TN for True Negative and represents the number of non-bot nodes that were correctly identified, FP for False Positive and represents the number of non-bot nodes that were incorrectly identified, FN for False Negative and represents the number of bot nodes that were incorrectly identified. FPR is the False Positive Rate.

Our three approaches detect bot nodes separately. The final results are obtained by performing an ensemble operation on the results provided by the three approaches. The individual results of all approaches are shown in table 4, and the final results are shown in table 5 in both datasets.

Table 4. Results of approach, I, II and III in both the datasets

Dataset	Methods	Accuracy	Precision	Recall	FPR
Testbed dataset	Approach I	0.9999	0.9417	0.9604	0.00003
	Approach II	0.9999	0.8475	0.9901	0.00009
	Approach III	0.9999	0.8065	0.9901	0.0001
CTU-13 dataset	Approach I	0.9991	0.0333	1	0.0009
	Approach II	0.9991	0.0327	1	0.0009
	Approach III	0.9989	0.0306	1	0.0011

Table 5. Final results of the proposed method

Dataset	Accuracy	Precision	Recall	FPR
Testbed dataset	0.9999	0.9429	0.9802	0.0001
CTU-13 dataset	0.9991	0.0333	1	0.0009

Table 4 shows that approaches I, II, and III deliver a 99.99% accuracy in the testbed dataset. Similarly, approaches II and III show 99.01% recall in the testbed dataset. However, the FPR of approach III is higher than approaches I and II. Therefore, the precision of approach III is 80.65% which is lower than approaches I and II. The best precision is shown by approach I, which is 94.17% in the testbed dataset.

In the CTU-13 dataset, approaches I, II , and III detect all the bot nodes. Therefore they show 100% recall. Since the dataset contains three bot nodes, only one false positive node also adversely affects the precision result. Therefore, the precision of all the approaches in the CTU-13 dataset is significantly low. However, the FPR is significantly low in all the approaches in the CTU-13 dataset. The accuracy is above 99.8% in all the approaches.

After performing ensemble operations on the independent results of approach I, II ,and III, we get the final result of the proposed method. As shown in table 5, the proposed method delivers more than 99.9% accuracy in both datasets. In the testbed dataset, the proposed method shows 98.02% recall, 94.29% precision ,and less than 0.01% FPR result. On the other hand, the CTU-13 dataset contains three bot nodes. So, only one false positive node also adversely affects the precision result. Therefore, we get a 3.33% precision to result in the CTU-13 dataset. However, the method delivers 100% recall and 0.09% FPR in the CTU-13 dataset.

5. Comparison with Previous Design

In this section, we compare our method's performance to the methods discussed in the related work. The methods we have compared use datasets that require more than the source and destination IP addresses. Two of the three methods[5, 17] make use of packet header fields. This necessitates the analysis of a large volume of data. As a result, these methods will have a significant processing overhead. They will be ineffective in situations where a large number of bots is to be detected, such as in our testbed dataset. The other method[20] uses fields from the flow record, which also increases the volume of data to be analyzed. Moreover, they assume certain traffic patterns. They made use of the upstream and downstream packet ratios. In a real-world scenario, the majority of traffic in a host is normal traffic. Due to the low volume of bot traffic, such bot traffic is unlikely to change the upstream and downstream packet ratio and is likely to be lower than that caused by the randomness of communicated packets. As a result, this method is not expected to produce better results in a realistic dataset.

As our dataset contains only source IP and destination IP addresses, the above methods, which require additional data cannot be compared quantitatively using our dataset.

The comparison results are shown in Table 6. Method 2 uses their private testbed dataset. The method in sl. no. 1 gets more than 99% accuracy. However, other metrics such as precision, recall, TPR, and FPR are not shown. The method in sl. no. 2 and 3 get 100% TPR. However, they do not show other performance metrics like accuracy, precision, or recall. The FPR of the method in sl. no. 3 is higher than our method. From the comparison result, it is seen that our method has outperformed all other methods.

