Masaryk University
Faculty of Informatics

Selected botnet detection techniques using flow data

Master's Thesis

Tanay Bhattacharya

Brno, Fall 2016

Masaryk University
Faculty of Informatics

Selected botnet detection techniques using flow data

Master's Thesis

Tanay Bhattacharya

Brno, Fall 2016

Replace this page with a copy of the official signed thesis assignment and the copy of the Statement of an Author.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Tanay Bhattacharya

Advisor: Zdeněk Říha Ph.D.

Acknowledgement

I would like to convey my sincere gratitude towards my advisor Ing. Mgr. et Mgr. Zdeněk Říha, Ph.D. for his constant guidance during my research work. I would also like to acknowledge my sincere regards to prof. RNDr. Václav Matyáš, M.Sc., Ph.D. for his overall guidance during the entire study. At last but not least, I thank my family for all the supports during my study.

Abstract

The botnet is one of the most widespread malware in the wild aimed at coordinated compromise, control, and misutilization of vulnerable machines. These compromised machines are known as zombies/bots and the controller is often called *botmaster*. These machines are further used as launching platform to carry out coordinated attack on target system(s)/network [1].

In general, depending on the characteristics of bots, different techniques have been proposed to carry out their detection and bring down such coordinated attack by Security Researchers/Vendors. To counter this, bot writers have come up with detection evasion techniques such as Encrypted Communication, Fast Flux, Domain Generation Algorithm (DGA), migration from earlier IRC based communication to recent Peer to Peer (P2P) techniques etc. It is a catch and run game between bot writers and bot detectors and so far there seems to be no clear winner!

My research will concentrate mostly on Analysis and Implementation of selected bot detection methods that are independent of common detection evasive techniques. I will be using Flow based bot detection methodologies that generally consider OSI Layer 3 and 4 metadata without payload profiling for IRC/HTTP/P2P/DNS based botnets. Different existing botnet detection solutions will be considered and some of them will be implemented in experimental setup and tested with live bots and various other Datasets. The performance of such solutions will be evaluated. This will help to figure out optimum botnet detection parameters and techniques for different botnet types.

Keywords

Botnet, Netflow, Machine Learning, Classification, IRC/HTTP/P2P, Command and Control(C2), Anomaly based, Domain Generation Algorithm(DGA), Fast Flux.

Contents

1 Introduction **1**
- 1.1 *Thesis contribution* . . . 2
- 1.2 *Thesis organization* . . . 2

2 Existing approaches and Related Works **3**
- 2.1 *Background* . . . 3
- 2.2 *Command and Control (C2) Infrastructure* . . . 6
- 2.3 *Related Works* . . . 9
  - 2.3.1 IRC botnet detection . . . 9
  - 2.3.2 HTTP botnet detection . . . 11
  - 2.3.3 P2P botnet detection . . . 12
  - 2.3.4 DNS based botnet detection . . . 14
  - 2.3.5 Mining based botnet detection . . . 15
- 2.4 *Flow data for botnet detection* . . . 18
  - 2.4.1 Flow introduction . . . 18
  - 2.4.2 Application of Flow-based botnet detection . . . 19

3 Botnet detection setup **20**
- 3.1 *Implementation Methodology* . . . 22
  - 3.1.1 Flow setup . . . 22
  - 3.1.2 Data source . . . 22
  - 3.1.3 Selection of botnet type . . . 24
  - 3.1.4 Flow data optimization . . . 24
  - 3.1.5 Implementation Setup . . . 26
- 3.2 *Selected detection techniques for Implementation and Analysis* 28

4 Experiment and Analysis **31**
- 4.1 *P2P botnet detection* . . . 31
- 4.2 *DNS based botnet detection* . . . 42
- 4.3 *Mining based botnet detection* . . . 44
- 4.4 *Summary* . . . 58

5 Conclusion **59**

Bibliography **61**

A Appendix A **69**

B Appendix B **70**

C Appendix C **71**

D Appendix D **72**

List of Tables

2.1 Botnet attributes *5*

3.1 Benign P2P Application used *24*
3.2 Bots with centralized C2 communication *25*

4.1 Various experimental Flow characteristics *32*
4.2 P2P Detection *33*
4.3 P2P program *33*
4.4 Geographical distribution of peers *41*
4.5 DNS based bot Detection *42*
4.6 Mining based HTTP bot Detection *45*
4.7 Selected NFDUMP Parameters *48*
4.8 HTTP bot Dataset Characteristics *50*
4.9 HTTP mixed Dataset Characteristics *50*
4.10 HTTP bot detection accuracy (with Decision Tree) *52*
4.11 HTTP botnet detection: Decision Tree vs Naïve Bayes *53*
4.12 HTTP botnet detection: Decision Tree vs Naïve Bayes (modified) *56*

C.1 Feature Set: Haddadi vs Zhao *71*

D.1 ASN distribution *72*

List of Figures

2.1 Bot life cycle [11] *4*
2.2 Botnet detection techniques *7*
2.3 Abstracted Timeline and Illustration [18] *8*

3.1 Botnet detection setup *20*
3.2 Zeus builder setup *21*

4.1 P2P Avg bytes per flow between μTorrent and Zeus *34*
4.2 P2P Avg bytes per flow among benign and bot program *35*
4.3 Packet Symmetry *36*
4.4 Packet Symmetry with other P2P bot *37*
4.5 P2P Links *38*
4.6 Unique P2P Connections w.r.t time *39*
4.7 Limitation: Unique P2P Connections w.r.t time *40*
4.8 DNS traffic for P2P bot *43*
4.9 Normal DNS traffic *43*
4.10 DNS traffic for HTTP bot *45*
4.11 Machine learning setup *46*
4.12 Part of Attribute-Relation File Format(ARFF) for HTTP bot classification *49*
4.13 Mix Dataset creation setup for Mining *51*
4.14 ROC Curve:HTTP bot *54*
4.15 ROC Curve:Equal traffic ratio (without filter) *57*
4.16 Decision Tree: HTTP bot classification (without filter) *57*

1 Introduction

Botnet in recent days has become the talking point for malware researchers for its coordinated network of attack vectors spread across the cyber space [1]. They are mostly driven by commercial interests of hackers and recent trends are towards hiring of botnet for malicious intents [2]. They compromise vulnerable systems, mostly because of poorly configured devices, software vulnerabilities, careless operators etc. botnets are popular to carry out malicious activities like DDoS, Spamming, Adware, Clickjacking, Phishing etc. Security experts have developed layered security measures to counter botnet at various levels of the network. Most of them work on the principle of Signature based detection, some Anomaly based detection and some based on Data Mining techniques. Whereas, bot writers are constantly changing their mode of operation by majorly going towards encrypted communication. Because of this phenomenon, it is difficult for any traditional Signature-based detection technique to detect bot activities by applying only Deep Packet Inspection (DPI) of payloads. Hence there is a requirement to have the technique which is independent of payload analysis but rather observe the network behavior that is typical to botnets.

Flow-based botnet detection is useful in scenarios where it is not possible to use DPI techniques, user privacy is taken care of, nor there is the availability of hard coded Signatures of botnet under surveillance (e.g. Zero day bots). In addition, flow-based detection techniques are useful because it puts less overhead to overall network bandwidth but provides enough metadata for useful detection of botnets. Flow-based techniques used to generate statistical information of flows in the network along with other useful information that can be used to differentiate between benign and bot/malicious traffic. Metadata gathered from flow dataset also helps in Classification, Correlation, Clustering etc(i.e.Data Mining techniques) to segregate benign and malicious traffic.

1.1 Thesis contribution

In this thesis, I will be focusing on the recent techniques used to detect botnet activities within the network using NetFlow data. For this, both live bots and also Open malicious Datasets (along with benign traffic) will be used to check the strength and limitation of these detection techniques. This will help us in the formulation of optimum botnet detection techniques based on Flow data.

During this research work, I will mostly be using proactive approach by identifying bot(s) that are likely to be part of botnet before the actual attack begins. This will be done by extracting and analyzing flow characteristics that are similar to botnet Command and Control (C2) communication.

1.2 Thesis organization

The structure of this thesis has been categorized into following chapters. The broad topic covered in these chapters are:

Chapter 2: This is the first chapter focused at brief background note on botnet lifecycle and related works on detecting botnet based on their C2 communication channels and methods like Signature, Anomaly, and Mining based approaches will be discussed.

Chapter 3: This chapter will give details on experimental setup and implementation methodologies for Flow-based botnet detection.

Chapter 4: This chapter will discuss the results obtained based on the detection methods chosen and analysis of these results. Thereafter, optimization techniques will be discussed to put value addition to these detection methods.

chapter 5: This is the last chapter aimed to give a holistic review of the botnet detection methodologies based on the results obtained in the experiment.

2 Existing approaches and Related Works

2.1 Background

Botnet Command infrastructure [3][4][5][6] for maintaining bot network has seen lots of variation from time to time. It all started with IRC [7] channel where Command and Control (C2) server used to replay commands to bots using IRC channel. It was easy for botnet detectors to bring down such network by tracking the C2 servers. Thereafter, bot writers started using conventional HTTP channel to communicate with C2 server. The idea behind this was to look like benign network traffic and evade detection. It was hard to segregate such C2 channel. P2P network is the recent advancement in botnet technology where bots communicate with peers for sharing commands and configurations for its malicious activities. It is hard to detect such bot infrastructure as there is hardly any single point of presence (PoP) for C2 server in the wild. These peer lists are generally hard coded inside the P2P bot program. Apart from P2P, there are other methods applied by bot writers for remote communication. These are DNS based. Earlier, DNS names generated by bot program were tracked by detectors and maintaining of such blacklisted DNS names was easy for bot detectors to counter remote communication by bots (e.g DNS-based block list-DNSBL [8]). Later, Dynamic DNS (DDNS), Fast Flux techniques were used by P2P bots, where DNS names are dynamically generated using algorithm like Domain Generating Algorithm (DGA) [9][10], and DNS names are resolved with multiple unique IP addresses of C2 server (e.g. Fast Flux) to counter IP address blocking by LEAs/ISPs. Another challenging part of bot detection was the use of encryption techniques by bot writers. Encryption has been used at various levels of bot life cycle(figure 2.1). These include SSL-based communication, Usage of Digital signature, Peer Authentication using shared Symmetric Key [12] etc. Table 2.1 describes important attributes that characterize botnet.

A number of research activities have been carried out to detect bot infrastructure. Mostly these are based on bot communication channel protocols, such as IRC/HTTP/P2P etc.

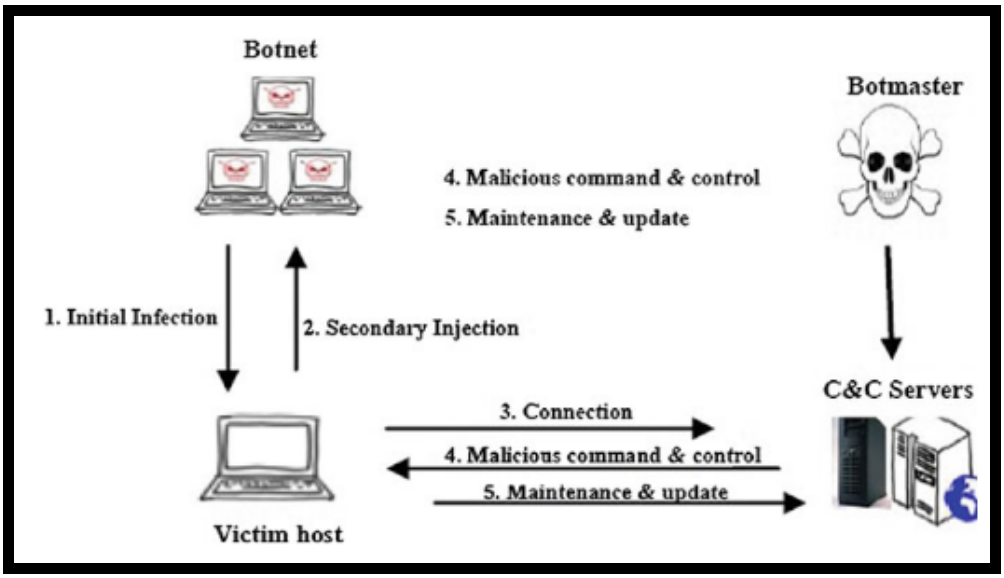

Figure 2.1: Bot life cycle [11]

Botnet Detection approaches Botnet detection approach have broadly been categorized into two parts:

- Honeynet based, and
- Intrusion detection based.

Honeynet based detection: Honeypot/Honeynet helps to fetch malicious program from wild. However, their behavior in real world scenario remains unknown. Botnet detection using honeynet suffers from issues regarding [4]:

- Limited scale of exploited activities that they can track,
- Cannot capture bots that do not use propagation methods other than those based on scanning, spam, and web driven downloads,
- Only able report information about the infected machines placed as traps.

Intrusion Detection based: Intrusion detection is a super set of many approaches utilized by modern day bot detection engines. Some

Attributes	Previous	Present
Topology	Central	Decentralized
Protocol	IRC, HTTP	P2P, DNS
Detection	Easy	Hard
Resiliency	Low	High
Anonymity	Low	High
IP List	Static/Hard coded	Static/Hard coded or Domain names
Domain Flux	Predictable/periodic	Hidden domain content/e.g.use of social media(Twitter etc)
DGA output	Random characters	Dictionary word combinations
IP Flux records	Single flux network (e.g.Strom,2007)	Double flux network (e.g.Asprov,2008)
HTTP_DIY Kit	Paid	Open_source(e.g.Zeus)
HTTP_protocol	plain text	Encrypted
Resiliency	Low	High
P2P_protocol	No-Authentication	Authentication
DNS	NIL	bot instruction/Data exfiltration

Table 2.1: Botnet attributes

of them are based on already known signatures (e.g. SNORT [13], BRO [14] etc.) and some of them are based on Host or Network level Anomalies without prior knowledge about the bot. Signature based detection technique largely depends on the already known signature of bot binaries. Hence, it is not possible for this technique to detect unknown botnets. At the same time, zero-day bot binaries are not possible to be detected by Signature based detection. Host-based Anomaly detection is based on how the malware behaves within the system. This type of detection mechanism scans the system memory/disk/Processes to see interesting activities and raise alarm accordingly. This also needs

prior knowledge on the illicit behavior of bots. Differentiation between malware (e.g. Virus, Trojan horse etc.) and botnet remains difficult to distinguish for this type of detection approach, as it does not consider network activity forensic that is essential for botnet tracking. Network Anomaly based approach to detect bots (e.g. IRC, HTTP, P2P, SMTP, DNS etc.) are quite common and a considerable amount of research work have been conducted to characterize such bots. In this type of detection method , unique features of individual types of bots are considered to generate bot detection pattern.Most of these detection patterns are:

- Time based,
- Space based,
- Flow size based, etc.

Apart from Signature and Anomaly based detection, there are Mining based detection methods, which relies on Correlation, Clustering, Classification methods to detect unknown botnets. Many Mining based approaches have been proposed either by considering only Flow data or the combination of Flow with other system parameters (e.g. System logs, IDS logs etc.). Above mentioned detection techniques proposed by researchers[4][15] are shown with the help of mind-tree (refer figure 2.2) below:

2.2 Command and Control (C2) Infrastructure

Botnets are commanded and controlled by C2 server(s) spread across the globe. The objective is to evade regulations of any particular geopolitical region and take advantage of varying rules and regulations of different countries and lack of coordination among them. Depending on the architecture of C2, botnets are further categorized as Centralized, Decentralized, Hybrid and Random [4][16]. Figure 2.3 below describes the evolution of botnets based on C2 channel protocols [17].

