1-1-2018

Botnet Detection Using Graph Based Feature Clustering

Ravi Kiran Akula

Botnet detection using graph based feature clustering

By

Ravi Kiran Akula

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Industrial and Systems Engineering
in the Department of Industrial and Systems Engineering

Mississippi State, Mississippi

May 2018

Botnet detection using graph based feature clustering

By

Ravi Kiran Akula

Approved:

_____
Linkan Bian
(Major Professor)

_____
Mohammad Marufuzzaman
(Committee Member)

_____
Hugh R. Medal
(Committee Member)

_____
Stanley F. Bullington
(Graduate Coordinator)

_____
Jason M. Keith
Dean
James Worth Bagley College of Engineering

Name: Ravi Kiran Akula

Date of Degree: May 4, 2018

Institution: Mississippi State University

Major Field: Industrial and Systems Engineering

Major Professor: Linkan Bian

Title of Study: Botnet detection using graph based feature clustering

Pages in Study 59

Candidate for Degree of Master of Science

Detecting botnets in a network is crucial because bot-activities impact numerous areas such as security, finance, health care, and law enforcement. Most existing rule and flow-based detection methods may not be capable of detecting bot-activities in an efficient manner. Hence, designing a robust botnet-detection method is of high significance. In this study, we propose a botnet-detection methodology based on graph-based features. Self-Organizing Map is applied to establish the clusters of nodes in the network based on these features. Our method is capable of isolating bots in small clusters while containing most normal nodes in the big-clusters. A filtering procedure is also developed to further enhance the algorithm efficiency by removing inactive nodes from bot detection. The methodology is verified using real-world CTU-13 and ISCX botnet datasets and benchmarked against classification-based detection methods. The results show that our proposed method can efficiently detect the bots despite their varying behaviors.

DEDICATION

I would like to dedicate this research to my Grandparents, Akula Sambaiah garu and Sita Ravamma, and my parents, Mallikarjuna Rao and Dhana Lakshmi, for their relentless support of my pursuance of academic excellence, and my sister and brother, for their continuous encouragement.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Linkan Bian for giving me his valuable guidance, feedback and an opportunity to work under him. I would like to express my deepest gratitude to my committee members for supporting me. I would like to thank Dr. Usher for helping me to find funding initially in the ISE Department. I would like to thank my project team members Mojtaba Khanzadeh (M.K) and Sudipta Chowdhury, who has always helped me throughout this project. I also thank my friends Pavan Yeddanapudi, Sushil Raj Poudel, and all the Indian Student Association (ISA) committee members (2015-2016), who have guided me in settling down in Starkville and giving me moral support in handling any kind of issues when I entered The United States of America. I would like to thank Dr. Ravi Sadasivuni and Dr. Lalitha Dabbiru for being my local guardians and mentoring me throughout these years. I would like to mention a special person, my dear friend, Yuwei Sun (RA), who was always there with me.

Finally, I would like to thank The United States of America and Mississippi State University for giving this wonderful opportunity to me and many International students like me. HailState!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Introduction

During the last 15 years, botnets have caused some of the most devastating and costly internet security incidents in the world [1]. The term "bot" comes from robot which is also sometimes called Zombie. A bot may also be known as a Web robot or WWW robot. It is a type of malware [2] that an attacker can exploit to control an infected computer. It is installed into a compromised computer which can be controlled remotely by an attacker or a group of attackers for fulfilling their own gain. One of the most common methods for a bot program to infect a compromised computer is by a malicious website the user is visiting that silently searches and exploits vulnerability in the user's system in order to install the bot on it. Some other ways to infect include sending the bot as an attached file with spam emails, or as a program dropped from the payload of another malware. After successful installation of bot code into the compromised computer, it becomes part of large network of compromised computers and hence the term “botnet” is used. Attacker can issue commands to a single bot, or to all the bots in botnet. The attacker controlling the botnet is sometimes referred to as the “botherder””botmaster” or “controller” [3]. Figure 1 shows a typical botnet cycle. Contrary to existing malware such as viruses and worms, which focus on attacking the

infecting host, bots can receive commands from botmaster and can also be used in a distributed attack platform [4].

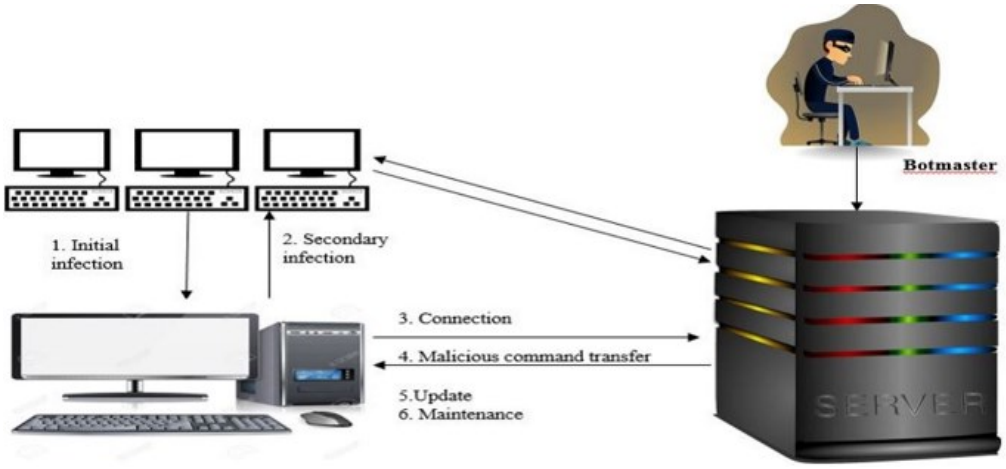

Figure 1.1 Botnet life cycle

Botnets can significantly damage the security of individuals and businesses. They pose a serious and growing threat against cyber-security as they provide a distributed platform for many cyber-crimes such as Distributed Denial of Service (DDoS) attacks against critical targets, malware dissemination, phishing, and click fraud [5, 6]. Even in some cases, botmasters sell access to the botnet to other criminals – either on a rental basis or as an outright sale [7]. As a result, botnet detection has been a major research topic in recent years. Researchers have proposed several detection approaches for botnet detection to combat botnet threat against cyber-security [8]. A majority of the existing Botnet detection approaches concentrate primarily on particular Botnet command and control (C&C) protocols (e.g., HTTP, IRC) and structures (e.g., centralized or P2P). They follow rule based approaches to detect botnets in network. However, these approaches

can become ineffective and obsolete if botnets change their structure and C&C techniques to evade detection [4]. Thus, a robust botnet detection approach that can detect any type of botnet with varying characteristics is of utmost importance. Before exploring existing botnet detection schemes in literature, we first survey some of the studies done in anomaly detection. Later, existing efforts dedicated to bot detection are identified that can be divided into two broad categories: botnet detection using NetFlow based features and graph-based features.

1.2 Anomaly detection techniques

Researchers have conducted extensive research on anomaly detection techniques over the years. For example: Fadlullah et al. [9] develop a novel detection technique called DTRAB to infer DDoS attacks. The authors investigated the detection of attacks against application-level protocols that are encapsulated via encryption. In essence, this detection scheme is a distributed detection mechanism capable of detecting the anomalous events as early as possible. Moreover, DTRAB is able to simultaneously construct a defensive mechanism to discover attacks as well as find out the root of the threat by tracing back the attacker's original network. The effectiveness of this scheme is validated via simulation. Flow correlation information is utilized by Zhang et al. [10, 11, 12] to further improve the classification accuracy considering only a small number of training instances based on K-NN and Naive Bayes classifier that are used to detect anomalies in the network. Yan et al. [13] propose a framework of security and trust for 5G based on the perspective that the next generation network functions will be highly virtualized and software defined networking is applied for traffic control. The proposed approach by the researchers utilizes adaptive trust evaluation and management

technologies as well as sustainable trusted computing technologies to achieve computing platform trust and software defined networking security. A qualitative comparison between the advantages and disadvantages of software defined networking and traditional networking regarding security issues concerning overall architecture and a detailed analysis of the threats of software defined networking from the perspective of functional layers and attack types is provided by Shu et al. [14].

1.3 Flow-Based Methods

The botnet detection literature using NetFlow based features is a rich one and many researchers have significantly contributed in this area (e.g. [15-17]). Most of the existing detection schemes falls into either of the two types of methods: clustering and classification ([24, 27, 29]), and others.

Clustering is a popular approach taken by researchers to detect botnets using flow based features. Zeidanloo et al. have proposed a botnet detection framework that can detect botnets without prior knowledge of them [19]. This detection framework is based on finding similar communication patterns and behaviors among the group of hosts that are performing at least one malicious activity using X-means clustering. Using Audit Record Generation and Utilization System (ARGUS) [20], the authors have collected flow based information such as source IP address, destination IP address, source Port, destination Port, duration, protocol, number of packets, and number of bytes transferred in both directions, which are later used to detect the group of hosts that exhibit similar behavior and communication pattern. Karasaridis et al. have developed a K-mean based method that employs scalable non-intrusive algorithms that analyze vast amounts of summary traffic data [22]. Gu et al. have proposed a novel anomaly-based botnet

detection system that is independent of the protocol and structure used by botnets [26]. This detection system has exploited the essential definition and properties of botnets, i.e., bots within the same botnet exhibit similar C&C communication patterns and similar malicious activities patterns. It utilizes a number of flow based information such as time, source IP, destination IP, source port, destination port, duration, and the number of packets and bytes transferred in both directions. C-plane clustering method is used to read the communication logs generated by C plane monitor and find clusters that share similar communication pattern. Arshad et al. have developed an anomaly-based method that require not a priori knowledge of bot signatures, botnet C&C protocols, and the C&C server addresses [28]. Flow characteristics such as IP, port, packet event times, and bytes per packet are examined by Amini et al. to detect botnets where these NetFlow data is collected, filtered, and is finally clustered using hierarchical clustering [25]. Rule based methods are then applied to refine the clusters to reduce the percentage of false positives.

Among the authors' who use classification techniques, Strayer et al. have developed detection approaches by examining flow characteristics such as bandwidth, packet timing, and burst duration, where they first eliminate traffic that is unlikely to be a part of a botnet, classify the remaining traffic into a group that is likely to be part of a botnet by using J48 decision trees, naïve Bayes, and Bayesian classifier, and finally correlate the likely traffic to find common communications patterns that would suggest the activity of a botnet [24,29]. Fairly recently, a decision tree classifier has been used by Zhao et al. to detect botnets by investigating 12 flow based features [27]. Their proposed method can detect botnets during the C&C and attack phases based on the observation of network flow characteristics for specific time intervals. It does not require significant

malicious activity to occur before detection as it can recognize command and control signals. Simultaneously, it does not require the group behavior of several bots before it can be confident about making a decision.

Lu et al. have incorporated both classification and clustering techniques in detection of botnets, developing an unsupervised botnet detection framework where they first identify network traffic from existing known applications, then focus on each application community that might include botnet communication flows [30]. This network traffic is then clustered to find the anomalous behaviors on that specific application community based on the n-gram features extracted from the content of network flows. The proposed detection framework has been evaluated on an IRC community and results show that this approach obtains a high detection rate with a very low false alarm rate when detecting IRC botnet traffic.

Apart from classification and clustering techniques, there are a number of other studies that employ other approaches in botnet detection using NetFlow based features. Interested readers can refer to [18, 21, 23, 31, and 32] for such related works.