Table 6. Comparison result of the proposed method with other methods

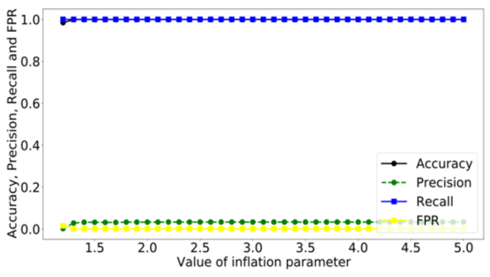
Sl. no.	Papers	Dataset used	Accuracy	Precision	Recall	TPR	TNR	FPR	FNR
1	[5]	Dataset of Peer Rush	>99						
2	[20]	Own testbed				100		0	
3	[17]	ISCX				100		0.706	
4	Our method	CTU-13	99.91	3.33	1	100	99.91	0.09	0
5	Our method	Testbed	99.99	94.29	98.02	98.02	99.99	0.01	1.98

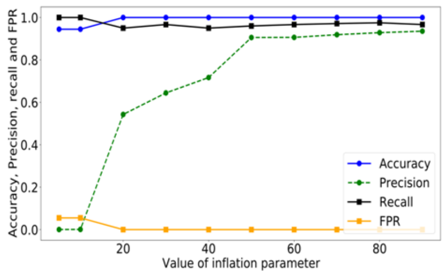
6. Tuning of Parameters in the Proposed Method

6.1. In Approach I

The inflation parameter, which regulates the level of granularity of the output clusters, is the parameter that needs to be adjusted in approach I. It impacts the total number of non-bot nodes in the cluster containing both nodes. We have experimented by changing the inflation parameter values from 1.2 to 5 to observe their sensitivity. Fig. 8(a) shows the results in the CTU-13 dataset and fig. 8(b) shows the results in the testbed dataset.

In the CTU-13 dataset, approach-I delivers consistent results for all the metrics from the inflation parameter 2.2. The accuracy and recall are above 99%, and precision and FPR are less than 5%. The testbed dataset shows fluctuation for the initial values of the inflation parameter. However, the approach shows consistent results from the values 1.5 and above. The accuracy, precision ,and recall are more than 90% from the value of 1.5. Similarly, FPR is less than 1% for the inflation parameter 1.5 and above. We observe acceptable accuracy, precision, recall ,and FPR results from the inflation value of 1.5 and above for both datasets.





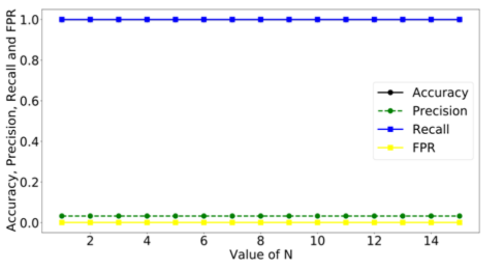
(a) In CTU-13 dataset (b) In testbed dataset

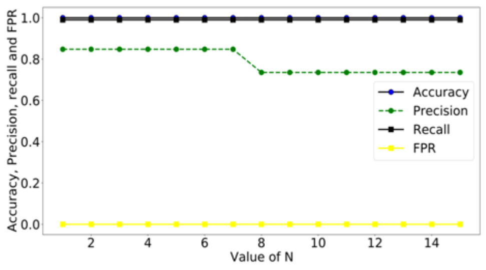
Figure 8. Variation of results in different values of inflation parameter of approach I in both datasets

6.2. In Approach II

In this approach, the parameter N is tuned to determine how many outermost loops should be run before choosing the optimal solution. We change N from 1 to 15 to observe the sensitivity of this parameter. Fig. 9(a) shows the results in the CTU-13 dataset and fig. 9(b) shows the results in the testbed dataset.

The approach delivers high accuracy and recall and low precision and FPR for all the values of N in CTU-13 datasets. We get more than 99% accuracy and precision, less than 1% precision ,and less than 1% FPR for all the values of N. On the other hand, we get more than 99% accuracy and recall for all the values of N in the testbed dataset. FPR is constant and less than 1% for all the values of N. We get more than 80% precision within the range of 1 to 7. However, from the value 8, precision results decrease and less than 60% result is obtained. We observe the best results for both datasets. from 1 to 7.