Centralized architecture is easy to manage and scalable. In this type of architecture, the *botmaster* has under his control C2 server(s). The main drawback of this setup is the single point of failure and low resiliency. IRC/HTTP botnets are mostly centralized. Decentral-

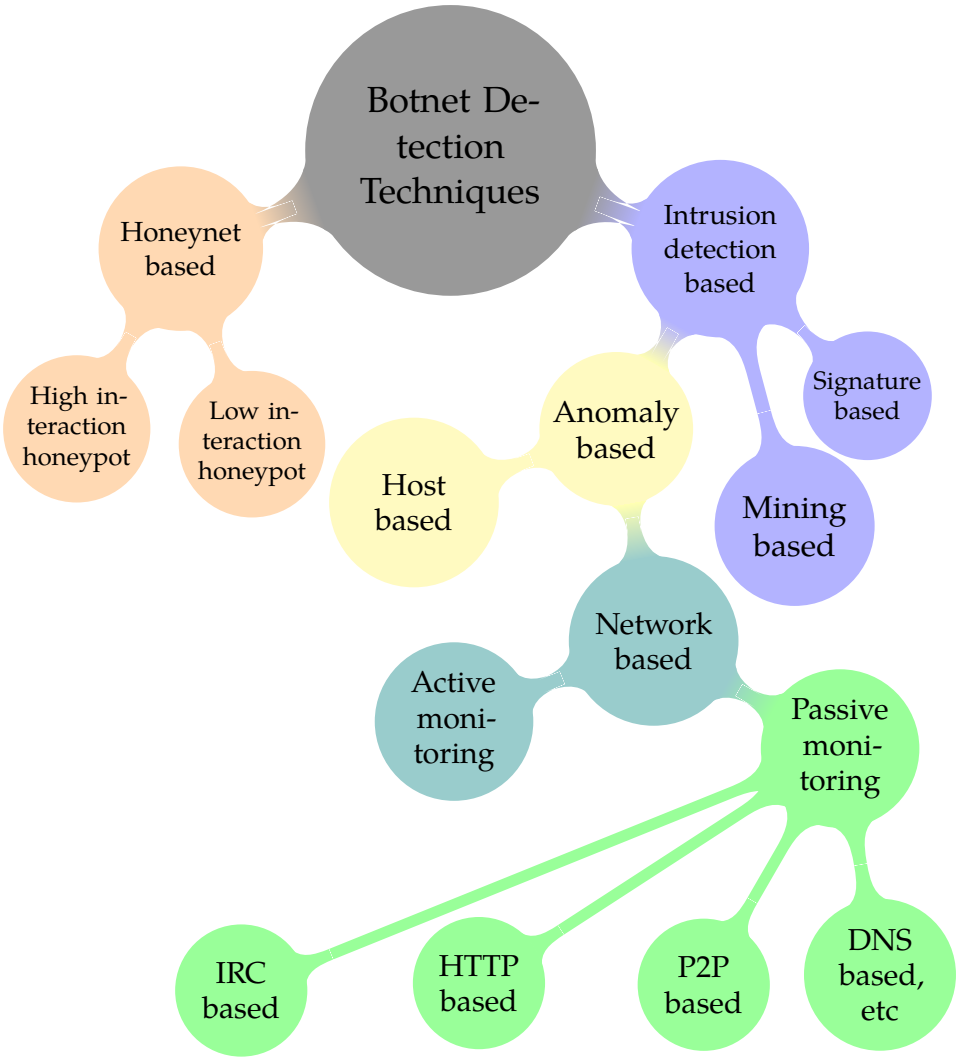

Figure 2.2: Botnet detection techniques

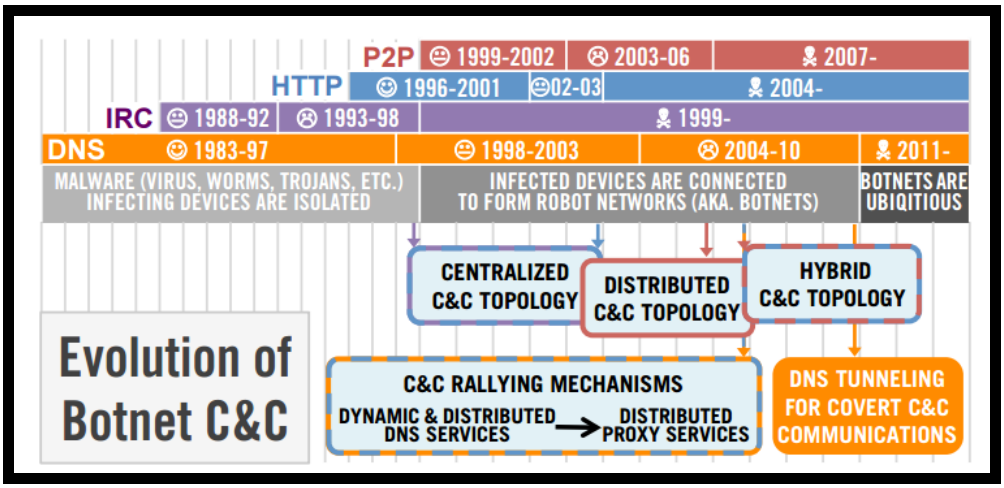

Figure 2.3: Abstracted Timeline and Illustration [18]

ized architecture helps mitigate these drawbacks with distributed C2 channels. Decentralized C2 architecture mostly uses Structured, Unstructured and Super-peer overlay networks [4].

Structured overlay network assigns keys to data items and organizes its peers into a graph that maps each data key to a peer. This structured graph enables efficient discovery of data items using the given keys [19]. Torrent is a typical example of the structured overlay network. Some structured overlay networks are Content Addressable Network (CAN) [20], Tapestry [21] etc. These type of network causes significantly higher overheads than unstructured P2P networks for popular content. Consequently, over the Internet today, the decentralized unstructured P2P overlay networks are more common [19, p. 73].

Unstructured P2P overlay networks organize peers in a random graph in a flat or hierarchical manner (e.g., Super-Peers layer) and use flooding or random walks or expanding-ring Time-To-Live (TTL) search, etc.[19, p. 82]. Some examples of unstructured P2P overlay networks are Gnutella, FastTrack, KaZaA , eDonkey etc. A Very famous application like Skype uses Super peer overlays. However, because of the exposure, these networks are more vulnerable to track and bring down by LEAs. Hence, usage of super peer overlays is more unlikely to be used by *botmasters* [4].

Hybrid P2P overlay network consists of client and server, which are used to publish and retrieve small pieces of data by creating a file-sharing network. This helps in the concurrent download of a file from multiple peers, detection of file corruption using hashing etc [19]. This architecture is based on servant and client bots [22][23]. Servant bots use static and routable IP addresses and accept incoming connections with determined ports from client bots. Client bots contain peer lists and periodically connects to the servant bots for commands. This architecture provides more resiliency, along with encryption, easy monitoring, and recovery from captured bot(s) [22].

Whereas, in the random model of C2, bots wait for the random time for commands from *botmaster*. Here, *botmaster* scans network/the internet to find its bots and issue commands. This type of C2 model is hard to detect as there is no fixed communication pattern and hence resiliency is high. However, scalability and coordination remain as challenging factors.

2.3 Related Works

The following section will focus on research works done on various methods of botnet detection in more details.

2.3.1 IRC botnet detection

Internet Relay Chat (IRC) based C2 infrastructure was the first to be used by bot writers. This command system was used to interpret commands, support, retrieve information etc. from compromised machine(s). One of the initial IRC bots was Eggdrop [7] and is still active. SDBot [24][25], Agobot [26] etc. are some more popular examples of IRC bots. IRC botnet mostly use multicast communication through groups. Multicast help such infrastructure to interact a number of bots at a time. However, unicast IRC commands between *botmaster* and bot are also possible. IRC bots initially used to have communication among themselves using 'nickname' in plain text that was easy to interrupt/detect. Another challenges for IRC-based botnet propagation was the use of uncommon protocol and hence raises alarm by network security apparatus. It also suffers from a single point of

failure. Nowadays, obfuscation of IRC message possible. Ciphering IRC commands/payloads makes it difficult for detection.

1. Binkley and Singh [27][28] combines IRC message statistics and TCP scan detection heuristic, also called TCP work weight to detect IRC-based botnet meshes. However, usage of the even trivial cipher can defeat the proposed approach.

2. Karasaridis [29] proposed botnet detection using mostly transport layer data (passive analysis using flow) for IRC botnet controller detection in Tier-1 ISP networks. This method used to detect, track and characterize IRC botnets.

3. Livadas [30][28] proposed machine leaning based classification technique to detect IRC based C2 traffic. Their work was based on two step approaches:

   - distinguishing between IRC and non-IRC traffic. In this step, identification of features that achieve good overall classification accuracy was discovered.
   - labeling flows as suspicious and non-suspicious (among IRC traffic).

4. Whereas Livadas [30] work mostly focused on classifying IRC traffic from non-IRC traffic, Strayer [31] work focused on add-on approach, that includes flows classification using machine learning techniques, then the flows that are in the IRC "chat" class are correlated to find clusters of flows that share similar timing and packet size characteristics. The cluster is then analyzed to try to identify the botnet controller host. This requires very high volume datasets for evidence of tight botnet C2 activity.

5. Goebel introduced an approach, called Rishi [32] that uses passively monitoring network traffic for unusual or suspicious IRC nicknames, IRC servers, and uncommon server ports. It uses a scoring system to detect bots that use uncommon channels that can evade IDS detection [33]. As Rishi depends on the signature, its detection capability is only dependent on bots where regular

expression exists. This means detection of encrypted traffic and non-IRC traffic is beyond Rishi's detection capability.

6. Cooke [34] tried to analyze detection techniques specific to IRC-based botnet detection. They considered IRC botnet detection based on string match in packet payload, traffic to fixed port (e.g. TCP 6667) and other high ports, honeypot based approach etc.

The above approaches only address IRC-based botnet detection. However, we can't have an assumption on botnet C2 channel and none of these methods can be used for generic botnet detection.

2.3.2 HTTP botnet detection

HTTP protocol used by botnets to evade suspicion by protocol based Anomaly detection methods and to look like normal traffic. It is hard to filter normal HTTP and bot HTTP traffic. Since IRC and HTTP both uses centralized C2 channel, hence the central point of failure is possible. A number of methods [35][36] etc. have been proposed in earlier works that are based on detection of HTTP botnet.

1. Guofei Gu [35] proposed 'Botminer' as a general detection framework independent of botnet C2 protocol and structure, and does not require any prior knowledge of botnets. Botminer performs Cross Cluster Correlation to identify the hosts that share both similar communication patterns and similar malicious activity patterns. The proposed approach relies on the principle that bots within the same botnet will exhibit similar C2 communication patterns and similar malicious activities patterns.

2. Gu [36] proposed 'BotSniffer' that uses Anomaly based detection on the local network to identify botnet C2 channels. This is done without any signature or C2 server addresses. This detection approach can identify both the C2 servers and zombies in the local network. BotSniffer captures spatial-temporal correlation in network traffic and utilizes statistical algorithms to detect botnets.

3. Perdisci [37] proposed network-level behavioral clustering of HTTP-based botnet C2 channel. They extracted detailed information from the network traces, such as the number and type of HTTP queries, the length, and structural similarities among URLs, the length of data sent and received from the HTTP server, etc.

   However, this is significantly different from Netflow based botnet detection approach and also suffers from encrypted HTTP traffic.

4. Lee's [38] work used to find malicious HTTP botnets by using the degree of periodic repeatability to access HTTP server by bot clients.

2.3.3 P2P botnet detection

IRC and HTTP based botnets use centralized C2 channel, whereas P2P botnets used decentralized C2 channel. E.g. Zeus bots have variants which use C2 channel for communicating with *botmaster* and also have variants (Zeus GameOver) that use P2P communication for regular activities like sharing of configuration files, peer list updates etc. From Figure 2.3 shown earlier, we have seen P2P based botnets are more recent development and mainly aimed at resiliency, robustness and are difficult to detect.

1. Francois [39] in their work used a setup called 'BotTrack' that is used to detect botnet using Netflow and page rank. The key concept is to analyze communication behavioral patterns among peers and to infer potential botnet activities. Linkage analysis and clustering techniques were used to group hosts sharing similar behavioral/communication patterns.

   (a) Earlier works for detection of IRC, HTTP, P2P based botnet detection were mostly based on "swarm effect". This means that detection was possible only when bot hots perform collective/group activities within a network. Hence, these detection methods cannot perform effectively in case these is single compromised host in the network and/or

the compromised hosts belong to different botnets. However, following works for P2P botnet detection are based on behavioral approach that is focused towards statistical features of the botnet that can even be applied to the botnet which has single compromised host in a network.

2. Dennis [12] work focuses on P2P protocol used by a particular family of botnet(i.e. Zeus bots).His work shows how peers communicate among themselves using UDP port (e.g. version reply, peer list request/reply, data request/reply etc.) and TCP port(for same purposes as UDP port and message exchange between harvester bots and proxy bots). They also showed how Domain Generation Algorithm (DGA) is used as backup method by P2P bots for getting signed peer lists from *botmaster*. From his work, important characteristics of P2P botnet taken are:

   - Identification of uniq characteristics of P2P botnets (i.e peers) such as IP address and UDP port,
   - Communication pattern (i.e. active thread) etc.

3. Dillon [40] work focused on detection of P2P botnet using Netflow data. His work used Flow characteristics such as Traffic Volume, Packet Symmetry, and Traffic Patters etc. to distinguish P2P botnet traffic from benign traffic. Benign and malicious P2P traffic were separated using a high number of failed connections. This method proposed detection of P2P botnet even if payload encryption is used.

4. Kheir and Wolley [41] used the behavioral approach to detecting P2P bots inside a network perimeter without considering DPI. They used Machine Learning Techniques to distinguish between malicious and benign P2P traffic. They used Time-based, Space-based, Flow-size based Training features for this.

5. David and others [15] showed that P2P botnet can be detected using Flow intervals. They compared the performance of Bayesian Network classifier and Decision Tree Classifier using reduced error pruning (REPTree).They mainly focused on Anomaly and Mining based approach for botnet detection.

2.3.4 DNS based botnet detection

DNS queries are used by botnets for various reasons. They may be for getting in touch with remote C2 Server for commands and the further injection of binaries [18], perform malicious behaviors(e.g. DDOs attack), Fast flux (when C2 server is unreachable), DGA for C2 migration etc. DNS communication also used by botnet for exfiltration of information from compromised hosts [18].Many ISPs, Anti-Virus Vendors, and Enterprise Email Vendors use Domain Name System-based Blackhole Lists (DNSBLs) to track Blacklisted IP addresses that originate spam and reject them [42][43][8].In the case of IP blocking by LEAs, change in IP address of C2 servers propagate almost immediately to bots due to short time-to-live (TTL) values for the domain names set by DDNS providers.

1. Manasrah [44] considered group activities of bot hosts by classifying DNS communication for detection of botnet within the network.