Limitation: Existing methods of botnet Detection based on NetFlow traffic features rely on computing statistical features of flow traffic or on deep packet inspection. As a result, these methods only capture the characteristics of bots effects on individual links, rather than on the topological structure of a neighborhood/subgraph as a whole. In particular, flow-based detection methods require the comparison of each traffic flow to all the others in order to determine malicious traffic, instead of monitoring the network behaviors in a holistic manner. Such techniques are also deficient in that attackers can evade detection by the use of encrypting commands or changes in data volume or change

in some other behavioral characteristics such as by the use of variable length encryption or changes in packet structure that leads to new behavioral characteristics. To overcome this deficiency, another stream of research has focused on detecting botnets based on graph-based features. This approach is fundamentally more efficient than flow based approaches since it avoids the need to cross compare flows across the dataset [33].

1.4 Graph-Based Methods

There are a number of studies that use different graph-based features to detect anomalies. Literature in this domain can be broadly categorized into two groups: one group detects anomalies in static graphs using graph-based features whereas another group does the same, but with dynamic graphs. The static graphs can be further categorized into plain graphs and attributed graphs. Among the studies that use plain graphs for anomaly detection, Ding et al. [34], Henderson et al. [35], Henderson et al. [36], Kang et al. [37], Aggarwal [38], Zimek et al. [39], Chen and Giles [40] and many more utilize structure-based patterns to detect anomalies. On the other hand, studies done by Sun et al. [41], Tong and Lin [42], Ambai et al. [43], Nikulin and Huang [44] focus on the utilization of community based patterns to detect anomalies. Similarly, for attributed graphs Davis et al. [45], Eberle and Holder [46], and Kontkanen and Myllymki [47] use structure based patterns whereas Gao et al. [48], Muller et al. [49], Perozzi et al. [50] use community based patterns to detect anomalies. With dynamic graphs, authors have used the notion of graph similarity based on certain properties such degree distribution, diameter [51-53], by resorting to matrix or tensor decomposition of the time-varying graphs [54-57], or by monitoring graph communities over time and reporting events when there is structural or contextual change in any of them [58,59].

Botnet detection studies using graph-based features mainly exploits the spatial relationships in communication traffic [31,60,61]. Collins and Reiter have proposed a method to identify bots by noting that scanning behavior initiated by bot infected hosts would tend to connect different disconnected components of protocol-specific traffic graphs [70]. Wang and Paschalidis [62] use behavioral characteristics of bots to detect botnets. Primarily, the authors have focused on analyzing the social relationships that are modeled as graph of nodes. The authors have considered both social interaction graphs and social correlation graphs and have applied the proposed method to a real-world case study. However, for this detection scheme to be successful bots need to show systematic pattern in behavior that may not be very robust for stealthy botnet. 'Graption' is a graph-based method proposed by Iliofotou et al. that identifies peer-to-peer flows by calculating the in-degree to out degree ratio of hosts in protocol traffic graphs [63]. However, this method can be defeated by protocol randomization. A graph-based detection approach to detect web-account abuse attack has been proposed by Zhao et al. where the correlations among botnet activities are uncovered by constructing large user-user graphs [64]. This approach, termed as 'BotGraph' has two components: aggressive sign-up detection and stealthy bot user connection. The first component ensures that the total number of possible bots are limited whereas second component detects stealthy bot users based on constructing a user-user random undirected graph. Only the edge weight feature has been used to detect bots in the graph. Although, the detection rate is very high, this method's accuracy can be disputed if other types of botnets besides the spamming one need to be detected. Jaikumar and Kak have presented a graph-based framework for isolating botnets in a network [65]. This framework uses temporal co-occurrences in the activity

space to detect botnets. This makes the framework independent of the software architecture of the malware infecting the hosts. The proposed framework has been validated by applying it to a simulated environment. However, this approach falls short if bots don't exhibit temporally co-occurring malicious activities. Nagaraja et al. have proposed a botnet detection technique based on structured graph analysis that localizes botnet members by identifying unique communication patterns arising from the overlay topologies prevalent in command and control structure [66]. However, this approach must be paired with some other malware detection scheme to clearly distinguish botnets from regular flows. Francois et al. have proposed an approach called 'BotTrack' where NetFlow related data is correlated and a host dependency model is leveraged for advanced data mining purposes [67]. They have used the popular linkage analysis algorithm 'PageRank' with an additional clustering process to efficiently detect botnets. However, to validate the proposed method, the researchers have only used 13.7 GB of real world data; also, they have generated the botnet randomly as the dataset was not labeled. Moreover, the authors' have assumed that a certain percentage of bots and their characteristic were known beforehand. So, if an unknown botnet exists in the network, their approach may not give good results. Francois et al. have further extended their work on 'BotTrack' by developing a scalable method called 'BotCloud' for detecting botnets regarding the relationships between hosts [68]. The evaluation of this method has showed a good detection accuracy and a good efficiency based on a Hadoop cluster. But, in this case also, the authors have initially used a botnet free dataset and later randomly have generated botnets in them. Hang et al. have used community detection based clustering to identify long-lived low intensity flows using graph-based features [69].

Limitation: Similar to botnet detection methods using statistical features of flow/packet traffic or in some cases even deep packet inspection, existing graph-based botnet detection methods available in the literature have some major limitations. Many of them apply the botnet detection scheme that operates in a simulated environment (e.g. [65]). Moreover, the detection approach proposed in the literature is mostly rule based, meaning that a predetermined rule needs to be established beforehand to detect botnets from a graph (e.g., [60]). This approach may lead to unwarranted result if bots behave differently from a common norm. Although many of the graph-based detection schemes use filtering to remove bot free data (e.g., [64, 66]) and then apply a detection method, the amount of data that needs to be investigated to detect botnets is relatively large. Simultaneously, if dataset is large, the computational expense is often high for the detection approach; which is a huge disadvantage if faster detection is required [64].

1.5 Significance of Our Approach

An important step towards developing a new graph-based detection approach would be to develop a method that is fast and does not follow any particular rule to detect botnets. Simultaneously, the approach must be validated on a real world dataset with different types of botnets. This detection scheme should also be robust enough so that it can be able to reduce the amount of data that is further investigated to detect any kind of botnet present in the dataset. In this study, we have proposed an approach based on graph-based features that can fulfil these requirements. Our main contribution can be summarized as:

- We present a novel graph-based method for the detection of botnets in a computer network.

- Our approach does not depend on any rules to detect botnets and is capable of capturing the changing behavior of bots.
- Seven graph-based features are used in this study to detect botnets.
- The proposed method can detect different types of botnets with different types of behavioral characteristics.
- A real world dataset is used to validate the results.

However, handling real world big data consisting of botnets is challenging.

The rest of the paper is organized as follows. Chapter 2 provides a brief description of the real world dataset used in this study. Chapter 3 discusses in detail the seven features used to detect botnets and the clustering methodology implemented to cluster these features. Chapter 4 provides numerical results obtained after applying clustering methodology to the real world dataset as well as giving a comparative overview of applying classification techniques. Chapter 5 concludes our work and reviews our main contribution to the existing literature.

CHAPTER II

DATA DESCRIPTION

2.1 Data Description

Big data has been an area of interest among researchers in recent years. For instance: Tsai et al. [71] have provided a comprehensive review on studies that attempt to develop new schemes capable of handling big data during the input, analysis, and output stages of knowledge discovery. They have found that majority of the existing literature is focused on innovative methods for data mining and analysis. However, little to no attention have been given to the pre- and post-analysis processing methods. Evolution based algorithms such as accelerated particle swam optimization is used to reduce the dimensionality of big data by Fong et al. [72]. Authors have investigated the applicability their method on exceptionally large volume of data with high degree dimensions and have found that the proposed method results in enhanced analytical accuracy within reasonable processing time. In this study, big data consisting botnet is used for validating the proposed detection methodology. In this study we use the CTU-13 dataset which is one of the biggest labelled datasets available that consists of botnet traffic as well as normal and background labeled data. It was captured at Czech Technological University in 2011. The developers of the dataset have originally developed it to compare three detection methods, namely Cooperative Adaptive Mechanism for Network Protection (CAMNEP) method, BCIus detetection method, and BotHunter method [73]. Researchers

have found that BCIus and CAMNEP detection methods cannot be generalized for all types of botnet behavior. Each of them seems fit for different types of behavior. Analysis of BotHunter detection method shows that in real environments it can still be useful to have blacklists of known malicious IP addresses known beforehand.

After the development of CTU-13 dataset, it has been used by Grill et al. [74] to evaluate the effects of Local Adaptive Multivariate Smoothing (LAMS) model on the NetFlow anomaly detection engine. The proposed method is able to reduce false alarm rate of anomaly detection based intrusion detection systems. Fairly recently, Chanthakoummane et al. [75] have utilized five scenarios of the CTU-13 dataset to evaluate the Snort-IDS rules detection botnets and analyze the function of the botnets in three rules packet such as botnet-cnc.rules, blacklist.rules, and spyware-put.rules. Experimental results show that botnet-cnc.rules can detect botnets for 29798 alerts. Blacklist.rules can detect botnets for up to 44 alerts. Spyware-put.rules cannot detect any botnet. The researchers eventually surmise that botnet-cnc.rules are most proficient in detecting botnets.

Although, researchers are excited about the potential of using CTU-13 datasets in detecting botnets, (e.g., see Malowidzki et al. [76], Chanthakoumman et al. [75]) according to best of this author's knowledge, no significant work has been done using CTU-13 data in the detection of botnets. CTU 13 dataset consists of 13 captures (called scenarios) of different botnet samples [61]. This dataset was designed with goals such as

- Dataset must have real botnet attacks, not simulated attacks
- Must have real world traffic

- Must have ground truth labels for training and evaluating methods discussed in [73]
- Must include multiple types of botnets.
- Must have several bots infected simultaneously to capture synchronization patterns.
- Must have NetFlow files to protect the privacy of the users.

A scenario in CTU-13 can be defined as a particular infection of the virtual machines using a specific malware. Data collection period for each scenario is significantly different from one another. The duration of recorded NetFlow data vary from 0.26 hours to 66.85 hours and subsequently the amount of NetFlow data also varies accordingly. Multiple types of bots are found in the scenarios. Majority of the scenarios have only one bots (scenario 1-8 and 13), whereas few (scenario 9-12) have multiple bots in them. Percentage of botnet flow is also very negligible (<2%) compared to total NetFlow for majority of the scenarios. However, botnet flow percentage increases (6-8%) when there are multiple bots present in the dataset (except scenario 12). Another distinctive feature of CTU-13 dataset is that, each scenario has been manually analyzed and labeled. The labeling process was performed inside the NetFlow files. Table 2.1 provides a summary of the amount of data on each botnet scenario and percentage of botnet on each scenario.

Table 2.1 Characteristics of Botnet Scenarios [37]

Dataset	IRC	Spam	CF	PS	DDoS	FF	P2P	US	HTTP	Note
1	✓	✓	✓							
2	✓	✓	✓							
3	✓			✓				✓		
4	✓				✓			✓		UDP and ICMP DDoS
5		✓		✓					✓	Scan web proxies
6				✓						Proprietary C&C.RDP
7									✓	Chinese hosts
8				✓						Proprietary C&C.Net BIOS,STUN
9	✓	✓	✓	✓						
10	✓				✓			✓		UDP DDoS
11	✓				✓			✓		ICMP DDoS
12							✓			Synchronization
13		✓		✓					✓	Captcha, Web mail

IRC: Internet relay chat
CF: Click fraud
PS: Port scanned
DDOS: Distributed Denial of service
ICMP: Internet Control Message Protocol
STUN: Simple traversal of UDP through NATs

FF; Fast flux
P2P: Peer to peer
HTTP: Hypertext Transfer Protocol
UDP: User Datagram Protocol
NetBIOS: Network basic input/output System

Another distinctive feature of CTU-13 dataset is that, each scenario has been manually analyzed and labeled. The labeling process was performed inside the NetFlow files. Table 2.2 provides a summary of the amount of data on each botnet scenario and percentage of botnet on each scenario.