(a) In CTU-13 dataset (b) In testbed dataset

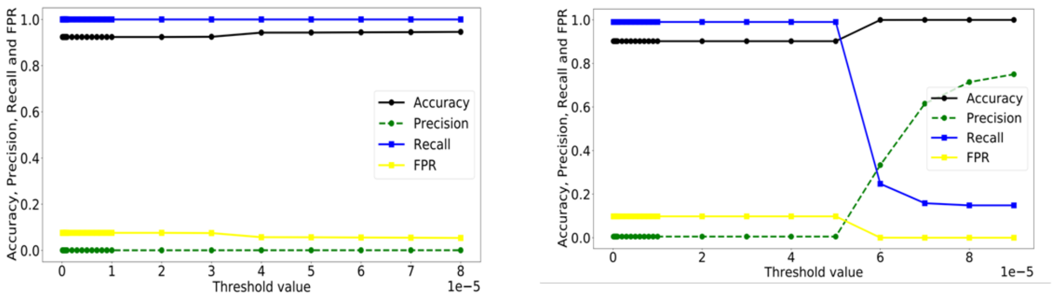
Figure 9. Variation of results in different values of N of approach II in both dataset

6.3. In Approach III

This method uses the PageRank and the HDBSCAN clustering algorithm to detect the bot nodes. We tune the threshold values for the PageRank algorithm to get the optimum result. Here we change the threshold values from 0.00009 to 0.0000001.

In fig. 10(a), we see the results of accuracy, precision, recall ,and FPR implemented in the CTU-13 dataset. The fig. shows that all the performance metrics deliver the same results with minimal fluctuation in all the threshold values. Accuracy and recall are above 90%, and precision and FPR are less than 1% for all the values. We observe the best results in both datasets in the range of

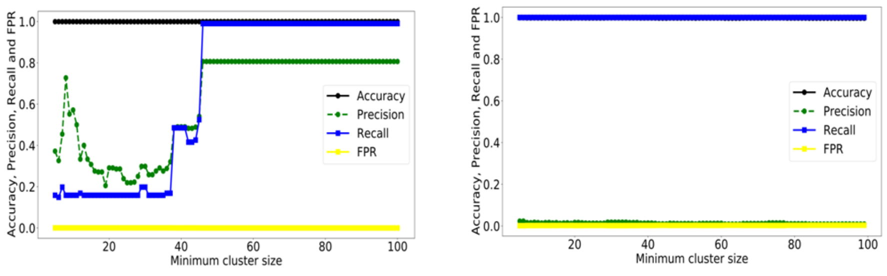
0.0000001 to 0.00004. Fig.s 10(b) shows the result of the PageRank algorithm implemented in the testbed dataset. The accuracy and recall are constant and high for the threshold values from 0.00000010 to 0.00004. After that, recall decreases to 20% ,and accuracy increases to 99%. FPR is consistent and less than 1% for all the values.



(a) In CTU-13 dataset (b) In testbed dataset

Figure 10. Results of threshold value variations of PageRank algorithm in both dataset

In the HDBSCAN clustering algorithm, we can tune the min_clus_size parameter. This parameter specifies the minimum number of nodes in a cluster. We have tuned this parameter from 5 to 100 and have got the following results for both datasets.



(a) In testbed dataset (b) In CTU-13 dataset

Figure 11. Parameter tuning in HDBScan clustering algorithm in both dataset

In the CTU-13 dataset, accuracy and recall are constantly high for all the values of the min_clus_size parameter. On the other hand, precision and the FPR show significantly low results for all the values. In the testbed dataset, the precision and recall show high fluctuation in the initial values of the min_clus_size parameter. However, from the parameter values 42 and above, we get consistently good results for all the matrices. On the other hand, we got more than 99% results for accuracy in all the parameter values in the testbed dataset. Finally, we get less than 1% FPR for all the values of min_clus_size in the testbed dataset.