2. Choi [45] also proposed Anomaly based botnet detection by monitoring group activities in DNS traffic. This was done in following steps:

   - finding of distinguishable features between botnet and legitimate traffic,
   - defining key feature of DNS traffic (i.e. group activity) and algorithm to differentiate and analyze botnet DNS query by using group activity feature,

   However, this approach has limited detection scope when botnet uses DNS only at initializing and never use it again. Also, dormant bots can defeat this approach as they may rarely make DNS queries to remote Servers.

3. Ricardo Salomon and Jose [46] proposed two approaches for identifying botnet C2 servers based on anomalous DNS traffic. They took two-fold approach to detect DNS query based botnet detection:

- looking for domain names that have high or concentrated query rates,
- looking for abnormally recurring DNS replies indicating the query against the nonexistent name (NXDOMAIN).

However, this work considered DNS queries with low TTL values as signature of botnet communication. However, low TTL value does not mean the query belongs to bot activity only, as more and more legitimate sites are using them (e.g. DNS based load balancing). Another limitation of this approach is for smaller size botnets (having fewer bots), concentrated DNS query may not be possible.

4. Krmíček [47] describes the limitations of 'only' flow based botnet detection. However, he proposed scope of Flow-based botnet detection based on two approaches:

   - Use of local DNS server. A large number of outside DNS queries will indicate bot hosts.
   - Time of DNS query. High-density DNS query from a group of bot hosts in a network may indicate botnet activity.

5. In general, most of the existing DNS-based botnet detection approaches include frequency of dynamic domain name usages, DNS behavior similarity among bots within the same botnet inside network etc. However, all these requires access to payload information for DNS communication. Mostly the parameters are domain names, TTL values, distinct IP address in answer etc. [47]. These parameters certainly are not possible to be captured through Flow data only.

2.3.5 Mining based botnet detection

Data Mining is focused on finding out irregularities (or regularities!) in large Dataset. Earlier detection methods including Anomaly based detection used some known features of network behavior such as High Network latency, Unused Port activities etc. On the other hand, Data Mining helps in selection of characteristics that will help optimization of botnet detection. Various Data Mining approaches can be used for

botnet detection using Flow data. These are Classification, Clustering etc. Classification helps in matching new flows with earlier stored flow pattern. However, for any new botnet traffic pattern, this approach cannot be efficient to detect botnet. Whereas, Clustering helps data points to cluster with each other based on their feature values. In a cluster, there is no target variable. Cluster only implies similar looking Flows together. As per Jignesh [33], Data Mining approach is the best botnet detection method.

1. Gu [35] in their work used clustering technique to group similar communication traffic and similar malicious traffic. It then performs cross cluster correlation to identify hosts that share same communication pattern and malicious activity pattern. It claimed to detect IRC/HTTP/P2P botnets.

   However, in this work A-Plane monitoring takes into account botnet's attack phase activities including IDS signatures (SNORT). Since the focus of my work mostly towards a proactive approach to botnet detection and with the help of only Flow data, the technique adopted by Botminer does not suit our criterion. If hosts within the network are compromised with different botnets, his approach is inefficient to detect them. Also, if in a monitored network there is only a single compromised host (however this single host may belong to a larger botnet), Botminer can't detect that as a bot. Randomization of bot behaviors also diminishes this approach.

2. Flow-based botnet detection by correlating multiple log files was proposed by Masud [48][11].Their work proposed temporal correlation of Host-based log files to detect bots within the network. Data mining technique was used to extract important features from Host log files and C2 traffic.

   However, this work took into account Mining of Host log files. Therefore, only Flow based approach is not sufficient enough to implement botnet detection using this method.

3. Udaya [49] in their work considered various Data mining techniques. They recommended that Decision Tree classifier is found

to be most effective detection method for P2P bot by analyzing the Flow Intervals. Support Vector and Bayesian Network techniques were found to be effective for detecting C2 communication for HTTP and IRC bots.

4. Sajjad [28] focused towards IRC and HTTP botnet detection by clustering Flows which has similar Netflow and Attack pattern in different time windows. The principle of detection was based on the fact that bots belonging to the same botnet receive same commands at the same point in time.

   However, the approach considered was ineffective in case there is single bot within the network. This method is also ineffective where bots remain in a dormant stage for a long time before they get activated. In this case, long Flow traces would be required, which may not be possible for a large Enterprise network.

5. Amini [50] in his work proposed two-step approach to detect C2 botnet. First, the collection of Flows and then cluster them as per pattern, policies. The second step generates another cluster based on Events, Alerts and Activities of Network Security Sensors.

   This work took into account various security sensors logs/ Alerts in addition to Flow data. Hence only Flow based botnet detection is not possible with this method.

6. Haddadi's [51] work was focused towards using Classifications technique to identify bot traffic. She showed that Decision Tree classifier provides better detection rate than Naïve Bayes classifier. Her work focused towards classification of HTTP botnet traffic.

   However, generation of Dataset was done based on script. This provides lack of visibility for real world like situation of botnet traffic and benign traffic. Dataset generated with live bot or open Dataset from other research initiatives need to be tested on Haddadi's method of botnet detection.

7. Zhao [15] utilized Naïve Bayes classifier to effectively detect P2P botnet. He described limitations of other botnet detection method that relies on reactive approach (i.e. detection of bot

activities when botnet performs malicious activities) and group behavior of bots within the network (i.e. not considering the situation where single bot machine is present in the network).

2.4 Flow data for botnet detection

A number of Flow-based bot detection methods have been proposed in [52][15][40][49][28][53][50][54]. Flow based detection is a passive approach to tap OSI Layer 3 and 4 meta data for analysis of network traffic. It is also useful for bot detection as it does not take into consideration the encrypted payload during bot communication.

2.4.1 Flow introduction

Flow summarizes the network traffic by looking into the packet headers. The most common way to denote a flow in the network is by using 5 properties from Network and Transport layers. These are Source IP address, Destination IP address, Source port number, Destination port number, and Transport layer protocol.Apart from these, characteristics such as Bytes per packet, Bits per second, In/Out packets, In/out Bytes, Flow duration, Flow interval etc. are also taken into consideration for detection of botnet communication. A flow record gets generated when Flow terminates normally (e.g. FIN bit set/session based),OR Flow monitor device does not see the packet in 'inactive timeout' [55] time OR flow monitor device sees active session for more than 'active timeout' time OR the monitoring device's cache gets full, etc. Cisco introduced Netflow [56] to collect and aggregate IP traffic information to Flow Collector. However, there are other vendors who provide different flavors of flow data, e.g Jflow [57] for Juniper Networks, NetStream [58] for Huawei Technologies, Rflow [59] for Ericsson etc. Flow Collector uses this Flow information to analyses various properties of network behaviors and detect abnormalities. IETF has standardized Flow export information and named it IPFIX (RFC 5101)[60]. Flow-based approach uses three basic stages for overall functioning:

- Flow exporter
- Flow collector

- Flow analyzer.

2.4.2 Application of Flow-based botnet detection

Flow-based botnet detection has significant advantages over conventional Signature based detection and Honeynet approach. Utilization and limitations of Signature and Honeynet based botnet detection approaches have already been mentioned earlier. There could be two approaches to detect botnet activity in the network. It can be a reactive approach, i.e. when botnet is Active and in actual attack phase or it can be a Proactive approach. During this research work, I have adopted a Proactive approach to detect botnet within the network by identifying host(s) that are likely to be part of botnet before the attack takes place by extracting and analyzing Flow characteristics that match botnet communications.

Major benefits of Flow based network analysis consists of:

- Flow only takes into account layer 3 and 4 information. Hence privacy issue is taken care of,
- Less additional network overhead caused by Flow export,
- Encrypted botnet communication does not affect efficiency.

3 Botnet detection setup

In this chapter I am going to explain the overall implementation approach that will be utilized to generate/replay both malicious and benign traffic and monitor/analysis the network behavior using various methods. I have used bot-infected machines in the virtual environment(VMware) and the setup is connected to the internet using NAT. Refer to figure 3.1.

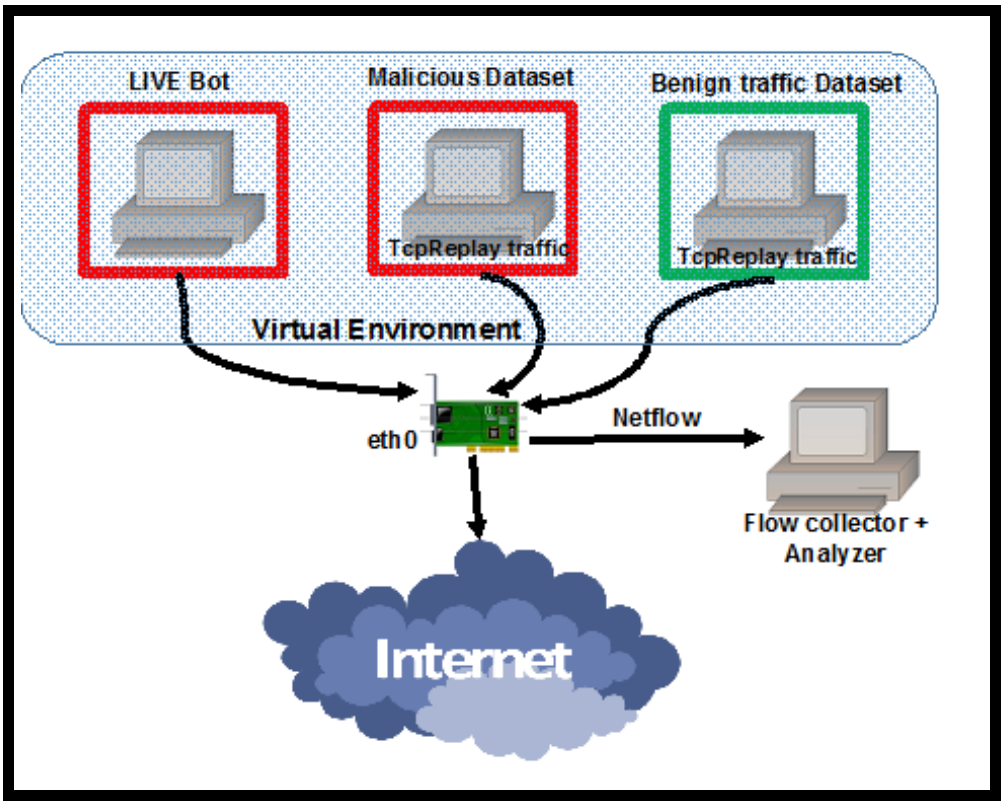

Figure 3.1: Botnet detection setup

Many of the earlier researchers mostly focused on a closed network for botnet activities detection (without real traffic to/from the internet). I connected the virtual setup to connect to the internet to realize real world like a situation where live bots can communicate with remote

C2 servers using IRC/HTTP protocols, make DNS queries, connect with peers etc.

Zeus bots with centralized C2 communication come up with bot builder kit as shown in figure 3.2. With this, we can build Zeus HTTP

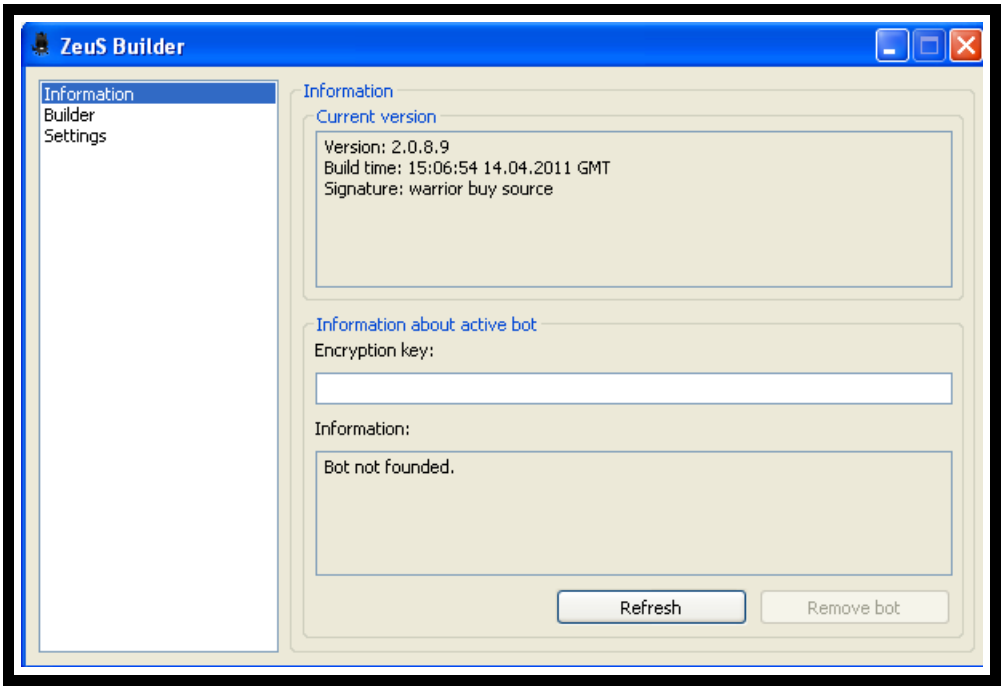

Figure 3.2: Zeus builder setup

bots. Also it comes up with configurable C2 server setup that is used to track botnet state and sending of commands to bots [49][61]. However, this method of botnet setup and thereafter detection within simulated network lacks visibility of real life botnet activities where communication happens through the internet. This method is also restrictive when dealing with botnets that require P2P communication, make repetitive DNS queries to outside world etc.

Connecting virtual setup with the internet would help create flows that give footprints of real life like situation. For P2P botnets communication, connection with the internet provides useful information regarding patterns of peer connection, Packet Symmetry calculation etc. As the virtual hosts are beyond NAT, it is not possible for the remote host to initiate a new connection with internal hosts. Hence,

the communications monitored in the experiment are connections initiated from inside hosts only. However, connecting virtual machines to the internet come up with a risk of spreading of malware within the internal network. The virtual environment, in this case, comes useful as it restricts the bot to spread to the Host machine. However, it was ensured that the Host has updated Antivirus definitions installed and Firewall is turned on.

3.1 Implementation Methodology

3.1.1 Flow setup

Fprobe [62][63] has been used for the most part of the research work. Many of the referred works for botnet detection where flow data have been considered had used Netflow version5. However, Fprobe supports up to version 7, the default is version 5. Also, some part of the work (especially for P2P botnet), Softflowd [64] has been used. This was used with version 9. Softflowd supports IPv6 and can replay pcap dump directly. Fprobe and Softflowd both are open source Netflow exporter that exports flow data from device interface to Flow Collector.

Nfcapd [65] has been used to collect Flows from Exporter and dump into the local repository. Nfcapd is netflow capture daemon of the Nfdump [66][67] tools. Nfdump is open source, easy to implement and supports various versions of Netflow (v5, v7, v9). Nfdump with '-B' option has been opted (for P2P botnet detection setup) to aggregate Netflow records as bidirectional flows. This is done on connection level by taking the 5-tuple protocol, srcip, dstip, srcport, and dstport, or the reverse order for the corresponding connection flow. Input and output packets/bytes are counted and reported separately [66].