Table 2.2 Amount of Data on each Botnet Scenario of CTU-13 dataset.

Dataset	Duration (hrs)	NetFlows	Size(GB)	Bot name	Number of Bots	Botnet flow
1	6.15	2824637	52	Neris	1	39933(1.41%)
2	4.21	1808123	60	Neris	1	18839(1.04%)
3	66.85	4710639	121	Rbot	1	26759(0.56%)
4	4.21	1121077	53	Rbot	1	1719(0.15%)
5	11.63	129833	37.6	Virut	1	695(0.53%)
6	2.18	558920	30	Menti	1	4431(0.79%)
7	0.38	114078	5.8	Sogou	1	37(0.03%)
8	19.5	2954231	123	Murlo	1	5052(0.17%)
9	5.18	2753885	94	Neris	10	179880(6.5%)
10	4.75	1309792	73	Rbot	10	106315(8.11%)
11	0.26	107252	5.2	Rbot	3	8161(7.6%)
12	1.21	325472	8.3	NSIS.ay	3	2143(0.65%)
13	16.36	1925150	34	Virut	1	38791(2.01%)

2.2 ISCX Botnet Dataset

ISCX botnet dataset was developed by Information Security Center of Excellence (ISCX) at the University of New Brunswick (UNB). Researchers at UNB have developed this dataset with the purpose to determine the performance of any intrusion detection approaches or making comparisons which requires experimentation with data that includes real time traffic [92]. ISCX botnet dataset is an evaluation dataset combining non overlapping subsets of three different available datasets: ISOT dataset [21], ISCX 2012 IDS dataset [93], and CTU-13 dataset [37]. In order to produce this synthetic dataset, the researchers have employed an overlay methodology [94] to combine all the three different datasets into one unified dataset which has wide range of bots. Final ISCX botnet dataset was divided into two training and test datasets, where we have selected as a test dataset to implement our methodology. Table 2.3 and 2.4 provide a clear insight to

different Botnet types and portion of flows in the ISCX botnet testing dataset. Access to this dataset is available upon request from the researchers of ISCX UNB.

Table 2.3 Distribution of botnet types in the ISCX botnet test dataset

Botnet name	Type	Flow portions in dataset
Neris	IRC	25967(5.67%)
Rbot	IRC	83(0.018%)
Menti	IRC	2878(0.62%)
Sogou	HTTP	89(0.019%)
Murlo	IRC	4881(1.06%)
Virut	HTTP	58576(12.80%)
NSIS	P2P	757(0.165%)
Zeus	P2P	502(0.109%)
SMTP Spam	P2P	21633(4.2%)
UDP Storm	P2P	44062(9.63%)
Tbot	IRC	1296(0.283%)
Zero Access	P2P	1011(0.221%)
Weasel	P2P	42313(9.25%)
Smoke Bot	P2P	78(0.017%)
Zeus Control (C&C)	P2P	31(0.006%)
ISCX IRC bot	P2P	1816(0.387%)

Table 2.4 List of all malicious in the ISCX botnet test dataset

192.168.2.112	131.202.243.84	192.168.5.122
198.164.30.2	192.168.2.110	192.168.4.118
192.168.2.113	192.168.1.103	192.168.4.120
192.168.2.112	192.168.2.109	192.168.2.105
147.32.84.180	147.32.84.170	147.32.84.150
147.32.84.140	147.32.84.130	147.32.84.160
10.0.2.15	192.168.106.141	192.168.106.131
172.16.253.130	172.16.253.131	172.16.253.129
172.16.253.240	74.78.117.238	158.65.110.24
192.168.3.35	192.168.3.25	192.168.3.65
172.29.0.116	172.29.0.109	172.16.253.132
192.168.248.165	10.37.130.4	

The proposed approach in this study is first of its kind to convert the NetFlow features available from CTU-13 and ISCX dataset into graph-based features and use these graph features to detect botnets. As CTU-13 dataset is the most complete real world dataset [41], we choose this dataset primarily to prove the concept of our novel approach and as an extension we also use ISCX Botnet test dataset to compare the efficiency of proposed bot detection approach, the details of which are discussed in Chapter 3.

CHAPTER III

METHODOLOGY

This section discusses in detail the seven features used to detect botnets and the clustering methodology implemented to cluster these features. First of all, a directed graph is generated for each of 13 datasets of CTU-13. A directed graph (digraph) can be defined as a set of nodes connected by directed edges where each edge points from first node of a graph pair to the second node of the pair. Mathematically, a directed graph can be expressed as an ordered pair where V is a set of nodes and E is an ordered pair of edges. Note that, in this study each node denotes a unique IP address and each edge denotes the connection between one IP address to another. Subsequently, the feature values are calculated of these directed graphs and afterwards clustering methodology is applied to find out the nodes with similar features.

3.1 Graph-Based Features Selection

We have initially tried different combinations of 11 extracted graph features to check the percentage of nodes to be eliminated when SOM technique is implemented on 13 CTU datasets (see Table 3.1). Among them, finally 7 features are selected. These 7 features include the indegree, outdegree, sum of ingoing edges weight, sum of outgoing edges weight, clustering coefficient, node betweenness centrality, and eigen vector centrality. A set of 9 features include indegree duration and outdegree duration along with the 7 features. Finally, 11 features include ingoing protocol mode and outgoing

protocol mode in addition to the 9 features. Results from the Table 3.1 clearly indicate that percentage of nodes to be eliminated (number of nodes in the largest cluster) in further investigation of remaining nodes for bot detection in each dataset is very high (almost greater than 98%) when 7 features are used.

Table 3.1 Combination of different graph-based features vs % of nodes to eliminated

CTU Dataset	**No. of Nodes**	**% of nodes to be eliminated for bot detection**		
		7 Features	9 Features	11 Features
1	311420	99.4608	59.8099	7.721405176
2	442471	99.5556	67.1314	8.295006904
3	434988	99.7388	91.6264	17.78968615
4	186245	99.3540	75.8372	8.27887997
5	41658	98.1180	83.7774	9.902059628
6	107343	98.4433	67.7836	11.46604809
7	38205	97.5265	80.356	10.04580552
8	383788	99.7086	78.2263	13.34252243
9	367264	99.7296	76.4496	11.14702231
10	197824	99.5814	13.3265	11.59515529
11	41933	96.8282	87.7877	9.155080724
12	94436	98.5895	77.9152	9.551442247
13	315769	99.2263	76.7827	9.36939978

These observations clearly prove that the computational costs and the time that is required to search for the malicious activity will be much lower if the right combination of features are selected. The percentage of nodes for further investigation in identification

of bots is very high for 9 and 11 feature combinations compared to 7 feature combinations. Hence, it can be stated that it is vital to be efficient while extracting graph-based features and implementing bot detection methodology on large datasets. Improper selection of features may result in increased computational cost. This is the main rationale behind to proceed with the 7 features combination to implement further investigations of our study. A brief discussion of these seven features is provided below:

3.1.2 In Degree:

If many suspected bots contact a malicious domain for C&C reasons, this will result in a relatively high in degree for this domain. Keeping this in mind, in degree has been chosen as a feature to detect botnet in a network. For a particular node in a directed graph, in degree can be defined as the total number of head ends adjacent to that node. High value of in degree for a node indicates the neighboring nodes tendency to establish more connection where as low value indicates the opposite. For example: Fig. 3.1 shows that node 'a' has an indegree of two.

3.1.3 Out Degree:

For a particular node in a directed graph, the total number of tail ends adjacent to a node is called the out degree of the node. A high value of out degree for a node implies that this node tends to make more connections with other nodes and low value implies the opposite. Bots tend to make more connection with other potential victim computers to spread the reach of botnet or to C&C domain for transferring information. So, out degree can be a useful indicator of botnet activity in a graph. As evident from Fig. 3.1, we can see that, node 'a' has an outdegree of two.

3.1.4 In Degree Weight:

In degree weight refers to the total number of data packets received by a particular node transferred from its neighboring connected nodes. The mechanics of transferring data packets consists of setting up the data connection to the appropriate ports and choosing the parameters for transfer. Besides the raw data every data packet contains, it also has headers that carry certain types of metadata, along with the routing information and trailers that help in refining data transmission [77]. Botnets tend to communicate with each other or to the C&C server to transfer information or update their commands. The same type of botnets usually communicate periodically and with a predefined set of commands. We assume that bots will receive the same type of command and receive approximately same volume of information that can be used to differentiate between bots and non-bots.

3.1.5 Out Degree Weight:

Out degree weight is the opposite of in degree weight which can be described as the total number of data packets sent by a particular node to its neighboring connected nodes. Same as in degree weight, we assume that bots will have similarity in the volume of data it sends out to other IP addresses in the network and can be a useful indicator of botnet activity in a network.

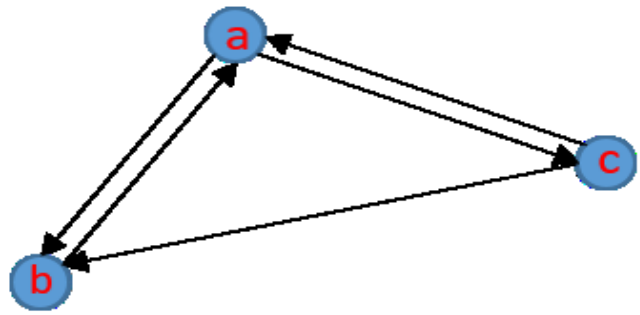

Figure 3.1 A directed graph with three nodes

3.1.6 Node Betweenness Centrality:

In graph theory, node betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. More specifically, node betweenness centrality indicates a particular node's centrality in graph, which refers to how many shortest paths from all nodes to all others pass through that particular node [78]. Node betweenness centrality can be mathematically expressed as [79]:

$$N_B(v) = \sum_{u \neq v \neq w} \frac{\sigma_{uw}(v)}{\sigma_{uw}} \tag{3.1}$$

Where σ_{uw} is the total number of shortest paths from node **'u'** to **'w'** and $\sigma_{eb}(v)$ is total number of shortest paths that pass-through node **'v'.** Figure 3.2 illustrates the concept of node betweenness centrality [80] by calculating for node a.

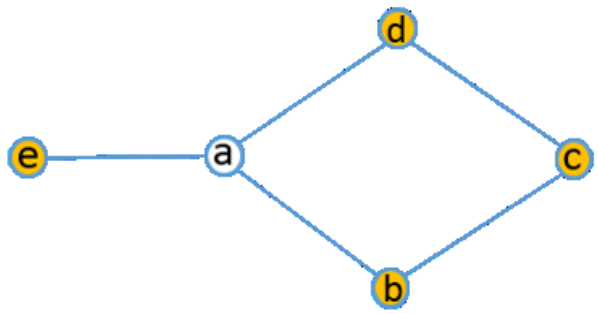

Figure 3.2 Node betweenness centrality

When calculating betweenness centrality for node a, total number of paths can be formed between these nodes i.e., (e,d), (e,b), (e,c).

$$n_B(a) = \frac{\sigma_{ed}(a)}{\sigma_{ed}} + \frac{\sigma_{eb}(a)}{\sigma_{eb}} + \frac{\sigma_{ec}(a)}{\sigma_{ec}} = 1 + 1 + 1 = 3 \tag{3.2}$$

Node betweenness centrality can be a useful feature to detect botnets especially in detecting P2P botnets where bots are more interconnected without a central C2C structure. So, we assume that for a P2P bot in a botnet should have a higher node betweenness centrality in a graph.