7. ROBUSTNESS OF OUR PROPOSED METHOD

To investigate the robustness of the proposed method in datasets with more than three bots, such as the CTU-13 dataset, we run an experiment in which we vary the number of bots from 5 to 101 while keeping the total number of nodes constant across the dataset, i.e. 194044. We performed this experiment in our testbed dataset because there is no other publicly available dataset with more than three bot nodes other than CTU-13. Figure 12 depicts the outcomes of our method across all datasets. According to the results, this method performs reasonably well in terms of accuracy, precision, recall and FPR, with this varying mixture of the bot and non-bot traffic.

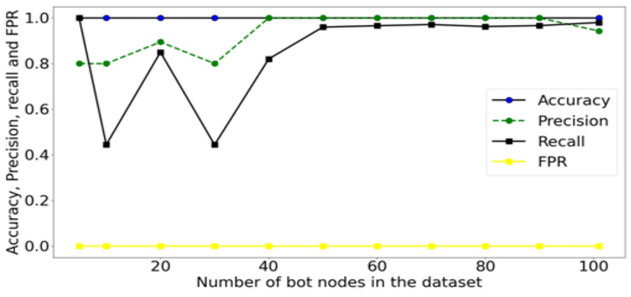


Figure 12. Results of our method in different datasets containing bot nodes from 5 to 101 bot nodes.

8. CONCLUSION

We have developed a method to detect P2P botnets. We have used three different approaches to detect bot nodes individually. Then an ensemble operation is performed on the results provided by the three approaches to get the final result. To test the performance of the proposed method, we have set up a testbed and generated a synthetic dataset. We also use a publicly available dataset, the CTU-13 dataset, for comparison. Experimental results show an accuracy of 99.99%, a precision of 94.3%, a recall of 98.02% ,and an FPR of 0.01% on the testbed dataset. We also obtain 99.91% accuracy and 100% recall using the CTU-13 dataset. The results are better than the existing works.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] Bailey, M., Cooke, E., Jahanian, F., Xu, Y., & Karir, M. (2009, March). A survey of botnet technology and defenses. In 2009 Cybersecurity Applications & Technology Conference for Homeland Security (pp. 299-304). IEEE.

[2] Zareh, A., & Shahriari, H. R. (2018, August). Botcointrap: detection of bitcoin miner botnet using host based approach. In 2018 15th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC) (pp. 1-6). IEEE.

[3] The ugly truth about bots in 2019. Resource Library. (2019, July 1). Retrieved September 16, 2022, from https://www.imperva.com/resources/resource-library/infographics/the-ugly-truth-about-bots-in-2019/

[4] Feng, L., Liao, X., Han, Q., & Song, L. (2012). Modeling and analysis of peer-to-peer botnets. Discrete Dynamics in Nature and Society, 2012.

[5] Dhayal, H., & Kumar, J. (2017). Peer-to-Peer Botnet Detection based on Bot Behaviour. International Journal of Advanced Research in Computer Science, 8(3).

[6] Cooke, E., Jahanian, F., & McPherson, D. (2005). The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. SRUTI, 5, 6-6.

[7] Resende, P. A. A., & Drummond, A. C. (2018). HTTP and contact-based features for Botnet detection. Security and Privacy, 1(5), e41.

[8] Tarng, W., Den, L. Z., Ou, K. L., & Chen, M. (2011). The analysis and identification of P2P botnet's traffic flows. International Journal of Communication Networks and Information Security, 3(2), 138.

[9] Stevanovic, M., & Pedersen, J. M. (2014, February). An efficient flow-based botnet detection using supervised machine learning. In 2014 international conference on computing, networking and communications (ICNC) (pp. 797-801). IEEE.

[10] Haq, S., & Singh, Y. (2018, December). Botnet detection using machine learning. In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) (pp. 240-245). IEEE.