3.1.2 Data source

In this work, I have taken four approaches for managing Datasets that will be utilized for traffic generation in a simulated environment. These are:

- Traffic generated by Live bots [68][69],
- Malicious Traffic gathered from Open Data sets:

  - Malware Capture Facility Project,CTU University, Czech Republic [70]
  - Contagio dump [71],
  - Sourcefire Vulnerability Research Team Labs [72],
  - DeepEnd Research [73].
  - NETRESEC Public files [74]
  - Open Security Research [75], used for DNS based bot traffic (using DGA).

- Benign Data sets generated by accessing Internet services

  - E.g. Manually accessing internet services like web access, online music, YouTube videos.
  - manually executing benign application to generate Flow data. Refer Table 3.1.

- Benign Datasets gathered from open data archive available from various research initiatives.

  - open data sets from Lawrence Berkeley National Laboratory(LBNL) [76] .After analysis of the Dataset, it was found that it contains packet header traces of LBNL's internal enterprise traffic. However, it has limitations like no normal DNS, P2P traffic.
  - Normal capture of traffic in a home network done by CTU. For privacy reason, the pcap file only includes the DNS traffic. Activities done includes Music streaming from 20songstogo.com, Gmail, Twitter, Jitsi chat connected to gtalk, SIP and CVUT jabber, normal webs with chrome etc. Capture duration: 1.9 hours [77].
  - This is a normal P2P capture by CTU from a Linux notebook in a home network. System IP: 10.0.0.46.Activities were: Use the Deluge P2P program from Linux to download some large files. Navigate some web pages, including Twitter and YouTube [78].

S.No	Torrent binary name	SHA256 Hash	Size
1	μTorrent	3e676378b6d39db37ec48 9da53777502a3f18663a9 8f0ea78f92aa72b8fa580c	2.3 MB
2	frostwire-6.3.5.windows.exe	be1a4793f73457bee89f5 dab088f3b0f605540ee19 fcff942180a9a7ae4f0967	24.6 MB
3	Skype 7.18.85.112.exe	0d750ed7976b448f59636 121641404ae725f899255e 2cb4951bac3547b07ba6e	42.70 MB

Table 3.1: Benign P2P Application used

3.1.3 Selection of botnet type

For botnet detection, I have used bots that have property of using both Centralized and Distributed communication channel. In this regard, I opted for Zeus variants and SpyEye variants, because some binaries of it use centralized C2 channel (both Zeus and SpyEye bots) and some uses P2P channel (e.g. Zeus Gameover) [12].Zeus and SpeEye were categorically chosen during this research activity because of their popularity in creating havoc financial implications. As per Wikipedia, Zeus carry out many malicious and criminal tasks including stealing banking information using techniques like keystroke logging and form grabbing etc [79]. Zeus further installs CryptoLocker Ransomware for further malicious purpose. Gameover ZeuS is successor of Zeus with feature like encrypted P2P communication. SpeEye too applies similar techniques like Zeus with additional feature called "Kill Zeus" [80]. Table 3.2 below shows the bots that uses centralized C2 channel used during research. These bots were used to compromise the virtual host. Network activities of these bots were observed with Flow data.

3.1.4 Flow data optimization

In real world scenario, it is normal to have networks where there are host(s) that is/are compromised with one or multiple botnets along with host(s) that is/are normal (not compromised). Thus, the

S.No	Zeus binary name	SHA256 Hash	Size
1	ZeuS_1	739b90a457df9f5976023a 0843241ec53944a70224d 1e2818c4cc134b00323f1	138.0 KB
2	ZeuS_2	352439d5b2f326b25eed2 160d03b9f7e4007de2830 37c4951cb9675f41bef832	138.5 KB
3	ZeuS_3	f4a1adf7c80c7d37fc3b8 14490e92f0eb079475c6e 4f00cb2834bbc16fa0350f	165.5 KB
4	SpyEye_1	1a2d9d93d8a2560cacfc4 a1b09f91a60570c02ec7b 99ecaab5d5db626498483	35.1 KB
5	SpyEye_2	3c39f3a5de207b0500ef9 0c649e3452017a1e98d5d 0a62605d58a9366b858e0	89.0 KB
6	SpyEye_3	035c51e3d393f0570927 16C230979cd3b3882f0c 5368e6f0af876105fb894ce	196.0 KB

Table 3.2: Bots with centralized C2 communication

aggregated traffic flowing through gateway devices like routers are a combination of benign as well as malicious traffic. Flow data consisting of both of these types of traffic in a large network is computationally intensive for botnet analyzers to handle. This especially becomes very much challenging for Enterprise networks where data volume and rate both are high. Hence it is important to only consider data of importance (i.e. as close as possible to the target host/subnet) that could be analyzed for botnet detection. Hence, some initial filtering has to be at place. Depending on the type of botnet we are interested in, a number of proposals have been discussed on methodologies to filter out mostly the malicious traffic from aggregated traffic which is a mixture of both malicious and benign traffic. However, it is very important to mention that filtering needs quantitative statistical information and

human judgment. Some typical examples that have been followed in my setup are:

- Only TCP (e.g. IRC, HTTP botnet) or UDP (e.g. P2P botnet) based flows,
- when considering the only C2 flows (in Proactive approach), we can remove port scanning activities, i.e. flows having only SYN or RST flags set. However, SYN/RST exchanges indicate some malicious behaviors, but they don't much have an effect on characterizing botnet C2 flows,
- removing large bytes per packets can filter out bulk traffic based on the principle that botnet does not have large size packets for C2 flows.

These filters would assist in resulting small volume flows with high probable botnet/malicious activity in comparison to entire aggregated traffic within a network. With this approach, processing complexity for efficient botnet detection decreases.

3.1.5 Implementation Setup

Basic introduction about the experimental setup had already been provided at the beginning of chapter 3 and expressed through Figure 3.1. The detection setup has mostly been established in the virtual environment. Three virtual hosts are used to generate/replay interesting traffic for botnet detection experiment. One of them having Windows XP OS (service pack2) and other two are having Linux (Ubuntu 64 bit, version 14.04.2). The windows virtual host has repeatedly been compromised by executing bots (e.g Zeus C2, SpyEye etc). The virtual machine 'Screenshots' were used to store and retrieve images of virtual hosts as and when required. This was required to have host compromised with specific bots. Second virtual host (Linux) was used to replay malicious Datasets available from various research projects as mentioned in para 3.1.2. These Datasets are mostly available in Pcap format. Hence TcpReplay [81][82] was used to replay the traffic. Third virtual host (Linux) was used to replay benign traffic. This was required to check for the differentiating characteristics of botnet traffic from benign traffic. Flows from both regular internet activities

and open Dataset from the internet were used for this purpose. Flow capture was done on eth0 of Host machine (Windows 7 Professional, Service pack 1). Four types of flows were generated and captured at eth0 interface:

- Flows from only Windows virtual host with bot compromised,
- Flows from only first Linux virtual host with replay of malicious bot traffic,
- Flows from only second Linux virtual host with replay of benign traffic,
- Mixed flows from compromised Windows virtual host (OR, from second Linux virtual host with the replay of malicious traffic) along with benign flows from third Linux virtual host.

The fourth type of flow setup was required to realize real life like a situation where network gateway handles both malicious and benign traffic. The filtering technique as mentioned in para 3.1.4 would be used on the fourth type of flow setup to filter out unnecessary flows from mixed flows. However, optimum filtering parameters to select only malicious flows from a mixture of normal and malicious flow is still challenging.

Initially, Zeus and SpyEye with the C2 channel of communication were used to compromise the first virtual machine (Windows XP) separately. As per Table 3.2, Zeus_1 to Zeus_3 and SpyEye_1 to SpyEye_3 were used. However, the VM host was compromised with individual bots. Changing from one bot to another was done by revert back to original VM image with the help of 'Screenshot'. All the samples of Zeus and SpyEye bots use HTTP for Central C2 channel communication. The importance of HTTP-based protocol for C2 communication has already been discussed earlier in this thesis and hence was ideal for consideration in my experiment. When manually executed, samples of Zeus and SpyEye started communicating with remote C2 Server. The network traffic was captured using Flow capture module and the same was exported to Flow Collector. Thereafter Analysis of Flows was carried out using Nfdump. The same way, traffic from benign P2P program were generated and analyzed.

Cases where the live program was not available, Pcap dump file of relevant bots (e.g. P2P bot)/benign Program (e.g. normal traffic) were

used to generate network traffic using TcpReplay. And, same Flow Analysis method was applied on that traffic.

3.2 Selected detection techniques for Implementation and Analysis

1. P2P botnet detection:

   Selected Techniques: To implement P2P botnet detection, I have considered Work done by Dillon [40], Dennis [12], Kheir and Wolley [41].

   Dataset used: In this regard, I have used dataset of Zeus Gameover. However, instead of Live bot, Pcap generated by such bot [71] was used as representative of P2P botnet traffic. I have also considered Open Dataset of CTU [78]. Some benign P2P traffic were generated by execution of μTorrent [83] and Frostwire [84].

   Reason of choosing P2P botnet detection: P2P botnet was categorically chosen for consideration because these types of bots mostly do not have central C2 communication and hence highly resilient. Also relatively limited research works done in comparison to traditional IRC and HTTP based botnet detection.

   Salient points of consideration from selected technique: Dillon [40] considered unique P2P Flow features like Bytes per Flow, Packet Symmetry etc. Details on these are discussed in chapter 4.1. Dennis [12] describes unique characteristics of P2P bot like Host IP Address and Source UDP port. His work also describes method of remote communication to Peer IPs by bot host. Kheir [41] and Yen [85] considered features of P2P botnet that include space-based feature, i.e. how peers interact with each other.

2. DNS based botnet detection:

   Selected Technique: Work done by Soniya [86], Dennis [12], Krmíček [47] were considered for implementation of effective DNS based botnet detection in my setup.

   Dataset used: For DNS based botnet detection, I will be using Malicious Dataset from Contagio, NETRESEC [74], Open Secu-

rity Research [75], Live bots [68], [69]. I will also consider normal DNS behavior using Open normal Dataset from CTU [78]

Reason of choosing DNS based botnet detection: In comparison to conventional DNS requests (which is mostly random in nature), botnet DNS queries show very interesting pattern and relatively less research works published in this direction.

Salient points of consideration from selected technique: Dennis [12] describes unique characteristics of DNS-based bot behavior like usage of Domain Generation Algorithm (DGA) and repeating pattern of DNS queries. Soniya [86] described repeating DNS request pattern for bot detection. Krmíček [47] describes the limitations of 'only' flow based botnet detection. One of his proposed detection methods like high-density DNS query from the group of bot hosts in a network has been considered (this is similar to DGA property).

3. Mining based botnet detection

   Selected Technique: I have mostly considered work done by Haddadi [51] for Machine Learning based botnet classification. Apart from this, I have also considered work of Udaya [49] and Zhao [15] for references on classification algorithm selection.

   Dataset used: I have used Datasets for Mining based HTTP bot detection, such as Zeus bot [68], Citadel bot [71], Sogou bot [70]. Benign traffic Dataset has been used from Lawrence Berkeley National Laboratory [76].

   Reason of choosing Mining based botnet detection: Mining based detection approach becomes crucial when distinguishable pattern of botnet traffic is not available and therefore segregating bot/malicious traffic from a pool of mixed traffic is difficult. In particular, I will be considering Machine learning based traffic classification technique for botnet detection.

   Salient points of consideration from selected technique: Haddadi [51] and Zhao [15] proposed that Decision Tree Algorithm is suitable for HTTP and P2P based botnet detection respectively. Haddadi compared Decision Tree and Naïve Bayes Classi-

fication algorithms using Performance metric like True Positive Rate(TPR), False Positive Rate (FPR) etc. for detecting classification accuracy. However, as per Udaya [49],Decision Tree classifier for P2P botnet detection and Bayesian network classifier for HTTP,IRC botnet detection is effective. Hence, though i will be mostly considering approach adopted by Haddadi [51], but i will also consider comparison of the result of Decision Tree with Naïve Bayes classifier to check their suitability on various Datasets.
Decision Tree and Naïve Bayes classifiers are very popular among various classifiers used. Brief description of Decision Tree classifier and Naïve Bayes classifier is given in Appendix A. Other important parameters for selection of particular classification technique includes Detection Rate (DR), True Positive Rate (TPR) and False Positive Rate (FPR) etc. Brief on these are given in Appendix B.

4 Experiment and Analysis

In this research work, I particularly focused on botnets with Centralized C2 channel communication and botnets with Decentralized communication. Various detection techniques as mentioned in chapter 2 have been considered that relies mostly on the principle of:

- Time based (e.g. repeating patterns)
- Flow size based (e.g. Avg bytes per flow, Packets per second, etc.)
- Space-based (e.g. P2P interaction pattern)
- Classification based (e.g. how test data is close to already trained data) etc.

To conduct the experiment, as mentioned in para 3.1.5, network traffic was generated/replayed within the virtual environment. Table 4.1 shows various program-specific flows generated by this approach.

4.1 P2P botnet detection

Table 4.2 describes the selected P2P bot detection techniques considered during the research.

As per Dennis [12], P2P program is uniquely identified by their IP and UDP port combinations. Be it benign P2P program (e.g. μTorrent) or Zeus Gameover bot, they all communicate with their remote peers using fixed UDP Source port. Table 4.3 shows the P2P program considered for research. Unique features of P2P bot in comparison to benign application need to find out. However, before that we need to find out features of network flows that differentiate Normal traffic and P2P traffic. Here, P2P traffic means both normal P2P and malicious P2P traffic. There are various methods to find the uniqueness of P2P traffic characteristics. The unique feature is huge number of failed connections made by the internal host(s) to outside world. As per Yen [85], this phenomenon is used to separate P2P traffic from normal traffic. In this case, the internal host (inside NAT) would request for data from remote host/peer but receives empty response from outside peer. This characteristic also known as Peer churn is almost similar among P2P

S.No	Type	Duration (in mins)	No of Flows	Bytes	Packets
1	Zeus_1 [68]	60	1606	974622	9072
2	Zeus_2 [68]	60	1411	1.5 M	7437
3	Zeus_3 [68]	60	300	1.0 M	7899
4	Zeus Gameover [71]	45	39	396129	1024
5	SpyEye_1 [69]	60	74	90520	889
6	SpyEye_2 [69]	60	105	93467	853
7	SpyEye_3 [69]	60	420	330890	3122
8	NSIS.ay (P2P) [70]	60	5996	4.0 M	22323
9	Benign μTorrent [83]	60	5528	116.2 M	192119
10	Benign Frostwire [84]	30	1399	32.9 M	97520
11	Benign Skype [87]	30	291	143330	2337
12	Benign normal DNS [77]	120	7058	1.2 M	11868
13	Benign Normal P2P [78]	25	9815	794.6 M	928310
14	Benign Normal HTTP [76]	15	27404	683.4 M	987386
15	Benign Internet Traffic	60	12391	201.8 M	262597

Table 4.1: Various experimental Flow characteristics

Selected method	Key aspets	Dataset used
Dillon [40]	Bytes per Flow Packet Symmetry	Zeus Gameover [71] CTU-Normal [78]
Dennis [12]	SrcIP:UDP Port	CTU-13 [70]
Kheir Wolley [41]	Space-based feature	μTorrent [83]
Yen [85]	Peer churn	Frostwire [84].