3.1.7 Local Clustering Coefficient

Local clustering coefficient of a node indicates how concentrated the neighborhood of that node is. More specifically, local clustering coefficient is a metric to evaluate how close a node's neighbors are to each other. If K_a denotes the number of neighbors of node 'a' and e_a denotes the number of connected pairs between all neighbors of node 'a', then local clustering coefficient for node 'a' can be given by [81]:

$$C_a = \frac{e_a}{K_a(K_a - 1)} \tag{3.3}$$

Figure 3.3 shows the clustering coefficient of node 'a' which is 0.083. Local clustering coefficient can also be a very significant indicator of a P2P botnet. As explained before, bots in P2P botnet have a decentralized structure where bots connect and communicate with each other to remove the need of a centralized server. As a result, interconnectedness can be a very significant feature to detect P2P botnets which is essentially the basis of local clustering coefficient.

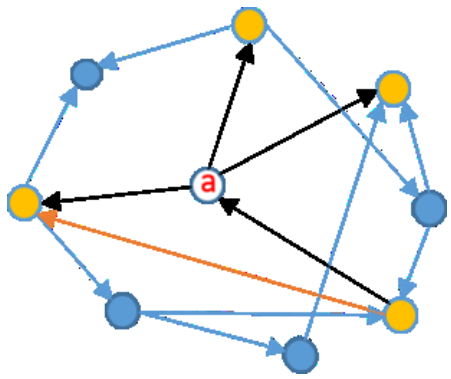

Figure 3.3 Clustering coefficient of node 'a' in a directed graph

Clustering coefficient of node 'a' in directed graph, $\boldsymbol{C_a = \frac{1}{(4*3)} = 0.083}$

Local clustering coefficient can also be a very significant indicator of a P2P botnet. As explained before, bots in P2P botnet have a decentralized structure where bots connect and communicate with each other to remove the need of a centralized server. As a result, interconnectedness can be a very significant feature to detect P2P botnets which is essentially the basis of local clustering coefficient.

3.1.8 Eigen Vector Centrality

Eigen vector centrality, also known as Eigen centrality is a measurement criterion of influence of a node in a graph. It is essentially the weight of a node in a graph [82].

Each node is assigned a relative value based on the concept that connections to high-scoring nodes contribute more to the score of the node than equal connections to low-scoring nodes. Let G(V,E) be a graph where V is total number of nodes and E is the total number of edges. Let, A= (a_(v,w)) be the adjacency matrix where

$$a_{v,w} = \begin{cases} 1 \; if \; node \; v \; is \; linked \; to \; node \; w \\ 0 \; if \; node \; v \; is \; not \; linked \; to \; node \; w \end{cases} \tag{3.4}$$

Then the centrality score can be given as

$$x_v = \frac{1}{\lambda} \sum_{w \in M(v)} a_{v,w} x_w \tag{3.5}$$

where $M(v)$ is the set of neighbors of node 'v' and λ is a constant. Now equation (1) can be rewritten as

$$Ax = \lambda x \tag{3.6}$$

There exists a positive solution λ with final eigenvector after using power method based on the Perron–Frobenius theorem [83]. λ is also the largest eigenvalue associated with the eigenvector of the adjacency matrix [84]. Eigenvector centrality is a natural extension of degree centrality. In-degree centrality awards one centrality points for every link a node receives. But not all nodes are equivalent: some are more important than others based on their edge weight, and, reasonably, connections from important nodes count more. We expect that a bots eigenvector centrality measure should be significantly different than non-malicious nodes and hence is used as a feature to detect botnets in this study.

3.2 Self Organizing Map

Self Organizing Map (SOM) belongs to an interesting class of unsupervised system that is based on competitive learning in which the output neurons compete amongst themselves to be activated. The primary goal of an SOM is to convert an incoming dataset of arbitrary dimension into a one or two-dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion [85]. In this study, we have considered on a particular kind of SOM known as Kohonen network that was developed by Tuevo Kohonen in 1982 [95-96].

The basic structure of SOM is shown in Figure 3.4 is a 3×3 SOM network. For this small SOM network, there are 63 connections. Notice that the map nodes$(C_1 - C_9)$ are not connected to one another. In this 2-D representation of SOM, each map node has a unique (i,j) coordinate. Simultaneously, as map nodes are only connected to input vector $(F_1 - F_7)$, map nodes are never aware of what other map nodes values are. A map node's weight (W) will only be updated if and only if the input vector tells it. Algorithm 1 illustrates the basic methodology behind SOM.

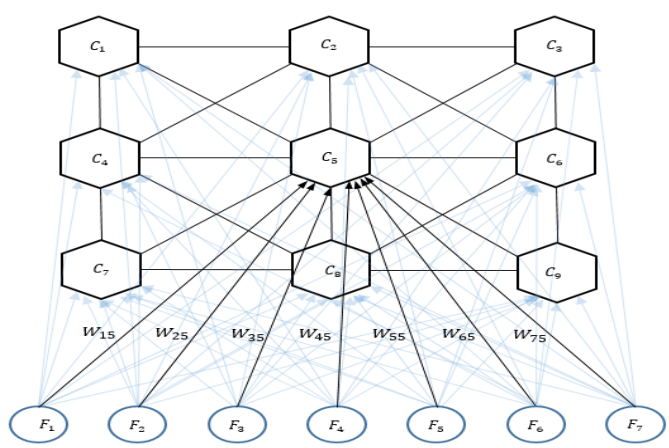

Figure 3.4 Structure of Self Organizing Map

Table 3.2 Algorithm 1: SOM Algorithm

1. Each map nodes($\boldsymbol{C_1 - C_9}$) weights (W) are initialized with small random values.

2. An input vector ($\boldsymbol{F_1 - F_7}$) is chosen from the training dataset and is presented to the network.

3. Each node is inspected to determine which node's weight best matches the input vector's weight. The winning node is termed as 'Winning Neuron' or 'Winner Takes All Neuron' or 'Best Matching Unit (BMU)'. BMU can be calculated as

$$\boldsymbol{BMU = distance\ from\ input\ vector = \sum_{i=1}^{D}(F_i - W_i)^2} \tag{3.7}$$

4. The radius of the BMU is calculated which is typically set to be the radius of the network that diminishes at each time-step. This can be calculated as

$$\boldsymbol{\sigma(t) = \sigma_0 e^{\frac{-t}{\lambda}}} \tag{3.8}$$

where t is the current iteration, $\boldsymbol{\lambda}$ is the time constant and σ_0 is the radius of the map. λ can be calculated as

$$\boldsymbol{\lambda = \text{number of iterations}/\text{mapRadius}} \tag{3.9}$$

Any node found within the radius of BMU is adjusted to make more like the input vector. This adjustment can be done by

$$\boldsymbol{W(t+1) = W(t) + \Phi(t)L(t)\big(X(t) - W(t)\big)} \tag{3.10}$$

Where,

$$\boldsymbol{L(t) = L_0 e^{\frac{-t}{\lambda}}} \tag{3.11}$$

$$\boldsymbol{\Phi(t) = e^{\frac{-distance\ From\ BMU^2}{2\sigma^2(t)}}} \tag{3.12}$$

W(t+1) is the new educated weight value of a given node and $\Phi(t)$ is a measure that is used to force the nodes closer to BMU to learn more than others who are further away.

5. Repeat 2 for desired number of iterations

In this study, an investigated has been conducted for the accuracy of the detection algorithm with three different SOM models, i.e., 4*4, 5*5, and 6*6. Among these three SOM models, 5*5 provides the best solution. With 4*4 SOM model, bots were being identified in a comparatively larger cluster than 5*5 SOM model whereas with 6*6 SOM model, many clusters were empty. Hence, 5*5 SOM model was used for the

demonstration of the effectiveness of this proposed detection method. However, the difference among the results was not significant and using any of them will result in good accuracy. Each map nodes weights in the network are initially assigned with seven random values. After that input vectors, each containing these seven features are presented to the network. Thenceforth, step 3 to 6 is followed to get the desired number of clusters.

In its essence, algorithm 1 is essentially screening the dataset and assigning the nodes to different clusters. This algorithm does not distinguish bots from non-bots. Hence another algorithm is developed to detect bots in the clusters. This bot detection algorithm is illustrated below:

Table 3.3 Algorithm 2: Bot search algorithm

1. Arrange the clusters in ascending order of size.
2. Remove the cluster with the highest number of nodes.
3. Starting with the smallest cluster, investigate all the nodes in the rest of the clusters.
4. Stop the algorithm when bots are detected. The number of nodes needed to identify the bots is denoted by N_s, which characterizes the efficiency of the proposed bot detection algorithm.

It is consequential that algorithm 2 is performed with caution as the efficiency of SOM method can be significantly hampered if bots are not identified properly in this step. After initiating step 1, step 2 is performed based on considering our finding that botnet flows are typically a small proportion of the overall dataset. As the percentage of botnet flow is very small, we delete the cluster that have the highest number of nodes by

assuming bots are not large in number and don't possess the usual similarities to share the same cluster with normal nodes. The criteria for differentiating bots from non-bots must be clearly defined and implemented for the success of this method. Eventually, this is also true that we cannot guarantee the largest cluster will not contain a bot if the bot acts like normal node and is not triggered to be like a bot until step 2 is initiated.

CHAPTER IV

CASE STUDY-DETECTING BOTS IN CTU-13

We apply SOM to the CTU-13 dataset to investigate the effectiveness and efficiency of our proposed method. An enhanced filtering algorithm, based on the degree of bots, is proposed to further improve the botnet detection efficiency. The results are benchmarked against a Support Vector Machine based classification algorithm to demonstrate the strength of our proposed procedure.

4.1 Graph Features Extraction

We first extract graph-based features of CTU-13 data sets as discussed in Chapter 3. Recall that the CTU-13 data sets contain more than 20 million NetFlow records. High performance computing is needed to streamline to the extraction of graph-based features. The computation tasks of feature extraction are performed using the Shadow system, super computer available at The High Performance Computing Collaboratory (HPC²) of Mississippi State University. The Shadow system is equipped with a Cray CS300-LC cluster with 4800 Intel Ivy Bridge processor cores and 28,800 Intel Xeon Phi cores. With the aid of high performance computing capacity, we are able to extract the graph-based features from all CTU-13 data sets within 20 hours. The resultant graph-based features are numbered and labeled by Feature 1 – Feature 7 for the notational convenience, as shown in Table 4.1.

Table 4.1 Graph-Based Features Used for Clustering.

1	2	3	4	5	6	7
In Degree	Out Degree	Sum of Ingoing Edges Weight	Sum of Outgoing Edges Weight	Clustering Coefficient	Node Betweenness	Eigen Vector

4.1.2 Graph-Based Botnet Detection Using Clustering

We apply the SOM-based botnet detection algorithm (Algorithm 1) to the extracted seven graph-based features. Fig. 4.1 demonstrates the results of SOM clustering based on CTU dataset 6. There is a total number of 25 cells, each representing a possible cluster of graph-based features. We choose the total number of cells to be 25 so that the SOM algorithm can captures various types of node behaviors while not significantly increasing computation costs.