[11] Ibrahim, W. N. H., Anuar, S., Selamat, A., Krejcar, O., Crespo, R. G., Herrera-Viedma, E., & Fujita, H. (2021). Multilayer framework for botnet detection using machine learning algorithms. IEEE Access, 9, 48753-48768.

[12] Muhammad, A., Asad, M., & Javed, A. R. (2020, October). Robust early stage botnet detection using machine learning. In 2020 International Conference on Cyber Warfare and Security (ICCWS) (pp. 1-6). IEEE.

[13] Khan, R. U., Zhang, X., Kumar, R., Sharif, A., Golilarz, N. A., & Alazab, M. (2019). An adaptive multi-layer botnet detection technique using machine learning classifiers. Applied Sciences, 9(11), 2375.

[14] Wang, J., & Paschalidis, I. C. (2014, September). Botnet detection using social graph analysis. In 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton) (pp. 393-400). IEEE.

[15] Joshi, H. P., Bennison, M., & Dutta, R. (2017, September). Collaborative botnet detection with partial communication graph information. In 2017 IEEE 38th Sarnoff Symposium (pp. 1-6). IEEE.

[16] Alharbi, A., & Alsubhi, K. (2021). Botnet detection approach using graph-based machine learning. IEEE Access, 9, 99166-99180.

[17] Liang, J., Zhao, S., & Chen, S. (2022). A Protocol-Independent Botnet Detection Method Using Flow Similarity. Security and Communication Networks, 2022.

[18] Sriram, S., Vinayakumar, R., Alazab, M., & Soman, K. P. (2020, July). Network flow based IoT botnet attack detection using deep learning. In IEEE INFOCOM 2020-IEEE conference on computer communications workshops (INFOCOM WKSHPS) (pp. 189-194). IEEE.

[19] Qiu, Z., Miller, D. J., & Kesidis, G. (2017, March). Flow based botnet detection through semi-supervised active learning. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2387-2391). IEEE.

[20] Dillon, C. (2014). Peer-to-Peer botnet detection using NetFlow.

[21] Yahyazadeh, M., & Abadi, M. (2012). BotOnus: An online unsupervised method for botnet detection. The ISC International Journal of Information Security, 4(1), 51-62.

[22] Nõmm, S., & Bahşi, H. (2018, December). Unsupervised anomaly based botnet detection in IoT networks. In 2018 17th IEEE international conference on machine learning and applications (ICMLA) (pp. 1048-1053). IEEE.

[23] Arshad, S., Abbaspour, M., Kharrazi, M., & Sanatkar, H. (2011, December). An anomaly-based botnet detection approach for identifying stealthy botnets. In 2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE) (pp. 564-569). IEEE.

[24] Pektaş, A., & Acarman, T. (2019). Deep learning to detect botnet via network flow summaries. Neural Computing and Applications, 31(11), 8021-8033.

[25] Shi, W. C., & Sun, H. M. (2020). DeepBot: a time-based botnet detection with deep learning. Soft Computing, 24(21), 16605-16616.

[26] Kabla, A. H. H., Thamrin, A. H., Anbar, M., Manickam, S., & Karuppayah, S. (2022). PeerAmbush: Multi-Layer Perceptron to Detect Peer-to-Peer Botnet. Symmetry, 14(12), 2483. MDPI AG. Retrieved from http://dx.doi.org/10.3390/sym14122483.

[27] Van Dongen, S. M. (2000). Graph clustering by flow simulation (Doctoral dissertation).

[28] Dongen, Stijn van. “MCL - a Cluster Algorithm for Graphs.” MCL - a Cluster Algorithm for Graphs, micans.org/mcl. Accessed 16 Sept. 2022.

[29] Edler, D., Bohlin, L., & Rosvall, M. (2017). Mapping higher-order network flows in memory and multilayer networks with infomap. Algorithms, 10(4), 112.

[30] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Stanford InfoLab.

[31] Campello, R. J., Moulavi, D., & Sander, J. (2013, April). Density-based clustering based on hierarchical density estimates. In Pacific-Asia conference on knowledge discovery and data mining (pp. 160-172). Springer, Berlin, Heidelberg.