Table 4.2: P2P Detection

benign and P2P bot program. It refers to the dynamics of peers joining and leaving P2P network. This phenomenon is often reflected as a high ratio of failed connections observed in P2P networks.Once, P2P traffic is segregated from normal traffic, the further experiment needs to be carried out to separate normal P2P traffic and bot P2P traffic. In this regard, findings of the experiment when implemented according to detection technique proposed by Dillon [40] are mentioned below.

S.No	Type	Src IPAddress	Port type
1	Zeus_Gameover	172.29.0.116	23575/UDP
2	μTorrent	192.168.44.135	37966/UDP
3	Frostwire	192.168.44.135	1038/UDP
4	Skype	192.168.1.102	9396/UDP
5	NSIS.ay	147.32.84.165	32234/UDP

Table 4.3: P2P program

1. Bytes per flow
   It is important to note that the total bytes communicated between peers in case of benign program are much higher than bot binaries. This is because benign P2P program mostly shares among peers high volume data/files in comparison to P2P bot peers. However, Flow characteristics such as Byte per flow, Byte per Packet etc. play an important role in differentiating between benign and bot Flows [40][12].
   As per work of Dillon [40], average bytes per flow is one of the differentiating features of P2P bot and benign program. When average bytes per flow for both P2P bot and μTorrent program

were compared, it was found that there is a difference in ranges of bytes per flow for benign and bot P2P program. This is shown in figure 4.1 below.

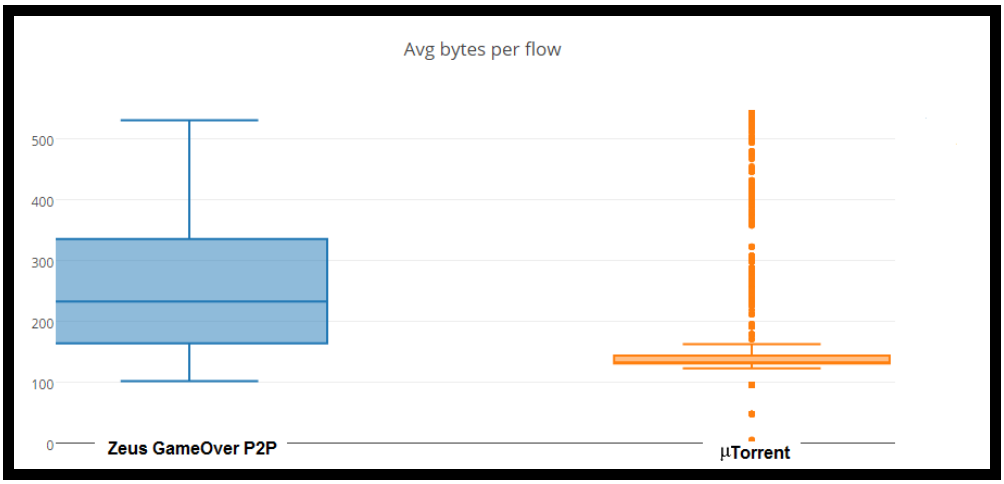

Figure 4.1: P2P Avg bytes per flow between μTorrent and Zeus

(a) Limitation:
However, avg bytes per flow may not be unique characteristics that separate all types of P2P benign and P2P bot flows. When tested with other benign P2P program, it was found that there may be other program (e.g Frostwire), where range of bytes per flow is close to the range of Zeus or NSIS bot flow data. This make unique identification characteristics (i.e. avg bytes per flow) of P2P bots in comparison to benign P2P challenging. This is shown in figure 4.2 below.

2. Packet symmetry
In the case of Bytes per flow, it was seen that even if there is distinct characteristics difference between P2P bot and certain benign program (i.e.μTorrent), there may be cases where the ranges of both are not clearly distinct (e.g. avg bytes per flow between Zeus Gameover and Frostwire).
As per Dillon [40], Packet Symmetry is another important distinguishable characteristics to separate P2P bot and P2P benign flows. Packet Symmetry is the ration between outgoing packets

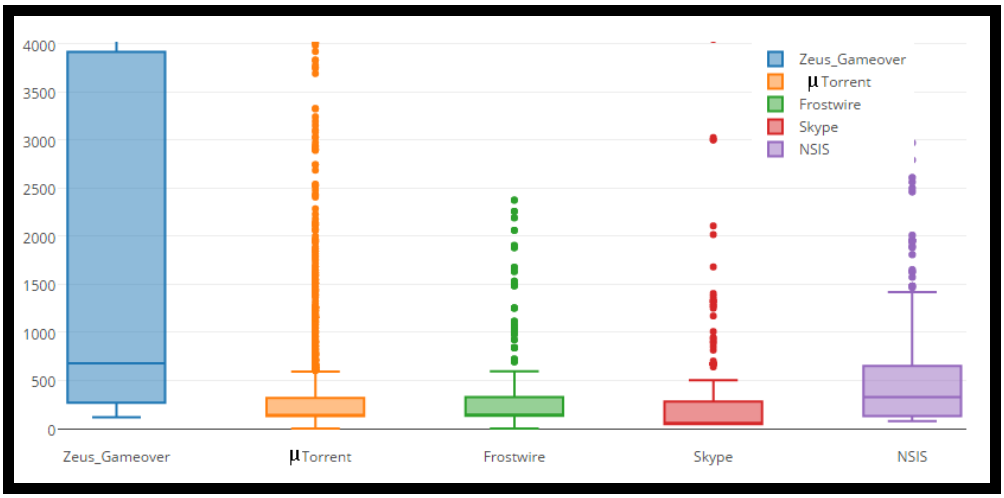

Figure 4.2: P2P Avg bytes per flow among benign and bot program

and incoming packets. This is very useful in detecting DDoS type attacks and also in detecting malicious traffic [40]. This phenomenon has been tested in my experimental setup with various Datasets. Experiment results show that, in most of the benign P2P applications, the ratio of outgoing packet vs incoming packets are more than equal to 1.0. Whereas, for P2P bots like Zeus Gameover, the ratio is somewhere in the range 0.3 to 0.4 (refer figure 4.3).

This scenario is explained by the phenomenon that for Zeus P2P, a lot of packets are received against single outgoing packet request [40].It happens because, for Zeus P2P, the Peer list request payload size is less than Peer list reply payload size. As per Dennis [12], the Peer list reply consists of 10 peers from the responding peer's peer list which are closest to the requested identifier.

It is also to be noted that this has been tested behind NAT, hence it is assumed that no remote peers could initiate connection requests to the inside host. To get the parameters out of Netflow data, I have considered Bi-directional Flows and was able to get parameters like in/out packets and in/out bytes.

(a) Limitation:

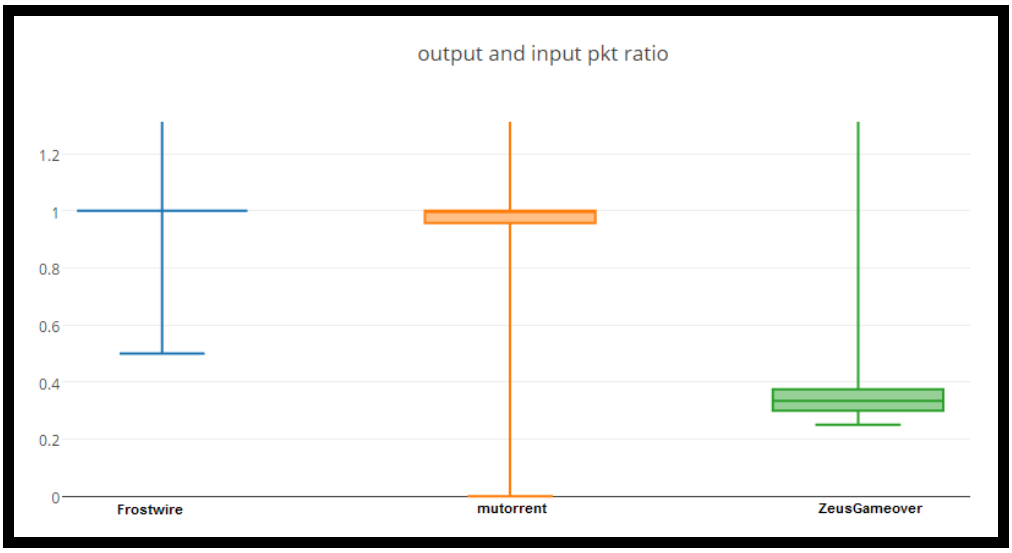

Figure 4.3: Packet Symmetry

Dillon's [40] detection criterion was majorly focused towards detection of Zeus bots with the help of characteristics like avg Bytes per Flow, Packet Symmetry and Periodic connection with remote peers. Packet Symmetry was considered to be important characteristics of P2P bots. However, the parameters chosen by him in the detection of P2P botnet does not fully fit into for all P2P bot detection. This has been tested with NSIS.ay P2P bot using CTU Dataset [70]. The outcome is shown in Figure 4.4.
CTU Dataset-53 [70] that contains NSIS.ay bot(P2P) traffic has been considered. The Pcap file as provided by CTU is 281 MB size. The flow parameters of this Pcap, when replayed using 'TcpReplay', has already been shared in Table 4.1. The CTU Dataset was generated by infecting three virtual machines (147.32.84.165, 147.32.84.191 and 147.32.84.192) in controlled network for approx. 1 Hr. In figure 4.4, I have considered Packet Symmetry against one of the infected machines (i.e.IP 147.32.84.165). During this experiment, a visible difference has been noticed in comparison to Zeus bot characteristics (compare Figures 4.3 and

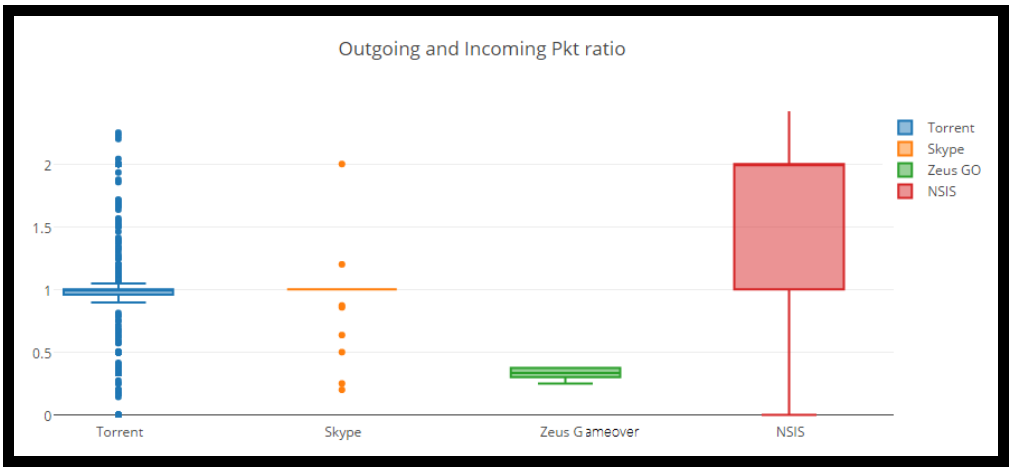

Figure 4.4: Packet Symmetry with other P2P bot

4.4). As the Packet Symmetry value falls within the range of other benign P2P programs (e.g. μTorrent), hence it can be concluded that Packet Symmetry property is not a generic distinguishable characteristic of P2P bot. There may be similar bots like NSIS that might have similar characteristics also.

3. Spatical distribution
   Kheir and Wolley [41] observed that P2P benign program tries to make much more unique peer connections in comparison to P2P bots. This means that P2P bots have lower chunk rate than P2P benign program. Torrents make more peer connections to share data/files among peers. These peers are distributed geographically all over the world. More the peers, much faster is the possibility to download files within torrent clients. However, this phenomenon is limited in P2P bots. In these types of bots, the bot host contact a limited number of peers for sharing information. During experiment, it was found that the number of remote peer connection attempts made by P2P benign and P2P bot are 3280 (within 60 mins) and 32 (within 45 mins) respectively. This clearly distinguishes between benign P2P and

bot P2P program. This is shown in figure 4.5 below (Algorithm used: Force-directed graph [88]).

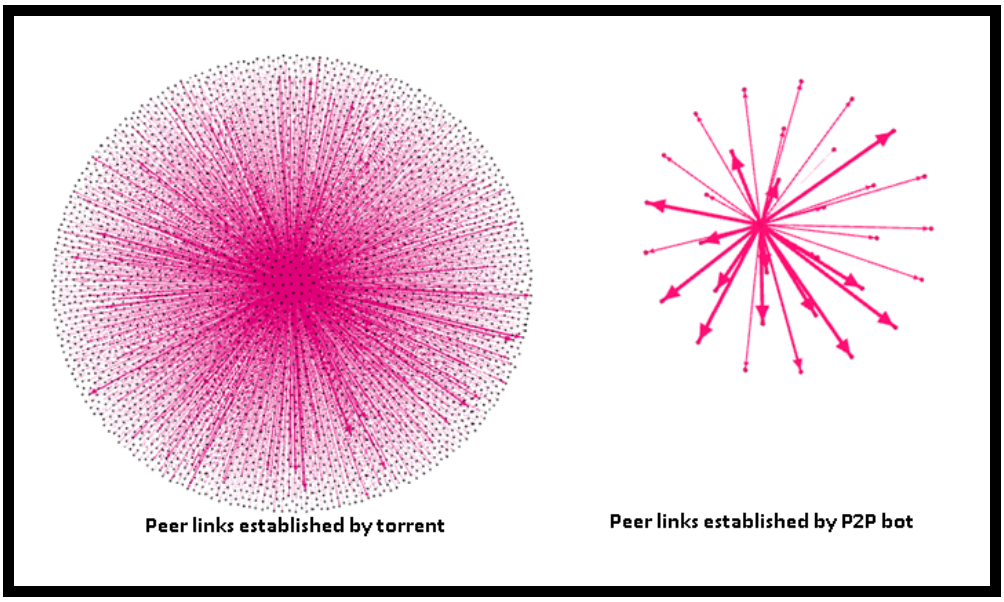

Figure 4.5: P2P Links

However, when compared with benign P2P application like Skype, this difference was not as large as μTorrent and it was noticed that remote connections made by Skype client were 139 (within 30 mins period).

(a) Peer Churn

As per Yen [85], Peer Churn is another distinguishable characteristics of P2P program. It is the characteristics of P2P program where peers join and leave the network after doing their job. Benign P2P program has high churn as the peers share files among themselves and leave the group. Whereas for P2P bots, churn is low because these bot peers share commands/updates among other peers and maintain the connection for a relatively long period. When compared with Zeus Gameover Dataset, we could see that μTorrent went on keeping regular unique contacts with its remote peers. But w.r.t time, P2P Zeus stopped making new contacts much early. This reflects that Torrent peers join and

leave the file sharing network at a very fast rate and Zeus P2P peers restrict themselves with limited peers. Higher the rate of new contacts, higher the churn. This is shown in figure 4.6 below.