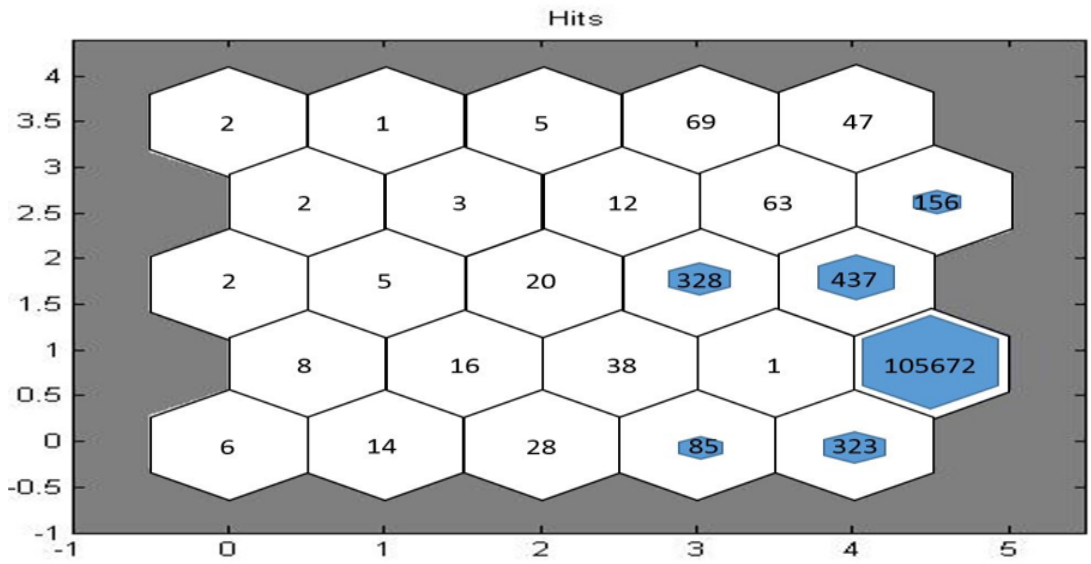

Figure 4.1 SOM hits on CTU-13 dataset 6

Table 4.2 Number of nodes in the biggest cluster (Normal Nodes).

Dataset	No. of Nodes	No. of Nodes in the biggest cluster	% of nodes to be eliminated for bot detection
1	311420	309741	99.4608
2	442471	440505	99.5556
3	434988	433852	99.7388
4	186245	185042	99.3540
5	41658	40874	98.1180
6	107343	105672	98.4433
7	38205	37260	97.5265
8	383788	382670	99.7086
9	367264	366271	99.7296
10	197824	196996	99.5814
11	41933	40603	96.8282
12	94436	93104	98.5895
13	315769	313326	99.2263

From Figure 4.1, the numbers in each cell represent the total number of nodes that belong to the corresponding cluster. These nodes share similar behaviors in terms of the identified graph-based features. For example, there exist 105,672 nodes in the biggest cluster (in blue), which accounts for over 99% of nodes in Dataset 6. Note that malicious behaviors i.e., the botnet flows are typically a small proportion of the entire dataset and when compared to normal flows, botnet flows possess high range of feature values because they are very active in the network. So, we delete the clusters that have the highest number of nodes that don't possess the usual characteristics that a bot might have. This helps to narrow down the identification of bots to the remaining few nodes, which account for less than 1% of the total nodes. Similar observations are made for the other

CTU datasets that the majority of the nodes belongs to the biggest cluster and can be eliminated from the consideration of bot detection (see Table 4.2). For most of the CTU scenarios, the biggest cluster consists of over 99% of nodes. This allows us to eliminate the majority of the dataset for further bot identification, significantly reducing the cost of computation.

We apply the proposed bot search algorithm (Algorithm 2) to the clusters obtained via SOM. Table 4.3 shows the number of nodes to search to identify all bots in each data set. The sizes of clusters that include the bots are also reported. Bots can be isolated in small clusters for most data sets. As a result, bots can be identified by examining limited number of nodes.

Table 4.3 Number of Nodes to Search for Bot Identification (N_s)

Dataset	Number of bots	Number of identified bots	Size of the bot cluster	N_s	% of nodes to search
1	1	1	27	120	0.038
2	1	1	12	41	0.009
3	1	1	26	125	0.028
4	1	1	40	238	0.127
5	1	1	6	26	0.062
6	1	1	38	163	0.151
7	1	1	11	44	0.115
8	1	1	184	563	0.146
9	10	3	21	73	0.019
		7	40	63	0.017
10	10	10	20	90	0.045
11	3	2	9	24	0.057
		1	770	1306	3.114
12	3	2	11	53	0.056
		1	19	60	0.063
13	1	1	16	64	0.020

It shows that the proposed method can detect botnet size in a cluster which is very small compared to the size of the total dataset. Hence, after applying SOM on the dataset, further investigating the nodes of the small clusters gives the bots present in the dataset. Bots have been mostly found in small sized clusters. More specifically, in more than 80% of the cases, bots have been found within the smaller clusters containing only 20% of the remaining nodes. Although, it still may take some computational effort to further investigate clusters after initial screening, it is considerably less than the computational time and complexity resulting from the framework where the entire dataset needs to be examined.

From Table 4.4, it is apparent that although SOM methodology provides good results in alienating bots from the rest of the nodes, there are no unique values of features across all bot clusters. From Figure 4.2 it is clear that feature values are far apart for different nodes. The highest and lowest values of features have been made bold to better clarify the finding. For example: feature 1 values range from 1 to 6842 and feature 2 values range from 3 to 11571 across all bot clusters. So, there is no fixed range for the feature values of bots across all the scenarios. A notable conclusion that can be made from this experiment is that rule based detection methods will not work well in detecting botnets as different bots behave differently in different scenarios. Thus, detecting botnets become very challenging. This limitation can be by passed by the proposed approach as it does not rely on any particular rule. With different types of bot behaviors, the proposed method can still detect bot with reasonable accuracy. What this approach ensures is that, bots will always be found in small sized clusters. A majority of the data (>97%) is removed from consideration, and thus the sample space becomes very negligible. This relatively smaller sample space need be further investigated to detect botnets. Hence, this proves the robustness of our proposed approach as it can detect botnets with varying behavior in different datasets.

Table 4.4 Feature values of bot cluster

Dataset	Size of the bot cluster	Features						
		1	2	3	4	5	6	7
1	27	176	2703	2595	48690	0.0113	0.0007	0.00022
2	12	110	4161	3140	134500	0.0109	0.0003	0.00017
3	26	1727	2391	**176967**	8806	0.0171	0.0019	0.00122
4	40	153	859	1970	43356	0.0847	0.0003	0.00056
5	6	7	483	310	28527	**0**	0.0004	1.14E-05
6	38	26	428	1443	12128	0.0449	0.0003	4.72E-05
7	11	25	385	231	21990	0.0283	0.0005	0.00033
8	184	34	289	470	17650	**0.1344**	**3.08E-5**	**3.84E-06**
9	40	150	6534	3214	121087	0.0512	0.0007	0.000990
	21	86	5240	2662	219182	0.0006	0.0006	0.00055
10	20	**6842**	7462	6581	**355098**	0.0145	**0.0023**	0.16417
11	9	1219	2883	1223	27505	**0**	0.0006	**0.19237**
	770	**1**	**3**	**3**	**100**	0.0110	6.22E-5	5.51E-06
12	11	509	328	57287	6897	**0**	0.0019	0.00017
	19	169	85	31608	2771	0.0217	0.0005	0.000283
13	16	161	**11571**	974	267041	0.0408	0.0003	4.69E-05

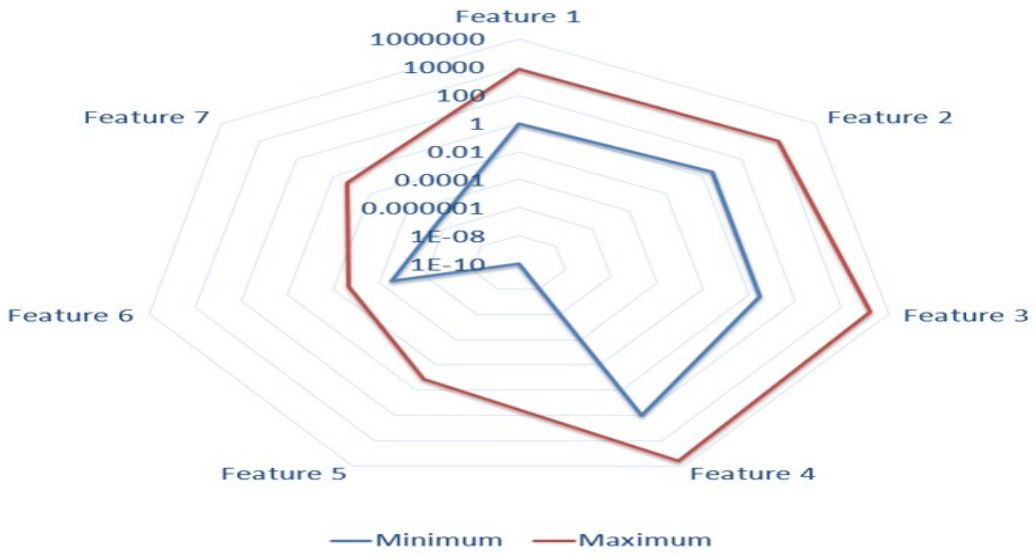

Figure 4.2 Variation of average feature values in Bot clusters

4.2 Feature Evaluations and SOM Based Botnet Detection on Filtered Dataset

In this section, SOM has been applied to filtered CTU-13 dataset to determine whether filtering the raw data provides better result than shown in section 4.1. What filtering is essentially doing is that it is removing nodes that cannot be a bot. The basic assumption made here is that 1-degree nodes can't be a bot as they are not very active in the network. As a result, total number of nodes where SOM needs to be applied get significantly reduced. The steps of filtering are provided below:

1. First convert the flow-based data into graph-based data.
2. Each IP is considered as a node and each connection is considered as an edge.
3. If there are multiple communications between two nodes, we still represent them with a single edge, and add other data as weight (attributes) of that edge.

4. Calculate the feature values of each node.
5. Compute the degree of each node, and then filter out (remove) the 1-degree nodes and their corresponding edges from the graph

Subsequently, SOM has been applied to this filtered dataset. Note that, we have only used ten filtered datasets for experimental purpose. Results obtained from applying SOM on filtered dataset is shown in Table 4.5.

Table 4.5 Efficiency of bot detection

Dataset	Total number of nodes	Number of bots	Number of identified bots	Size of the bot cluster	N_s	% of nodes to search
1	117119	1	1	27	115	0.098
3	20284	1	1	18	96	0.473
4	81544	1	1	33	181	0.002
5	1939	1	1	7	65	3.352
6	8240	1	1	45	252	3.058
7	2486	1	1	8	40	1.609
8	20666	1	1	61	307	1.485
10	91785	10	10	17	77	0.083
11	2498	3	2	8	21	0.840
			1	146	384	15.372
12	4743	3	3	5	36	0.759

Results in Table 4.5 show that, after filtering, total number of nodes to examine to apply SOM gets reduced. Moreover, after applying SOM, in majority of the cases the size of the cluster where bot is found is smaller than before. Figure 4.3 illustrates this phenomenon. It is clearly evident from the figure that, for the 10th scenario the bot cluster size is smaller after filtering. Here, the red star is the bot and black dots are the other non-malicious nodes in the cluster.