[32] Bustamam, A., Wisnubroto, M. S., & Lestari, D. (2018, October). Analysis of protein-protein interaction network using Markov clustering with pigeon-inspired optimization algorithm in HIV (human immunodeficiency virus). In AIP Conference Proceedings (Vol. 2023, No. 1, p. 020229). AIP Publishing LLC.

[33] Lestari, D., Raharjo, D., Bustamam, A., Abdillah, B., & Widhianto, W. (2017, July). Application of clustering methods: Regularized Markov clustering (R-MCL) for analyzing dengue virus similarity. In AIP Conference Proceedings (Vol. 1862, No. 1, p. 030130). AIP Publishing LLC.
[34] Permata, T. S., & Bustamam, A. (2015, October). Clustering protein-protein interaction network of TP53 tumor suppressor protein using Markov clustering algorithm. In 2015 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 221-226). IEEE.
[35] Alzahrani, T., & Horadam, K. J. (2016). Community detection in bipartite networks: Algorithms and case studies. In Complex systems and networks (pp. 25-50). Springer, Berlin, Heidelberg.
[36] Zeng, J., & Yu, H. (2018, August). A distributed infomap algorithm for scalable and high-quality community detection. In Proceedings of the 47th International Conference on Parallel Processing (pp. 1-11).
[37] Linhares, C. D., Ponciano, J. R., Pereira, F. S., Rocha, L. E., Paiva, J. G. S., & Travençolo, B. A. (2020). Visual analysis for evaluation of community detection algorithms. Multimedia Tools and Applications, 79(25), 17645-17667.
[38] Rosvall, M., Axelsson, D., & Bergstrom, C. T. (2009). The map equation. The European Physical Journal Special Topics, 178(1), 13-23.
[39] Orrite, C., Rodríguez, M., Martínez, F., & Fairhurst, M. (2008, September). Classifier ensemble generation for the majority vote rule. In Iberoamerican congress on pattern recognition (pp. 340-347). Springer, Berlin, Heidelberg.
[40] Rahman, A. F. R., Alam, H., & Fairhurst, M. C. (2002, August). Multiple classifier combination for character recognition: Revisiting the majority voting system and its variations. In International Workshop on Document Analysis Systems (pp. 167-178). Springer, Berlin, Heidelberg.
[41] Garcia, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. computers & security, 45, 100-123.
[42] https://www.cert.pl. 2022. ZeuS-P2P monitoring and analysis. [online] Available at: <https://www.cert.pl/wp-content/uploads/2015/12/2013-06-p2p-rap en.pdf> [Accessed 16 September 2022].
[43] Andriesse, D., & Bos, H. (2014). An analysis of the zeus peer-to-peer protocol. Technical Report IR-CS-74.
[44] Son, Nguyen Hong, and Ha Thanh Dung. "A Lightweight Method for Detecting Cyber Attacks in High-Traffic Large Networks Based on Clustering Techniques." International Journal of Computer Networks & Communications, vol. 15, no. 01, 30 Jan. 2023, pp. 35–51, https://doi.org/10.5121/ijcnc.2023.15103.

AUTHORS

Dhruba Jyoti Borah: Dhruba Jyoti Borah is currently a research scholar in the Department of Computer science, Gauhati University, India. He graduated from Dibrugarh University, India with an MCA (Master of Computer Application) degree. His research interest includes network security, botnet detection. Email: dhrubajyotiborah209@gmail.com

Abhijit Sarma: Abhijit Sarma had retired as an associate professor from the Department of Computer Science at Gauhati University, India. He acquired his PhD degree in the area of wireless networks from the Department of Computer Science and Engineering, Indian Institute of Technology, Guwahati (IITG) in 2014. He had presented his research findings in various peer-reviewed journals and international conferences. His research interest includes network security, wireless LAN and heterogeneous networks.Email: abhijit_gu@yahoo.com

APPENDIX A

Supplementary information

Testbed dataset: https://doi.org/10.6084/m9.figshare.20832001.v1