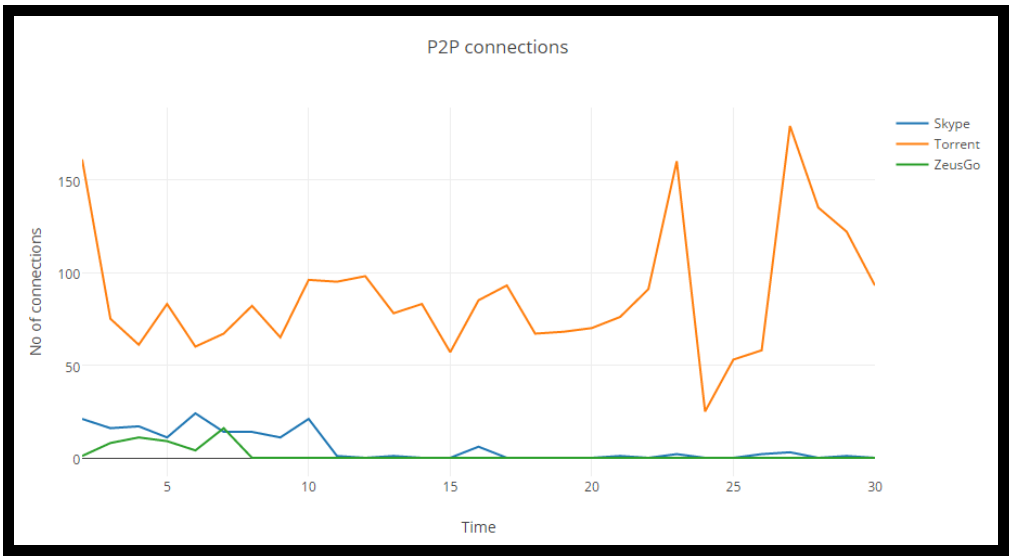

Figure 4.6: Unique P2P Connections w.r.t time

i. Limitation: However, Peer churn helps segregate P2P program from a benign program, there may be other P2P bots (unlike ZeusGameover) that show very similar Peer churn characteristics like other benign P2P programs. An example is shown in Figure 4.7 below where NSIS.ay P2P bot has higher Peer churn than Frostwire benign P2P program.

(b) Geographic Distribution

Kheir and Xiao [89] used the Geographic distribution of P2P bot peers to filter out non-P2P bot. They used the existence of distinct Autonomous System (AS) numbers within bot flow in order to discard non-P2P bots. This is shown when tested with various P2P bots and non-P2P bots (refer Table D.1).

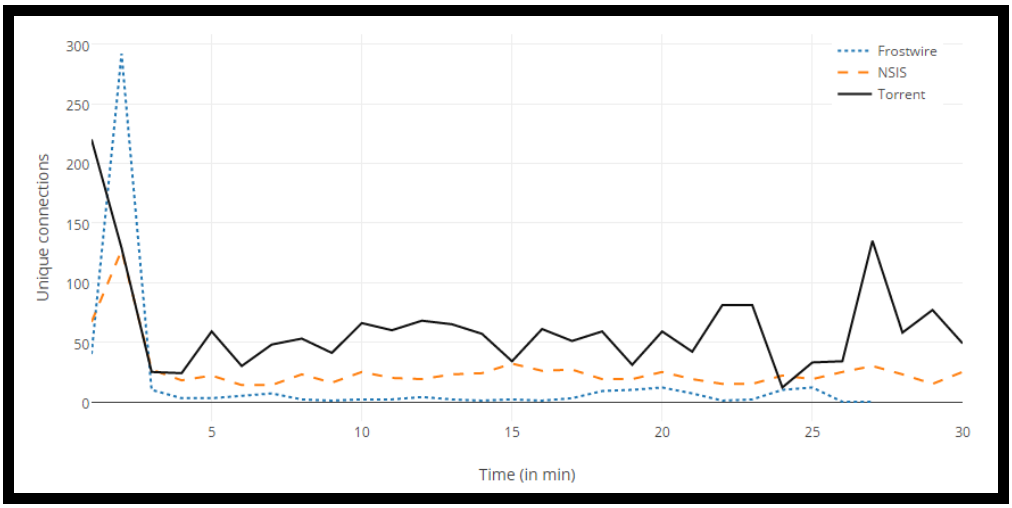

Figure 4.7: Limitation: Unique P2P Connections w.r.t time

i. Limitation:
The approach adopted by Kheir and Xiao [89] was to separate non-P2P bot program from P2P bot. However, if a mixture of traffic that has both benign P2P and bot P2P traffic, then this method is not efficient enough to distinguish these two. Similar to ASN, the geographic distribution of Peer IP addresses among P2P benign and P2P bots are also not distinguishable. Table 4.4 shows that in the case of benign P2P program, the peers were spread across the globe in huge number. Whereas for Zeus P2P bot, peers were limited in number and confined to the limited geographical region, but for NSIS P2P bot, the geographic distribution is similar to benign P2P.
This concludes that ASN distribution of remotely contacted machines might be effective in distinguishing non-P2P bot and P2P program. However, categorizing the P2P program as bot or benign P2P is not possible with this approach only.

Countries								
	Total	Number of Peers (P2P)						
		US	UK	Russia	India	China	Ukrain	Taiwar
benign P2P								
μTorrent [83]	133	209	72	103	131	19	16	23
Skype	38	23	11	6	11	-	2	1
Norm. P2P [78]	87	72	33	97	9	71	64	20
Bot P2P								
Zeus GO [71]	14	5	-	-	-	-	-	4
NSIS.a [70]	101	184	35	219	73	42	77	22

Table 4.4: Geographical distribution of peers

4.2 DNS based botnet detection

Table 4.5 describes the selected DNS based bot detection techniques considered during the research.

Selected method	Key aspets	Dataset used
Dennis [12] Soniya [86] Krmicek [47]	Repeating pattern High Density queries (DGA)	Zeus Gameover [71] CTU-Normal [78] Open Security Research [75] Zeus HTTP bots [68] SpeEYe HTTP bot [69].

Table 4.5: DNS based bot Detection

As discussed earlier, DNS queries are used by bots to get updates, commands, exfiltration of data etc. from/to remote C2 server(s). Be it HTTP based bot or P2P bot, all of them use DNS at some point of time in their life cycle for various operational purposes. For example, Domain Generation Algorithm (DGA) is used by bots to generate many domain names in case C2 servers domains gets blacklisted [12][8]. This should ideally generate spikes of DNS traffic flowing out of network interface. Peer to Peer botnet(e.g. Zeus Gameover) uses DGA to reach out to C2 server in case no response is received from its peers within predefined time period [75] (Refer Figure 4.8).

Also, even if DGA is not used, bots regularly keep sending DNS requests to outside world(sometimes to well-known servers like Google, Microsoft, Yahoo etc.) to check internet connectivity [68]. This phenomenon has been observed by Dennis [12] in the case of Zeus P2P bot.

To verify DNS traffic pattern in botnet communication, further testing was conducted with normal Dataset. Figure 4.9 below shows DNS traffic pattern going out of network for various types of benign applications [77][83][87].

From the figure, it is clearly visible that there exists no visible pattern in DNS requests made from the host/network. In comparison to this, Figure 4.8 shows interesting characteristics of DNS traffic for bot

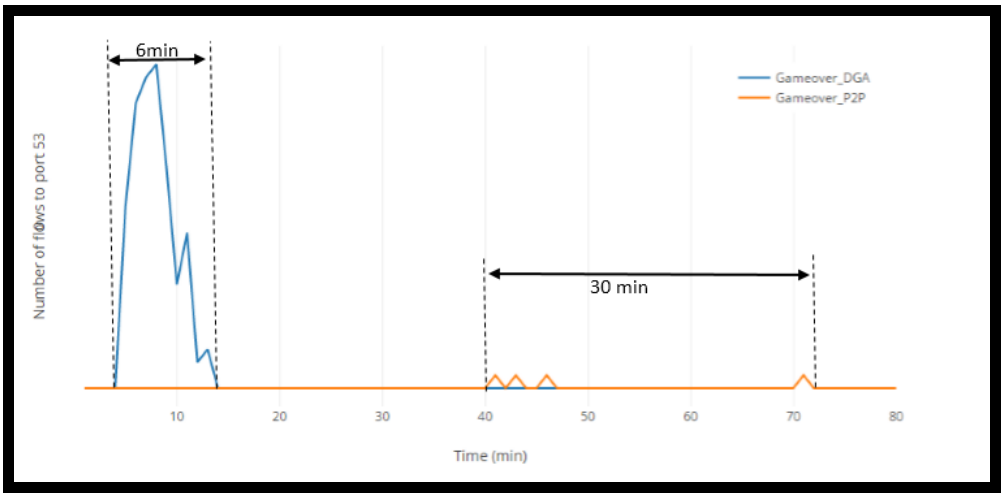

Figure 4.8: DNS traffic for P2P bot

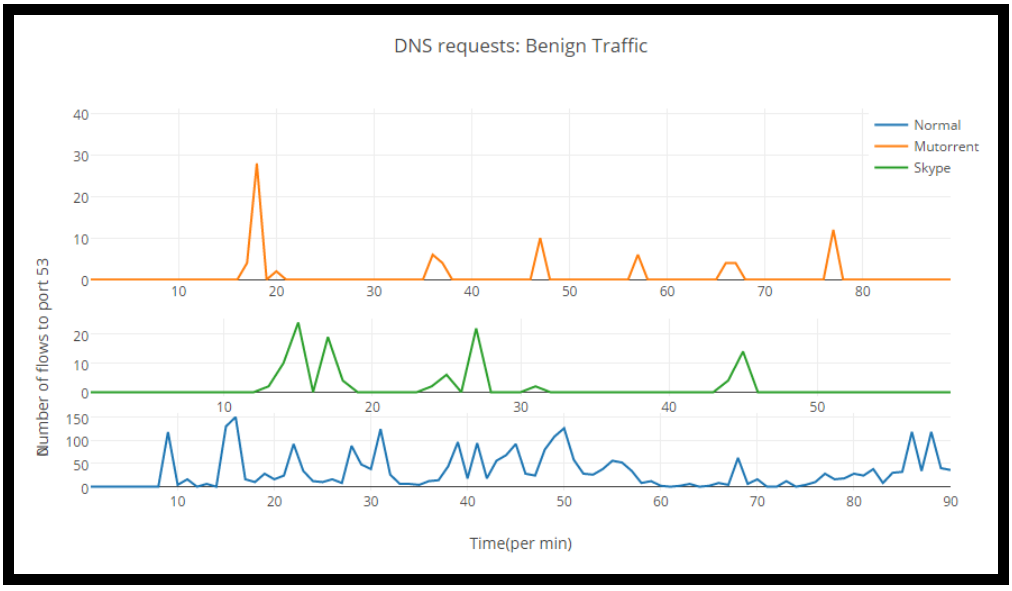

Figure 4.9: Normal DNS traffic

hosts. In this figure, two types of P2P bot characteristics have been discovered. P2P bots regularly send DNS requests to check internet connectivity when it senses unresponsive peer [12].Typical loop repeat time is 30 mins for Zeus Gameover sample considered. Secondly, DGA algorithm used by P2P bots. In this case, a sudden flux of many DNS requests within a short span of time is interesting bot detection parameter. This happens when bot hosts do not find active peer for communication even if it has working internet connection. DGA helps bots to generate huge domain names that are sent across the internet with an expectation of a response from C2 server. This phenomenon is also proposed as one of the detection parameters of DNS-based botnet detection inside network by Krmicek [47].

Work of Soniya [86] was also considered for checking repetitive DNS request pattern. In this work, bot communication was detected based on similar DNS request flows to a destination IP or domain at periodic intervals. The same was implemented in my experimental setup by checking all DNS request flows destined to port 53. Categorically, DNS flows to known destination IP was not selected, rather flows to destination port 53 were selected. This was done because depending on type and variant of HTTP bots, DNS request may be made to default DNS server of monitored network or to open DNS servers. Hence, monitoring flows to a destination IP may not detect all types and variants of bot. In my experimental setup, DNS characteristics were checked for HTTP bots like SpyEye and Zeus. DNS requests show regular communication pattern for these bot samples. During analysis, it was seen that sometimes DNS requests are also made to genuine websites like Google, Microsoft etc. (requested URLs were checked with Wireshark [90]).

Results show SpyEye sample maintain DNS requests that have repeating loop after each 30 mins. Similarly, Zeus samples show repeating pattern after 5 mins. Refer to Figure 4.10 for pictorial representation.

4.3 Mining based botnet detection

In addition to P2P and DNS based botnet detection, bot traffic classification using Data Mining has also been considered for implementation

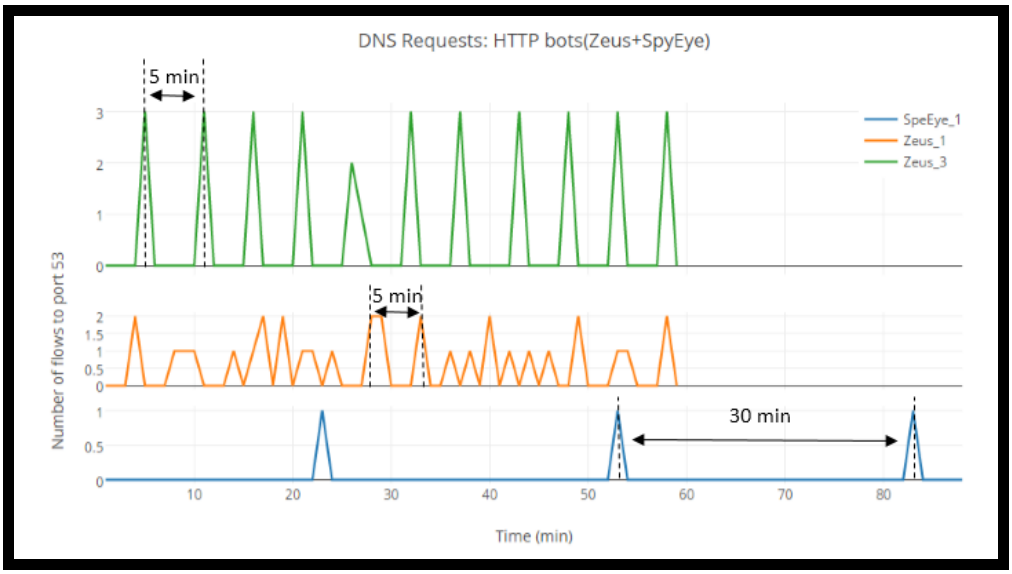

Figure 4.10: DNS traffic for HTTP bot

in our setup. Table 4.6 describes the selected Mining based bot detection techniques considered during the research.

Selected method	Key aspets	Dataset used
Haddadi [51]	Flow Feature Selection	For HTTP botnet classification:
Other Ref:	Classification Algorithm	
Zhao [15]	True Positive Rate (TPR)	Zeus_1 [68]
Udaya [49]	False Positive Rate (FPR)	Citadel [73]
		Lawrence Berkeley National Laboratory [76]

Table 4.6: Mining based HTTP bot Detection

Data Mining technique for botnet detection has been proposed in a number of researches as mentioned earlier in para 2.3.5. Mining based approach helps in selection of right parameters that can be used in

optimum botnet detection. Figure 4.11 below describes our approach in a pictorial way for Mining based botnet detection. The steps shown in this figure give the broad sequence of operations used for effective botnet detection using Machine Learning.