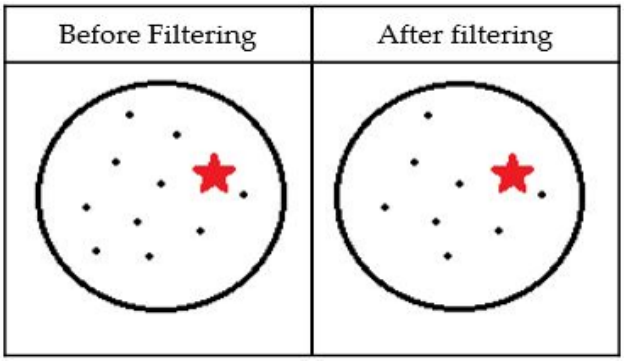

Figure 4.3 Bot cluster size before and after filtering

As a result, the numbers of nodes to search for bot identification (N_s) are shown in Table 4.6 Significant reduction in N_s can be observed. For example, 1306 nodes need to be searched for identifying the third bot in Dataset 11. After filtering, 384 nodes need to be searched only, a reduction of over 70% of the total number of nodes. For dataset 12, the two clusters containing bots are combined into one after filtering, requiring searching 36 nodes only compared to 113 nodes before clustering. However, we also observed the N_s values slightly increase for datasets 5 and 6, which may result from randomness of the clustering algorithm.

Table 4.6 Improvement of N_s using Filtering

Dataset	Botnet detection without filtering		Botnet detection after filtering	
	N_s	% nodes to search	N_s	% nodes to search
1	120	0.038	115	0.036
3	125	0.028	96	0.022
4	238	0.127	181	0.097
5	26	0.062	65	0.156
6	163	0.151	252	0.234
7	44	0.115	40	0.104
8	563	0.146	307	0.079
10	90	0.045	77	0.038
11	24	0.057	21	0.050
	1306	3.114	384	0.915
12	53	0.056	36	0.038
	60	0.063		

4.3 Extension of SOM Implementation on ISCX botnet test dataset

In order to check the efficiency of the proposed methodology to compare with CTU-13, we have extracted the same 7 graph-based features for the new ISCX botnet test dataset. We have implemented the proposed SOM Algorithm 1, same as before. From Figure 4.4 the numbers in each cell represent the total number of nodes that belong to the corresponding cluster. These nodes share similar behaviors in terms of the identified graph-based features. For example, there exist 26,652 nodes in the biggest cluster (in blue), which accounts for over 93% of nodes (Table 4.7) in ISCX botnet test dataset.

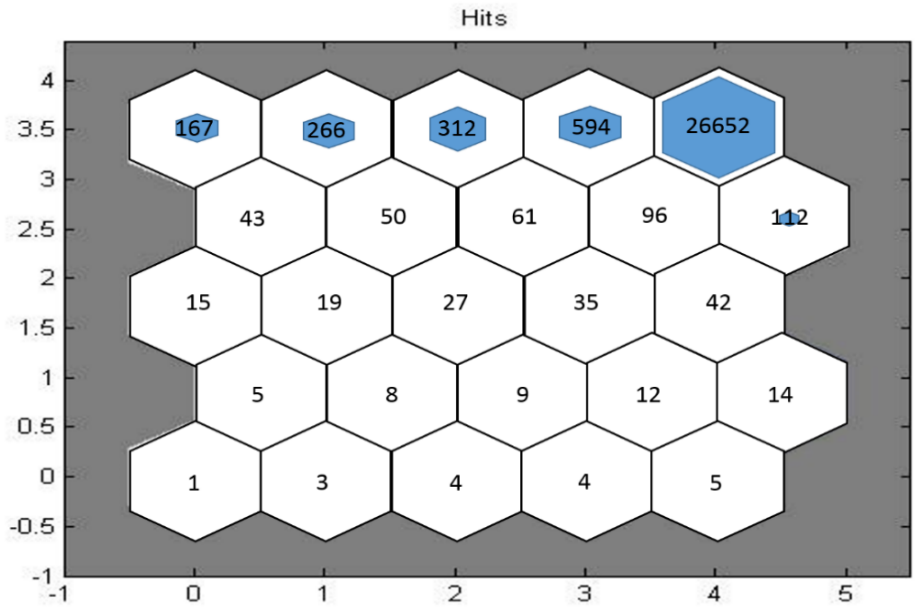

Figure 4.4 SOM on ISCX botnet test dataset

From Table 4.7 number of bots identified in each cluster shows the malicious nodes of ISCX dataset have been scattered into the smallest clusters. As we discussed before abnormal/malicious behaviors are rare in most of real-world networks and the biggest cluster (with maximum number of nodes) are unlikely to be bots. In further inspection, the nodes in the biggest cluster can be eliminated to reduce the computational costs in further investigation of bots. From Table 4.8 the percentage of nodes to be eliminated is more than 93%. Comparing with the results of CTU-13 dataset, it is true that similar observations are drawn when the proposed methodology has been implemented on ISCX Botnet test dataset. It again proves the robustness of the proposed methodology and also we expect the method of detecting bot in the rest of the small

clusters by eliminating the biggest cluster will hold true for any other new datasets but we cannot guarantee it.

Table 4.7 Number of nodes in each cluster of ISCX Botnet test dataset after implementing proposed methodology

Cluster number	Number of nodes in each cluster	Number of bots identified in each cluster
1	1	0
2	3	0
3	4	2
4	4	0
5	5	2
6	5	1
7	8	0
8	9	0
9	12	0
10	14	2
11	15	1
12	19	2
13	27	1
14	35	1
15	42	5
16	43	1
17	50	0
18	61	3
19	96	4
20	112	3
21	167	1
22	266	2
23	312	0
24	594	3
25	26652	0

Table 4.8 Create a short, concise table title and place all detailed caption, notes, reference, legend information, etc in the notes section below

Dataset	Number of Bots	Total number of nodes	Number of nodes in the biggest cluster	% of nodes to eliminated
ISCX Botnet Test	35	28556	26652	93.3324

4.4 Benchmark Against Classification Techniques

We compare our proposed clustering approach with some of the available classification techniques to detect bots.

4.4.1 Support Vector Machine Classifier

Support vector machine (SVM) is a powerful supervised machine learning technique [54], which is used for classification and regression analysis. It is introduced by Cortes and Vapnik [53]. Basically, SVM classifies the data into two classes by generating an optimal hyper-plane, which has the largest distance to the nearest training samples. To predict the class of new observations, the SVM learning algorithm, splits data to training and validation set. The decision boundary (i.e., a hyper-plane) is determined using training set. Subsequently, SVM classifier predicts class of the observations for validation set based on the distance of each observation from decision boundary. Optimal hyperplane dividing the data in to two classes can be written as set of point $\vec{x}$ satisfying $\vec{w}.\vec{x} + \mathrm{b} = 0$, where $\vec{w}$ is the normal vector of the hyperplane [55]. The parameter b gives the offset distance from the origin. The parallel marginal hyperplanes can be given by the equations,

$$\vec{w}.\vec{x} + \mathrm{b} = 1 \tag{4.1}$$

$$\vec{w}.\vec{x} + \mathrm{b} = -1 \tag{4.2}$$

Two parallel marginal hyperplanes are generated on both the sides of the optimal hyperplane that separates the data. The sample points which are used to generate the optimal hyper-plane are called the support vectors (SVs). The distance between the two marginal hyperplanes with the largest margin is given by $M = \frac{2}{||w||}$

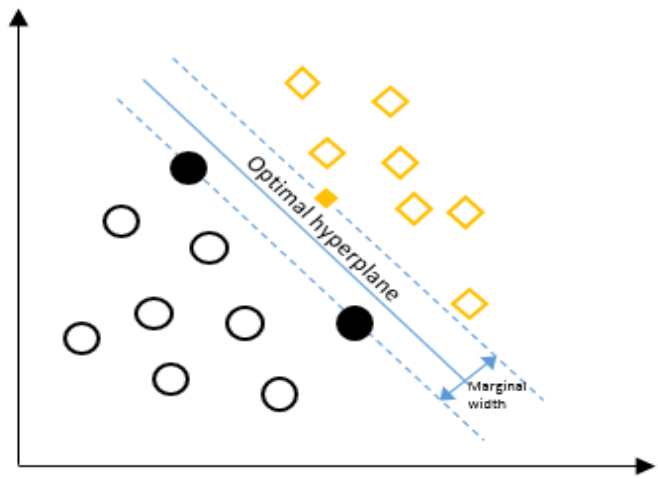

Figure 4.5 Shows the linear separating hyperplane for the separable case, and the solid circle and squares on the margin are called support vectors.

If the dataset is not linearly separable, one can use more general kernel functions that provides non-linear decision boundaries by generating a hyperplane in a multi-dimensional feature space. The kernel function(ϕ) plays a critical role in the SVM training and classification. Some commonly implemented kernel functions [9056] are the Gaussian radial basis function (RBF) kernel, Polynomial kernel and the Sigmoid kernel. The advantage of SVM is that it works well with small training datasets.

4.4.2 K-Nearest Neighbor (K-NN) Classifier

K-NN is one of the widely-used machine learning algorithms, which is an extension of the nearest neighbor (NN) classifier [9157]. K-NN classifies an object by choosing the majority vote of its nearest neighbors. Here, the object will be designated to a class based on the most frequent class of its K nearest neighbors, where K is a user defined constant. In a multidimensional feature space, all the training sample are vectors assigned with a class label. During the training phase, the classifier remembers the class labels and feature values of the training samples. For instance, assume that x_0 is a test point (an unlabeled vector) which is needed to be classified in a testing phase. When a K-nearest neighbor query starts, it grows like a spherical region until the query is enclosed by K training samples. When the classifier finds the set of desired K nearest neighbors in the training set to x_0, it classifies the test point as the most frequent class among the K neighbors closer to it.

Considering the outcome of K-NN on 1 nearest neighbor as shown in the example Figure 4.6 the prediction of K-NN of the test sample (orange circle) will be '+' as it is closer to it. If K = 2, K-NN will be not able to classify the test sample outcome since the second closest sample is '-', both the minus and plus signs receive the same score. If K is 3 then the outcome is '-', and if K is 5 then it is '+' as the respective signs dominate the nearest neighbors in each case. In our case the k value is 5 and Euclidian distance method is used to compute the nearest distance between the test sample and training sample.

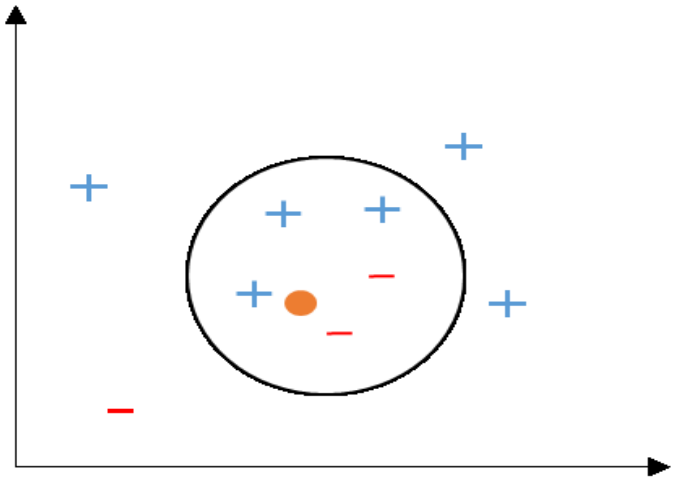

Figure 4.6 K-NN classification approach

4.4.3 Decision Tree classifier (DT)

Decision Tree is a well-known supervised machine learning technique that is used for classification and regression analysis. The basic idea of DT is to predict the class of a variable based on the training model by learning decision rules. The algorithm of DT is very simple and it can be represented by a tree structure. Initially while training, the algorithm tries to split the root node into subsets based on the decision value and it goes till the leaf node is found. Hence, whenever there is a new set of data point to predict, DT simply compares the new data point with the trained model and determines which class it belongs to. For instance, form the Figure 4.7, if there is a new data Z and needs to be defined weather Bot or None-Bot, then first the algorithm tries to use the attributes value of the test data and compare with the training set. Assume if the value of the attribute x_2 is < -0.55 then it picks the left branch and goes to next subset and again if the value of attribute x_1<1.5 then it predicts the class of Z as a Bot else Non- Bot.