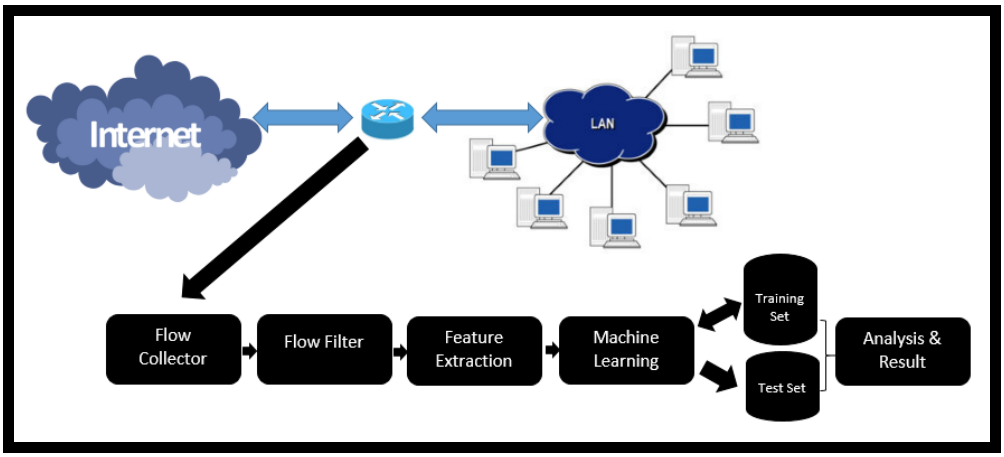

Figure 4.11: Machine learning setup

1. Chosen Mining technique:
   Based on the already referred work of Haddadi [51] and Zhao [15], I have used Classification technique for optimizing botnet detention for the reasons as mentioned below:
   - Identification of suitable botnet detection parameters
   - Using Classification method to check detection strength with the help of performance metrics like True Positives (TP) and False Positives (FP) etc.

2. Comparison between selected methods:
   As per research proposals by Haddadi [91] and Zhao [15], Decision Tree Classifier was mentioned as most accurate classification algorithm in comparison to Naïve Bayes/Bayesian networks. Decision Tree provides solution in the form of decision rules that are understood by the human operator. The C4.5 algorithm used in Decision Tree helps find out most appropriate features of flows. It also helps convert the trained tree into the if-then rule set. Whereas, Naïve Bayes is a type of supervised learning

algorithm and represents some kind of black box solution for the operator. However, it is important to note that their works were focused on different types of botnets. Work of Zhao [15] was mostly towards P2P based botnet detection and work of Haddadi [91] was towards HTTP based botnet detection. However, as per Udaya [49], Decision Tree classifier for P2P botnet and Bayesian networks classifier for HTTP, IRC botnet is effective. Hence, I will first consider checking the effectiveness of Decision Tree classifier (using C4.5 algorithm [1]) and then compare the result with Naïve Bayes classifier.
Table C.1 in Appendix C shows the Flow attributes considered by Haddadi and Zhao for Decision Tree based botnet Classification. Haddai in her experiment considered total 23 Flow attributes that can be achieved from Nfdump output directly. However, in her work, Haddadi did not consider Source IP Address and Port. She argued not using them because IP spoofing by Malware is possible and HTTP-based bots use a dynamic port. This in a way limits us from direct identification of bot host within the network from classification outcome. On the other hand, without Source IP Address and Port provides the detection engine enough flexibility to detect bot traffic based on the selected Flow attributes only. It's a tradeoff and human judgment is necessary to choose which one to choose! Whereas, Zhao considered attributes (total 13 no, including Src and Dst IP Addresses and Ports) , some of them can directly be retrieved Nfdump output and some of them have to be derived.
For the implementation of selected Mining based bot traffic classification purpose, Haddadi's approach has been considered. Though I will be using the Flow attributes selected by Haddadi, I will also consider the possibility of optimization of these attributes to achieve better classification accuracy.

3. Choosing of Mining platform:
   I have used well known open source Data Mining platform named Weka [93] from Waikato University, New Zealand. Weka

1. As per Weka data mining tool, J48 is an open source Java implementation of the C4.5 algorithm [92].

is a repository of all well-known machine learning algorithms. It has been used in various research projects like botnet traffic classification [94], Detecting malicious C2 Traffic Communications [95], Protocol recognition using NetFlow [96], botnet behavior analysis [51] etc.

4. Generation of Dataset compatible with Mining platform:
   It was important to create a suitable file format that can be accepted by Weka. Weka accepts file like .ARFF (Attribute-Relation File Format) for the same. An example of various parameters obtained from Nfdump are shown below in Table 4.7 (with limited values).

Duration	Protocol	srcIP	srcPort	InPkt	InByte	dstIP	dstPort	OutPkt	OutByte	Tpye
1827.654	17	172.29.0.116	1026	31	1927	75.75.75.75	53	32	3354	benign
34.640	6	172.29.0.116	1488	10	1214	91.191.168201	80	4	1282	benign
6.263	6	172.29.0.116	1489	12	1080	206.123.104	80	8	1204	benign
34.033	6	172.29.0.116	1490	102	6466	209.59.217102	80	148	197814	benign
262.974	17	172.29.0.116	23575	851	149372	69.208.93.133	28885	867	1227769	bot
0.00	17	172.29.0.116	23575	2	562	79.5.82.66	27863	0	0	bot
217.173	17	186.93.216136	20051	22	9150	172.29.0.116	23575	6	1402	bot

Table 4.7: Selected NFDUMP Parameters

Conversion of regular Nfdump output to ARFF file format was a tricky part. A sample of such file format used during my research generated from Nfdump output is shown in figure 4.12 below.

5. Mining for HTTP botnet Classification
   HTTP bots were considered during the machine learning experiment. As mentioned earlier, Work of Haddadi [51] focused on detection of HTTP botnet using Decision Tree classifier.

   (a) Dataset preparation and feature extraction:
   Various variants of Zeus C2 bots were taken and tested with Weka for evaluating detection strength. Features proposed at Table C.1 (1st column) was considered. Since there is

```
% 1. Tile: HTTP Bot dataset
% 2. Project: Selected Botnet Detecting using FLow data

@RELATION network
@ATTRIBUTE Duration NUMERIC
@ATTRIBUTE Src_port NUMERIC
@ATTRIBUTE Dst_port NUMERIC
@ATTRIBUTE Source_AS NUMERIC
@ATTRIBUTE Dst_AS NUMERIC
@ATTRIBUTE In_interface NUMERIC
@ATTRIBUTE Out_interfcae NUMERIC
@ATTRIBUTE Total_pkt NUMERIC
@ATTRIBUTE In_pkt NUMERIC
@ATTRIBUTE Out_pkt NUMERIC
@ATTRIBUTE Total_bytes NUMERIC
@ATTRIBUTE In_bytes NUMERIC
@ATTRIBUTE Out_bytes NUMERIC
@ATTRIBUTE Flows NUMERIC
@ATTRIBUTE Tos NUMERIC
@ATTRIBUTE Src_Tos NUMERIC
@ATTRIBUTE Dst_Tos NUMERIC
@ATTRIBUTE Src_mask NUMERIC
@ATTRIBUTE Dst_mask NUMERIC
@ATTRIBUTE Fwd_status NUMERIC
@ATTRIBUTE Src_vlan_label NUMERIC
@ATTRIBUTE Dst_vlan_label NUMERIC
@ATTRIBUTE bps NUMERIC
@ATTRIBUTE pps NUMERIC
@ATTRIBUTE bpp NUMERIC
@ATTRIBUTE Protocol NUMERIC
@ATTRIBUTE class        {Bot,Benign}
@DATA
2.966,63852,80,0,0,0,6,6,6,374,374,422,2,0,0,0,0,0,0,0,1008,2,62,6,Bot
116.076,63900,80,0,0,0,4,4,4,202,202,226,2,0,0,0,0,0,0,0,13,0,50,6,Bot
4.750,3389,60656,0,0,0,29,29,37,1565,1565,1764,2,0,0,0,0,0,0,0,2635,6,53,6,Bot
0.115,2982,80,0,0,0,10,10,8,1094,1094,1258,4,0,0,0,0,0,0,0,76104,86,109,6,Benign
```

Figure 4.12: Part of Attribute-Relation File Format(ARFF) for HTTP bot classification

a paucity of readily available training Dataset for HTTP based bot detection, it was decided to mix traffic of malicious HTTP bot Zeus_1 (Table: 4.8) and benign traffic from Lawrence Berkeley National Laboratory (Table:4.1, Sr no:15).This benign normal traffic dataset upon analysis showed that it has around 15 mins traffic duration and there are about 423 unique Source IP addresses that are communicating. The characteristics of the merged traffic are given in Table 4.9 below.

Zhao [15] proposed a method of merging Datasets. In their experiment, they converted infected machine IP address to IP address that is within the background traffic subnet. This was done because Zhao considered Flow features including Source and Destination IP addresses (ref Table C.1, 2nd column) during their traffic Training and Testing phase. However, in my experiment, I didn't follow this approach

Source	Duration (min)	Flows	Bytes	Packet	Avg bps	Avg pps	Avg bpp	Norma Traffic	Bot Traffic
Zeus_1	60	1606	974622	9072	2123	2	107	0%	100%
Zeus_2	60	1411	1.5M	7437	3230	2	195	0%	100%

Table 4.8: HTTP bot Dataset Characteristics

Source	Duration (min)	Flows	Bytes	Packet	Avg bps	Avg pps	Avg bpp	Norma Traffic	Bot Traffic
LBNL	15	28023	685 M	987812	8 M	1439	693	100%	0%
Zeus_1	15	141	95666	1096	2771	3	87	0%	100%
Zeus_1 + LBNL	15	28164	685.4 M	988908	8.0 M	1437	693	99.5%	0.5%

Table 4.9: HTTP mixed Dataset Characteristics

because I did not consider IP address during Training and Testing of Classifier. This was done in line with the proposed detection method of Haddadi [91](Ref: TableC.1, 1st column).
The process of mixing Datasets is shown in figure 4.13 below.

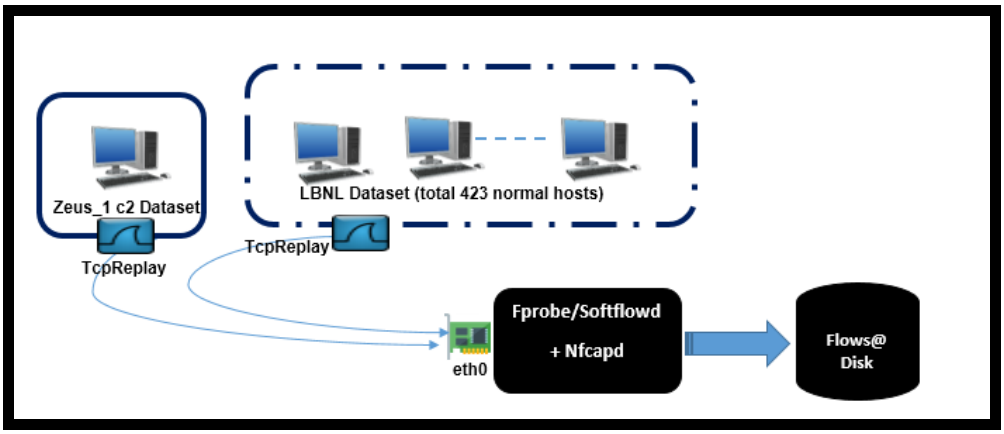

Figure 4.13: Mix Dataset creation setup for Mining

(b) Classification with merged Dataset:

Table 4.10 below shows the detection capability of Decision Tree Classifying Algorithm to detect bot traffic from a mixture of bot and benign traffic. Here Accuracy of bot detection capability in terms of True Positive Rate (TPR) and False Positive Rate (FPR) has been calculated with or without a filter.
The classification was done with k-fold cross validation (k=10) [97] using C4.5 Decision Tree classification algorithm with other default settings. k-fold cross-validation estimator provides lower variance than single hold-out set estimator if the amount of data available is limited.
I have used two feature sets. First set consists of the features already considered by Haddadi and the second set is the set consisting of features selected by Haddadi plus Protocol

Test_1: HTTP Dataset (LBNL + Zeus_1)				
Features(without HTTP filter)				
	Malicious		benign	
	TPR	FPR	TPR	FPR
Haddadi	73.5%	0%	100%	26.5%
Haddadi + Protocol + Port	76.5%	0%	100%	23.5%
Features(with HTTP filter)				
	Malicious		benign	
	TPR	FPR	TPR	FPR
Haddadi	80%	0.2%	99.8%	20%
Haddadi + Protocol + Port	96.7%	0%	100%	3.3%

Table 4.10: HTTP bot detection accuracy (with Decision Tree)

and Ports used. This second feature set was considered to explore any further scope of improvement possible over Haddadi's approach in botnet detection.
With the first feature set, we could get detection accuracy with TPR of 73.5% for bot traffic classification. However, by introducing HTTP filter, the accuracy increased to 80%. When the same was repeated with the second feature set, we could get a reasonable amount of HTTP bot detection accuracy with TPR of 76.5%(without filter) and 96.7%(with filter).
This concludes two things. First, with the introduction of filter, bot detection capability increases and second, features selection play vital role in the accuracy of bot traffic classification.

(c) Comparison between Classification Algorithms:
To cross check the classifying strength of Decision Tree classifying technique, Naïve Bayes classification algorithm was chosen. This was categorically done because as per

Haddadi [51], Naïve Bayes provides poor HTTP botnet detection than Decision Tree. Table 4.11 below shows the result obtained from this classifying technique.

Test_2: HTTP Dataset (LBNL + Zeus_1)				
Features: Only Haddadi without HTTP filter				
	Malicious		benign	
	TPR	FPR	TPR	FPR
Decision Tree	73.5%	0%	100%	26.5%
Naïve Bayes	91.2%	30.7%	69.3%	8.8%
Features: Haddadi+ Protocol + Port + without HTTP filter				
Decision Tree	76.5%	0%	100%	23.5%
Naïve Bayes	91.2%	9.8%	90.2%	8.8%
Features: Only Haddadi with HTTP filter				
Decision Tree	80%	0.2%	99.8%	20%
Naïve Bayes	100%	21.6%	78.4%	0%
Features: Haddadi+ Protocol + Port + with HTTP filter				
Decision Tree	96.7%	0%	100%	3.3%
Naïve Bayes	100%	0.3%	99.7%	0%

Table 4.11: HTTP botnet detection: Decision Tree vs Naïve Bayes

After comparing the two widely used classifying algorithms, we could see that Naïve Bayes classifier can better classify HTTP bot traffic with or without a filter, in comparison to Decision Tree Based Classification.
Further the same was cross-checked using Receiver Operating Characteristic (ROC) curve. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) of a classifier. The area under ROC curve represents the accuracy of class detection. More the area under ROC curve better is the detection strength. When ROC curve was

plotted with and without a filter using features selected by Haddadi (Test_2), it was further noticed that Naïve Bayes classifier could better classify Zeus HTTP bot traffic with or without a filter, in comparison to Decision Tree Based Classification. The same is shown in figure 4.14 below.