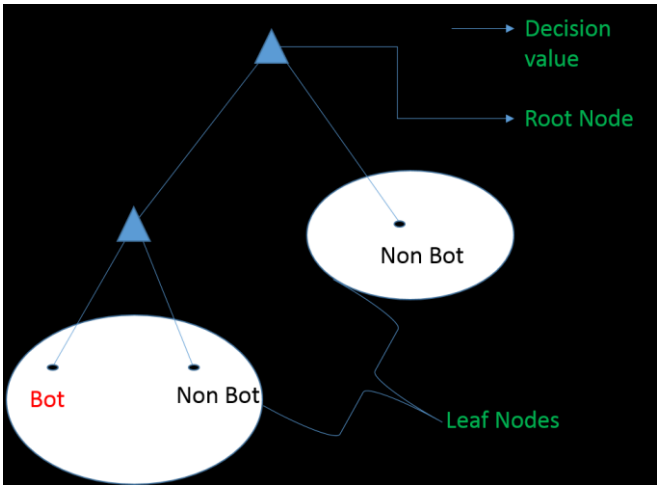

Figure 4.7 Example of Decision Tree Classification

4.5 Classification Results

The programs of all the three classifiers is available in MATLAB packages. We train the SVM, K-NN and DT classifiers using ISCX botnet test dataset and use CTU-13 datasets for testing the classifiers. Training the classifiers in one of the important step while implementing classification techniques. As ISCX dataset is a combination of three different subset datasets which includes CTU data we choose this as a training dataset. During the training phase the classifier learn and frame guidelines in differentiating bot and non-bot based on the feature values provided.

The classification techniques cannot provide efficient result in accurately classifying the CTU-13 datasets to Bot and Non-Bot classes, since there is high variation in feature values. Specifically, due to the complex behaviors of bots, classification becomes challenging as characteristics of training and testing data can significantly vary. The percentage of misclassification by using three classifiers are presented in Table 4.9. The three classifiers are only capable of determining just 10% of accurate classification

for the 10th and 11th datasets only. However, for the rest of the scenarios, misclassification rates are 100%.

Table 4.9 Classification result.

Implemented on CTU 13 Dataset	**% of Misclassification and comparison between classifiers trained with ISCX Botnet dataset**		
	SVM	**DT**	**KNN**
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100
6	100	100	100
7	100	100	100
8	100	100	100
9	100	100	100
10	90	90	90
11	90	90	90
12	100	100	100
13	100	100	100

CHAPTER V

Conclusion

In this work, we propose a graph-based botnet detection approach that can detect changing behaviors of bots. This is novel because the existing approaches mainly rely on flow-based features and thus do not capture the changes in the topological structure of networks caused by bot activities. We investigate seven graphed-based features that are may be connected to bot activities: in degree, out degree, in degree weight, out degree weight, clustering coefficient, node betweenness, and eigenvector centrality. SOM is applied to establish the clusters of nodes based on these graphed features. Our approach is capable of isolating bots in clusters with very small sizes (less than 100 nodes), which enables fast detection of bot nodes. The proposed algorithm is further enhanced by filtering out inactive nodes, which are unlikely to be bots. We verify the proposed methods using CTU-13 and ISCX Botnet dataset. Numerical results show that our proposed procedure is capable of detecting the bots by searching limited number of nodes.

We compare our approach with three different classification algorithms using the same graph-based features. All the methods are not capable of detecting most of the bots because of the varying values of bot features across different datasets (Bot features vary from one dataset to another). The advantage of our approach is that we focusing on capturing the abnormal behaviors of bots in terms of their graph-based behaviors. In other

words, our method is more robust against the changing behaviors of bots because the proposed approach does not rely on any particular value/range of features. With different types of bot behavior, the proposed method can still detect bot with reasonable accuracy. What this approach ensures is that, bots will always be found in small sized clusters with the majority of nodes (>99%) removed from further consideration. Our study shows that, as long as the bots behave differently from normal nodes, such different behaviors can be captured by our clustering-based detection algorithm and further testing is needed in determining the bots from the smallest clusters. Future work is needed to incorporate additional graph-based features and reduce the computational costs of graph feature extraction and testing on other datasets. Note that, as feature extraction cost contributes to the overall computational cost, future work is needed to investigate how feature extraction cost can be minimized. Effect of incorporating more relevant graph-based features into the detection methodology is also a future research direction.

REFERENCES

[1] Welivesecurity. Botnet malware: What It Is and How to Fight It. Available from: <*http://www.welivesecurity.com/2014/10/22/botnet-malware-fight/*>; **2014** [accessed 21.12.15]

[2] Barford, P., & Yegneswaran. V. (2006). An Inside Look at Botnets. *Special Workshop on Malware Detection: Advances in Information Security* **2006**.

[3] F-scure. Articles: Botnets. Available from <***https://www.fsecure.com/en/web/labs_global/botnets***>; **2016** [accessed 21.02.16]

[4] Zeidanloo, H. R., Shooshtari, M. J. Z., Amoli, P. V., Safari, M., & Zamani, M. (2010, July). A taxonomy of botnet detection techniques. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on* (Vol. 2, pp. 158-162). IEEE.

[5] Ianelli, N & Hackworth, A. (2005). Botnets as a Vehicle for Online Crime. Available from <*https://resources.sei.cmu.edu/asset_files/WhitePaper/2005_019_001_51249.pdf*>; **2005** [accessed 24.04.2016]

[6] Bacher, P., Holz, T., Kotter, M., & Wicherski, G. (2005). Know Your Enemy: Tracking Botnets. Available from< *https://www.honeynet.org/papers/bots/*>; **2005** [accessed 24.04.2016]

[7] Kaspersky. What is Botnet attack? Available from: <*https://usa.kaspersky.com/internet-security-center/threats/botnetattacks#.V1du3TUrIdV*>; **2016** [accessed 21.02.16]

[8] Sonawane, S.R. (2016). A Review on Botnet and Botnet Detection Methods. International Journal of Computer Science and Innovation **2016**; volume 1: pp.107-116.

[9] Fadlullah, Z. M., Taleb, T., Vasilakos, A. V., Guizani, M., & Kato, N. (2010). DTRAB: Combating against attacks on encrypted protocols through traffic-feature analysis. *IEEE/ACM Transactions on Networking (TON)*, *18*(4), 1234-1247.

[10] Zhang, J., Chen, C., Xiang, Y., Zhou, W., & Xiang, Y. (2013). Internet traffic classification by aggregating correlated naive bayes predictions. *IEEE Transactions on Information Forensics and Security*, *8*(1), 5-15.

[11] Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., & Guan, Y. (2013). Network traffic classification using correlation information. *IEEE Transactions on Parallel and Distributed Systems*, *24*(1), 104-117.

[12] Zhang, J., Chen, C., Xiang, Y., Zhou, W., & Vasilakos, A. V. (2013). An effective network traffic classification method with unknown flow detection. *IEEE Transactions on Network and Service Management*, *10*(2), 133-147.

[13] Yan, Z., Zhang, P., & Vasilakos, A. V. (2015). A security and trust framework for virtualized networks and software- defined networking. *Security and communication networks*.

[14] Shu, Z., Wan, J., Li, D., Lin, J., Vasilakos, A. V., & Imran, M. (2016). Security in software-defined networking: Threats and countermeasures. *Mobile Networks and Applications*, *21*(5), 764-776.

[15] Zhang, J., Perdisci, R., Lee, W., Sarfraz, U., & Luo, X. (2011, June). Detecting stealthy P2P botnets using statistical traffic fingerprints. In *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on* (pp. 121-132). IEEE.

[16] Choi, H., & Lee, H. (2012). Identifying botnets by capturing group activities in DNS traffic. *Computer Networks*, *56*(1), 20-33.

[17] Livadas, C., Walsh, R., Lapsley, D., & Strayer, W. T. (2006, November). Usilng machine learning technliques to identify botnet traffic. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on* (pp. 967-974). IEEE.

[18] Goebel, J., & Holz, T. (2007). Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, **2007**.

[19] Zeidanloo, H. R., Manaf, A. B., Vahdani, P., Tabatabaei, F., & Zamani, M. (2010, June). Botnet detection based on traffic monitoring. In *Networking and Information Technology (ICNIT), 2010 International Conference on* (pp. 97-101). IEEE.

[20] Argus (Audit Record Generation and Utilization System. Available from< *http://www.qosient.com/argus*>; **2016** [accessed 21.02.2016]

[21] Binkley, J. R., & Singh, S. (2006). An Algorithm for Anomaly-based Botnet Detection. *SRUTI*, *6*, 7-7.

[22] Karasaridis, A., Rexroad, B., & Hoeflin, D. (2007). *In USENIX Workshop on Hot Topics in Understanding Botnet*, **2007**.

[23] Gu, G., Zhang, J., & Lee, W. (2008). BotSniffer: Detecting botnet command and control channels in network traffic.

[24] Strayer, W. T., Lapsely, D., Walsh, R., & Livadas, C. (2008). Botnet detection based on network behavior. In *Botnet Detection* (pp. 1-24). Springer US.

[25] Amini, P., Azmi, R., & Araghizadeh, M. (2014). Botnet Detection using NetFlow and Clustering. *Advances in Computer Science: an International Journal*, *3*(2), 139-149.

[26] Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008, July). BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection. In *USENIX Security Symposium* (Vol. 5, No. 2, pp. 139-154).

[27] Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., & Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, *39*, 2-16.

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

[29] Strayer, W. T., Walsh, R., Livadas, C., & Lapsley, D. (2006, November). Detecting botnets with tight command and control. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on* (pp. 195-202). IEEE.

[30] Lu, W., Rammidi, G., & Ghorbani, A. A. (2011). Clustering botnet communication traffic based on n-gram feature selection. *Computer Communications*, *34*(3), 502-514.

[31] Al-Duwairi, B., & Al-Ebbini, L. (2010, May). BotDigger: A fuzzy inference system for botnet detection. In *Internet Monitoring and Protection (ICIMP), 2010 Fifth International Conference on* (pp. 16-21). IEEE.

[32] AsSadhan, B., Moura, J. M., Lapsley, D., Jones, C., & Strayer, W. T. (2009, July). Detecting botnets using command and control traffic. In *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on* (pp. 156-162). IEEE.

[33] Venkatesh, B., Choudhury, S. H., Nagaraja, S., & Balakrishnan, N. (2015). BotSpot: fast graph based identification of structured P2P bots. *Journal of Computer Virology and Hacking Techniques*, *11*(4), 247-261.

[34] Ding, Q., Katenka, N., Barford, P., Kolaczyk, E., & Crovella, M. (2012, August). Intrusion as (anti) social communication: characterization and detection. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 886-894). ACM.

[35] Henderson, K., Gallagher, B., Eliassi-Rad, T., Tong, H., Basu, S., Akoglu, L., ... & Li, L. (2012, August). Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1231-1239). ACM.

[36] Henderson, K., Gallagher, B., Li, L., Akoglu, L., Eliassi-Rad, T., Tong, H., & Faloutsos, C. (2011, August). It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 663-671). ACM.

[37] Kang, U., McGlohon, M., Akoglu, L., & Faloutsos, C. (2010, December). Patterns on the connected components of terabyte-scale graphs. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on* (pp. 875-880). IEEE.

[38] Aggarwal, C. C. (2013). Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter*, *14*(2), 49-58.

[39] Zimek, A., Campello, R. J., & Sander, J. (2014). Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *Acm Sigkdd Explorations Newsletter*, *15*(1), 11-22.

[40] Chen, H. H., & Giles, C. L. (2013, August). ASCOS: an asymmetric network structure context similarity measure. In *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on* (pp. 442-449). IEEE.

[41] Sun, H., Huang, J., Han, J., Deng, H., Zhao, P., & Feng, B. (2010, December). gskeletonclu: Density-based network clustering via structure-connected tree division or agglomeration. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on* (pp. 481-490). IEEE.

[42] Tong, H., & Lin, C. Y. (2011, April). Non-negative residual matrix factorization with application to graph anomaly detection. In *Proceedings of the 2011 SIAM International Conference on Data Mining* (pp. 143-153). Society for Industrial and Applied Mathematics.

[43] Ambai, M., Utama, N. P., & Yoshida, Y. (2011). Dimensionality reduction for histogram features based on supervised non-negative matrix factorization. *IEICE TRANSACTIONS on Information and Systems*, *94*(10), 1870-1879.

[44] Nikulin, V., & Huang, T. H. (2012). Unsupervised dimensionality reduction via gradient-based matrix factorization with two adaptive learning rates. In *ICML Unsupervised and Transfer Learning* (pp. 181-194).

[45] Davis, M., Liu, W., Miller, P., & Redpath, G. (2011, October). Detecting anomalies in graphs with numeric labels. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 1197-1202). ACM.

[46] Eberle, W., & Holder, L. (2007, October). Discovering structural anomalies in graph-based data. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on* (pp. 393-398). IEEE.

[47] Kontkanen, P., & Myllymäki, P. (2007). MDL histogram density estimation. *Rn*, *1000*, 2.

[48] Gao, J., Liang, F., Fan, W., Wang, C., Sun, Y., & Han, J. (2010, July). On community outliers and their efficient detection in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 813-822). ACM.

[49] Muller, E., Sánchez, P. I., Mulle, Y., & Bohm, K. (2013, April). Ranking outlier nodes in subspaces of attributed graphs. In *Data Engineering Workshops (ICDEW), 2013 IEEE 29th International Conference on* (pp. 216-222). IEEE.

[50] Perozzi, B., Akoglu, L., Iglesias Sánchez, P., & Müller, E. (2014, August). Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1346-1355). ACM.

[51] Kang U, Papadimitriou S, Sun J, Tong H (2011b) Centralities in large networks: algorithms and observations. In: Proceedings of the 11th SIAM international conference on data mining (SDM), Mesa, AZ, pp 119–130

[52] Gao, X., Xiao, B., Tao, D., & Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and applications*, *13*(1), 113-129.

[53] Bunke, H., Dickinson, P. J., Kraetzl, M., & Wallis, W. D. (2007). *A graph-theoretic approach to enterprise network dynamics* (Vol. 24). Springer Science & Business Media.

[54] Akoglu, L., & Faloutsos, C. (2010, December). Event detection in time series of mobile communication graphs. In *Army science conference* (pp. 77-79).

[55] Rossi, R. A., Gallagher, B., Neville, J., & Henderson, K. (2013, February). Modeling dynamic behavior in large evolving graphs. In *Proceedings of the sixth ACM international conference on Web search and data mining* (pp. 667-676). ACM.

[56] Ishibashi, K., Kondoh, T., Harada, S., Mori, T., Kawahara, R., & Asano, S. (2010, September). Detecting anomalous traffic using communication graphs. In *Telecommunications: The Infrastructure for the 21st Century (WTC), 2010* (pp. 1-6). VDE.

[57] Papalexakis, E. E., Faloutsos, C., & Sidiropoulos, N. D. (2012, September). Parcube: Sparse parallelizable tensor decompositions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 521-536). Springer Berlin Heidelberg.

[58] Leskovec, J., Lang, K. J., & Mahoney, M. (2010, April). Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web* (pp. 631-640). ACM.

[59] Peel, L., & Clauset, A. (2014). Detecting change points in the large-scale structure of evolving networks. *arXiv preprint arXiv:1403.0989*.

[60] Li, L., Mathur, S., & Coskun, B. (2013, October). Gangs of the internet: towards automatic discovery of peer-to-peer communities. In *Communications and Network Security (CNS), 2013 IEEE Conference on* (pp. 64-72). IEEE.

[61] Malware Capture Facility Project. The CTU-13 Dataset: A Labeled Dataset with Botnet, Normal and Background Traffic. Available from < http://mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html >; **2016** [accessed 26.01.2016]

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

[63] Iliofotou, M., Kim, H. C., Faloutsos, M., Mitzenmacher, M., Pappu, P., & Varghese, G. (2011). Graption: A graph-based P2P traffic classification framework for the internet backbone. *Computer Networks*, *55*(8), 1909-1920.

[64] Zhao, Y., Xie, Y., Yu, F., Ke, Q., Yu, Y., Chen, Y., & Gillum, E. (2009, April). BotGraph: Large Scale Spamming Botnet Detection. In *NSDI* (Vol. 9, pp. 321-334).

[65] Jaikumar, P., & Kak, A. C. (2015). A graph theoretic framework for isolating botnets in a network. *Security and Communication Networks*, *8*(16), 2605-2623.

[66] Nagaraja, S., Mittal, P., Hong, C. Y., Caesar, M., & Borisov, N. (2010, August). BotGrep: Finding P2P Bots with Structured Graph Analysis. In *USENIX Security Symposium* (pp. 95-110).

[67] François, J., Wang, S., & Engel, T. (2011, May). BotTrack: tracking botnets using NetFlow and PageRank. In *International Conference on Research in Networking* (pp. 1-14). Springer Berlin Heidelberg.

[68] Francois, J., Wang, S., Bronzi, W., State, R., & Engel, T. (2011, November). Botcloud: Detecting botnets using mapreduce. In *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on* (pp. 1-6). IEEE.

[69] Hang, H., Wei, X., Faloutsos, M., & Eliassi-Rad, T. (2013, May). Entelecheia: Detecting p2p botnets in their waiting stage. In *IFIP Networking Conference, 2013* (pp. 1-9). IEEE.

[70] Collins, M. P., & Reiter, M. K. (2007, September). Hit-list worm detection and bot identification in large networks using protocol graphs. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 276-295). Springer Berlin Heidelberg.

[71] Tsai, C. W., Lai, C. F., Chao, H. C., & Vasilakos, A. V. (2015). Big data analytics: a survey. *Journal of Big Data*, *2*(1), 21.

[72] Fong, S., Wong, R., & Vasilakos, A. V. (2016). Accelerated PSO swarm search feature selection for data stream mining big data. *IEEE transactions on services computing*, *9*(1), 33-45.

[73] Garcia, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. *computers & security*, *45*, 100-123.

[74] Grill, M., Pevný, T., & Rehak, M. (2017). Reducing false positives of network anomaly detection by local adaptive multivariate smoothing. *Journal of Computer and System Sciences*, *83*(1), 43-57.

[75] Chanthakoummane, Y., Saiyod, S., Benjamas., N & Khamphakdee, N. (2016). Evaluation Snort-IDS Rules for Botnets Detection. Available from< *http://www.it.kmitl.ac.th/~natapon/ncit2015/papers/p87-chanthakoummane.pdf*>; [accessed 11.04.2016]

[76] Małowidzki, M., Berezinski, P., & Mazur, M. (2015). Network Intrusion Detection: Half a Kingdom for a Good Dataset. In *Proceedings of NATO STO SAS-139 Workshop, Portugal*.

[77] Graph-tool. Available from < *https://graph-tool.skewed.de/*>

[77] Technopedia.Data Packet, Available from< *https://www.techopedia.com/definition/6751/data-packet*>; **2016** [accessed 05.06.2016]

[78] Rafiei, D. (2005, October). Effectively visualizing large networks through sampling. In *Visualization, 2005. VIS 05. IEEE* (pp. 375-382). IEEE.

[79] Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, *25*(2), 163-177.

[80] Rocchini, C. Hue Scale Representing Node Betweenness on a Graph, Available from < *https://commons.wikimedia.org/w/index.php?curid=1988980*>; **2007** [accessed 05.04.2015]

[81] Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *nature*, *393*(6684), 440-442.

[82] Langville, A. N., & Meyer, C. D. (2005). A survey of eigenvector methods for web information retrieval. *SIAM review*, *47*(1), 135-161.

[83] Kifer, Y. (1996). Perron-Frobenius theorem, large deviations, and random perturbations in random environments. *Mathematische Zeitschrift*, *222*(4), 677-698.

[84] Newman, M.E.J. The Mathematics of Networks. In: The New Palgrave Dictionary of Economics, 2nd ed. Imprint: Palgrave Macmillan, Basingstoke, **2008**.

[85] Bullinaria, J.A. Self-Organizing Maps: Fundamentals. Available from< *http://www.cs.bham.ac.uk/~jxb/NN/l16.pdf*>; **2004** [accessed 13.06.2016]

[86] Guthikonda, S.M. Kohonen Self-Organizing Maps. Availbale from< *http://www.shy.am/wp-content/uploads/2009/01/kohonen-self-organizing-maps-shyam-guthikonda.pdf*>; **2005** [accessed 20.01.1016]

[87] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

[88] Malhotra, R., & Jain, A. (2012). Fault prediction using statistical and machine learning methods for improving software quality. *Journal of Information Processing Systems*, *8*(2), 241-262.

[89] Durgesh, K. S., & Lekha, B. (2010). Data classification using support vector machine. *Journal of Theoretical and Applied Information Technology*, *12*(1), 1-7.

[90] Lin, K. C., Chen, S. Y., & Hung, J. C. (2014). Botnet detection using support vector machines with artificial fish swarm algorithm. *Journal of Applied Mathematics.*

[91] Dabbiru, L., Aanstoos, J. V., & Younan, N. H. (2016). Earthen levee slide detection via automated analysis of synthetic aperture radar imagery. *Landslides, 13(4)*, 643-652.

[92] Beigi, E. B., Jazi, H. H., Stakhanova, N., & Ghorbani, A. A. (2014, October). Towards effective feature selection in machine learning-based botnet detection approaches. *In Communications and Network Security (CNS), 2014 IEEE Conference on* (pp. 247-255). IEEE.

[93] Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3), 357-374.

[94] Aviv, A. J., & Haeberlen, A. (2011). Challenges in experimenting with botnet detection systems.

[95] Jafari-Marandi, R., Khanzadeh, M., Smith, B. K., & Bian, L. (2017). Self-Organizing and Error Driven (SOED) Artificial Neural Network for Smarter Classifications. *Journal of Computational Design and Engineering*.

[96] Khanzadeh, M., Marandi, R. J., Tootooni, M. S., Bian, L., Smith, B., & Rao, P. Profiling and Optimizing the Geometric Accuracy of Additively Manufactured Components via Self-Organizing Map