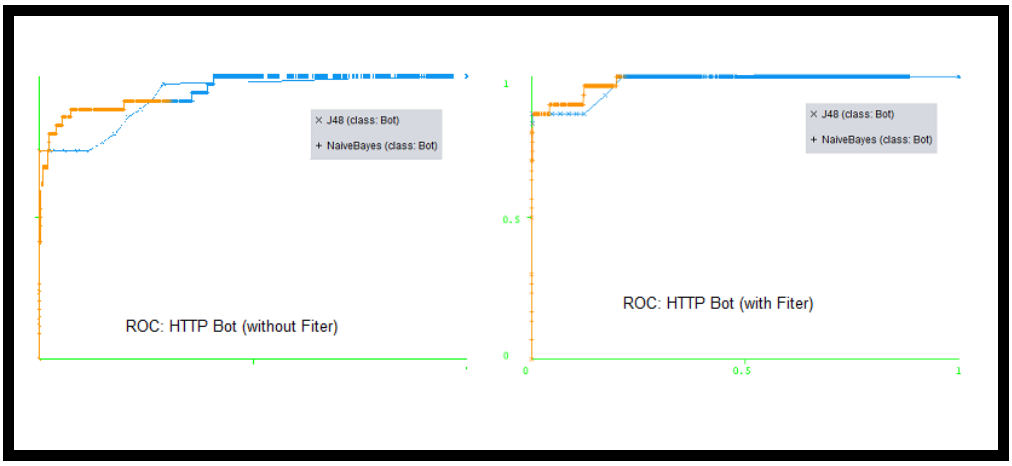

Figure 4.14: ROC Curve:HTTP bot

(d) Analysis of result on Mining based HTTP bot detection:
As in our experiment Naïve Bayes classifier showed better classifying capability for HTTP bot traffic with or without filter, hence further analysis of Haddadi's approach was considered and following were found:

i. In Haddadi's work, equal number of flows were taken from benign (Alexa Dataset) and bot traffic (Citadel/Zeus Dataset). Haddadi used script to generate "representative traffic" to known bot domain names and captured the flows. To verify the classification accuracy with a similar setting, it was decided to re-run the test (Test_2) with a newly formed merged dataset in our setup. The ratio of normal (LBNL) and malicious (Zeus_1) traffic inside the newly constructed Dataset this time was manipulated and now almost equal entries in .ARFF file represents bot and benign traffic. This manipulation

was done by considering flows from a single normal host out of 423 hosts within LBNL Dataset and merging that with Zeus bot flows. With this setting, the detection algorithm was run with 10 fold cross validation and the result shows a visible difference from earlier results. This is shown in Table 4.12 below. In this case (Test_3), I have created three sets of merged Datasets. First Dataset consists of an almost equal amount of Zeus bot and LBNL normal traffic. In second set, an almost equal amount of Citadel bot and LBNL normal traffic has been used. Use of second Dataset with Citadel bot was done because I wanted to verify the classification behavior with other HTTP bot (apart from Zeus) and also Haddadi used this bot during her experiment. Third Dataset consists of almost equal amount of LBNL normal traffic and Sogou HTTP bot traffic [70] .Interestingly, with this new setup, Decision Tree classifier did better classification of botnet traffic than Naïve Bayes classification.
Also, the ROC curve generated with this new setup (Test_3 with LBNL+Zeus_1 Dataset), shows improved detection capability of both Decision Tree and Naïve Bayes classifier. Refer to Figure 4.15 below.
Various Decision Trees formed by classifier are given in figure 4.16.
After analyzing the classification results provided by Decision Tree, it is found that optimum flow parameters (direct features retrieved from Nfdump) that have highest influences in HTTP bot detection are: Duration, Flows, Total Bytes, pps, Total Packets, bpp.

ii. Conclusion on Mining based HTTP bot detection:
We were able to create two situations wherein one (Test_1/Test_2) Naïve Bayes performed better and in another (Test_3), Decision Tree performed better. It is found that contribution of bot traffic within entire traffic influences on which classification algorithm would

Test_3				
Dataset having almost equal bot and benign entries (LBNL + Zeus_1)				
Features: Haddadi +Without Filter	Malicious		benign	
	TPR	FPR	TPR	FPR
Decision Tree	99.5%	0.4%	99.6%	0.5%
Naïve Bayes	91.5%	0.6%	99.4%	8.5%
Dataset having almost equal bot and benign entries (LBNL + Citadel)				
Features: Haddadi +Without Filter	Malicious		benign	
	TPR	FPR	TPR	FPR
Decision Tree	99.8%	0.2%	99.8%	0.2%
Naïve Bayes	90.9%	1.4%	98.6%	9.1%
Dataset having almost equal bot and benign entries (LBNL + Sogou)				
Features: Haddadi +Without Filter	Malicious		benign	
	TPR	FPR	TPR	FPR
Decision Tree	89.4%	5.7%	94.3%	10.6%
Naïve Bayes	31.6%	1.4%	98.6%	68.1%

Table 4.12: HTTP botnet detection: Decision Tree vs Naïve Bayes (modified)

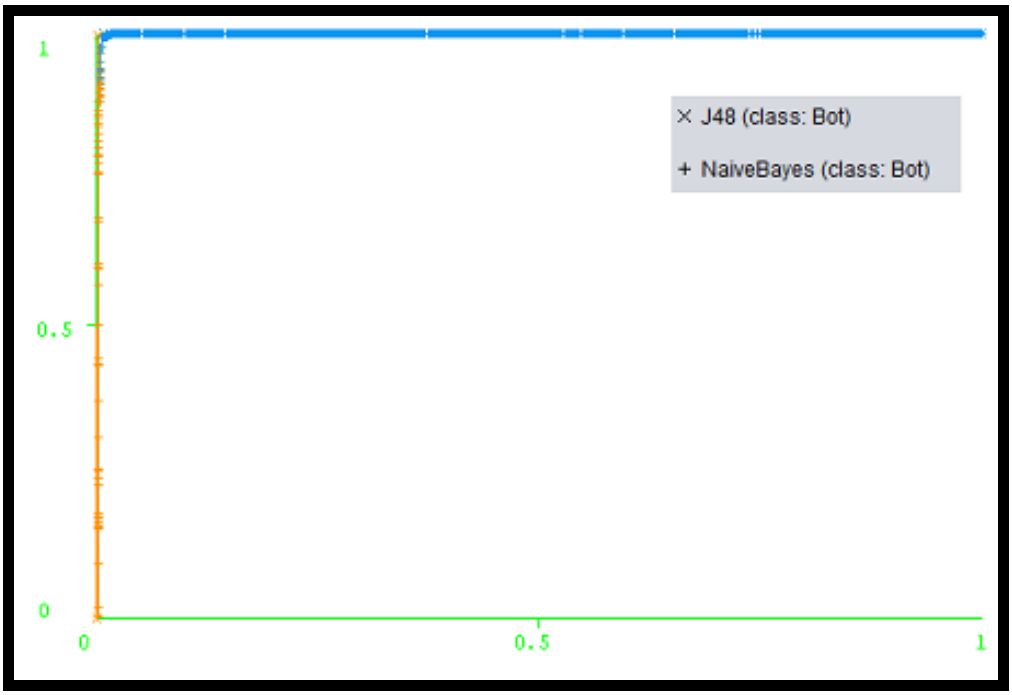

Figure 4.15: ROC Curve:Equal traffic ratio (without filter)

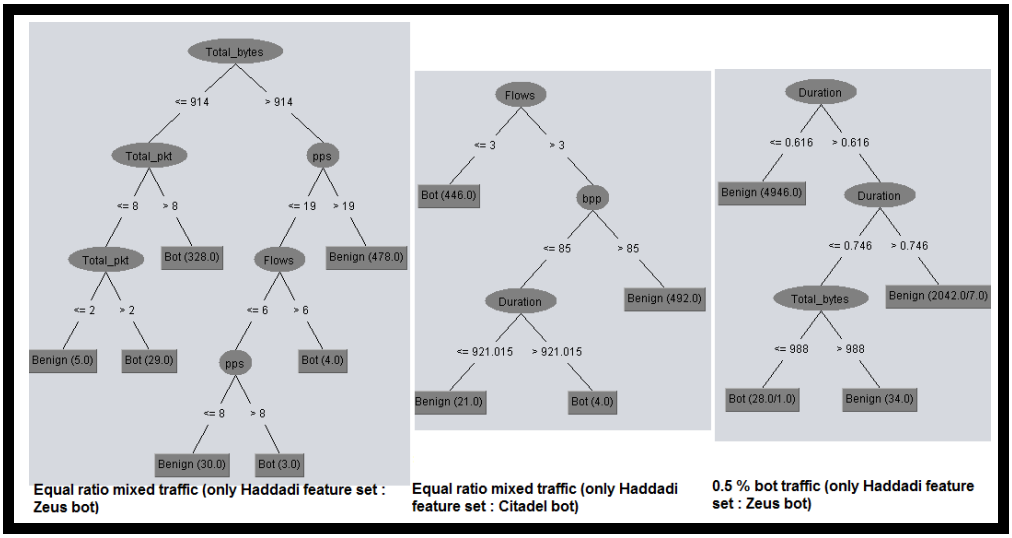

Figure 4.16: Decision Tree: HTTP bot classification (without filter)

perform better. This implies that, if the Flow probe is very close to the host under observation or probing very small subnet, a good percentage of bot traffic can be tracked and Decision Tree classification would be useful. And for those probes, that are covering bigger network and therefore the percentage of bot traffic is relatively small within the entire network traffic, Naïve Bayes classification would perform better.
We could also found that with the utilization of right filter, we can increase the efficiency of classification engine to great extent with high TPR and low FPR.

4.4 Summary

In this chapter, implementations of selected P2P, DNS and HTTP based botnet detection techniques were considered for analysis. However, variations/limitations were detected/discussed when tested with multiple datasets that were not considered in original/referred detection techniques. Whereas, machine learning based traffic classification appears to be more generic approach because it does not take into account any prior flow discrimination characteristics during botnet analysis. Rather, classification helps the detection engine to understand/learn about botnet traffic during training phase and then using the learned rules classify botnet traffic during test phase. Classification also helps choosing the right flow characteristics (e.g Decision tree based classification) for optimum botnet detection. This become more effective when new botnet is considered for analysis that does not have prior known pattern. Hence, a comprehensive look at these existing detection approaches helped formulate optimum generic botnet detection methodology, where data mining based traffic classification approach may be considered for efficient botnet detection.

5 Conclusion

A number of research papers have been considered where Flow based botnet detection method have been utilized. Among them, few selected approaches have been considered for in-house implementation and testing with multiple Datasets. These Datasets have been generated in-house and also collected from open research facilities. These Datasets consists of P2P, HTTP, and DNS bot traffic. In general, proactive approach has been adopted by identifying host(s) that are likely part of botnet before an attack takes pace, by extracting and analyzing flow characteristics that match botnet communication. HTTP, P2P, DNS based botnet detection have been considered because these bots are highly resilient and more prevalent in the Cyber world. Time-based, Flow statistics based, spatial distribution based and Classification based approaches were considered for detection of P2P, DNS, and HTTP-based botnet detection. I have implemented and analyzed a couple of interesting botnet detection techniques and also provided examples where variation/limitations of such techniques are possible. For example, spatial distribution and DNS based detection approaches show interesting results as below:

1. Host specific Peer churn might be important distinguishable characteristics for P2P botnet detection. This can be used to differentiate non-P2P bot, P2P benign and P2P bot. However, variation was found when compared among multiple P2P bots.

2. Host specific sudden flux of DNS traffic flowing out of monitored network and/or repetitive DNS queries may indicate DNS based bot communication. With Zeus P2P and other Zeus and SpyEye C2 bots, this property showed interesting results. However, there could be other bots which may not show any DNS pattern (e.g. dormant bots, bots using DNS for data exfiltration etc.).

After analysis of selected botnet detection techniques, following are concluded:

1. Most of the detection methods take into account limited Data sets from few selected bot samples for research purposes. When tested with other Datasets, variations in results are observed.

2. Mining based botnet detection is a more generic approach in classifying bot traffic from a mixture of benign and bot traffic. Mining provides a platform to classify traffic even if a distinguishable pattern is not known to the user. Additionally, classification may help figure out optimum parameters from flow data that carries highest discriminating power for classification of botnet traffic out of mixed traffic in a network.

3. Choosing the right features out of Nfdump output (direct or derived feature) is crucial and influences accuracy of classification algorithms.

4. Also, choosing right classification algorithm is important for botnet detection. It was found that accuracy of HTTP based botnet classification algorithm is highly influenced by the position of Flow Probe in the monitored network.

5. As Decision Tree classifier generates human understandable rule set, hence to get effective detection result based on Decision Tree, Flow probe should be placed as close as possible to the target machine(s)/subnet. For a large Enterprise network, probes at internal subnet level would give better visibility of traffic characteristics than probe at network peripheral/gateway layer.

6. Only flow based botnet detection carries some limitations in effective detection of botnet.

7. Future work of this research may include

   (a) Testing of detection accuracy with other classification algorithms, e.g SVM, Random Forest, REPTree, BayesNet etc.

   (b) Utilization of Clustering technique for botnet detection

   (c) botnet detection by combined verification of Flow data with other data (like IDS logs, Host behavior etc.).

Bibliography

[1] Hyunsang Choi, Heejo Lee, and Hyogon Kim. "BotGAD: detecting botnets by capturing group activities in network traffic". In: *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*. ACM. 2009, p. 2.

[2] Symantec. *Internet Security Threat Report*. https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf/. 2016.

[3] Trend Micro. *Taxonomy of Botnet Threats*. http://www.cs.ucsb.edu/~kemm/courses/cs595G/TM06.pdf/. [Online; accessed Sept-2016]. 2006.

[4] SéRgio SC Silva et al. "Botnets: A survey". In: *Computer Networks* 57.2 (2013), pp. 378–403.

[5] Michael Bailey et al. "A survey of botnet technology and defenses". In: *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*. IEEE. 2009, pp. 299–304.

[6] Hossein Rouhani Zeidanloo et al. "A taxonomy of botnet detection techniques". In: *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*. Vol. 2. IEEE. 2010, pp. 158–162.

[7] EggHeads. *EggHeads.org-eggdrop development*. http://eggheads.org/. 1993.

[8] Anirudh Ramachandran, Nick Feamster, David Dagon, et al. "Revealing Botnet Membership Using DNSBL Counter-Intelligence." In: ().

[9] Brett Stone-Gross et al. "Analysis of a botnet takeover". In: *IEEE Security & Privacy* 9.1 (2011), pp. 64–72.

[10] Martin Grill et al. "Detecting DGA malware using NetFlow". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE. 2015, pp. 1304–1309.

[11] Maryam Feily, Alireza Shahrestani, and Sureswaran Ramadass. "A survey of botnet and botnet detection". In: *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. IEEE. 2009, pp. 268–273.

[12] Dennis Andriesse and Herbert Bos. *An analysis of the zeus peer-to-peer protocol*. 2013.

[13] Prerika Agarwal and Sangita Satapathy. "Implementation of Signature-based Detection System using Snort in windows". In: (2014).

[14] Peter Wurzinger et al. "Automatically generating models for botnet detection". In: *European symposium on research in computer security*. Springer. 2009, pp. 232–249.

[15] David Zhao et al. "Peer to peer botnet detection based on flow intervals". In: *IFIP International Information Security Conference*. Springer. 2012, pp. 87–102.

[16] G.S.S. Krishnan et al. *Computational Intelligence, Cyber Security and Computational Models: Proceedings of ICC3, 2013*. Advances in Intelligent Systems and Computing. Springer India, 2013. ISBN: 9788132216803. URL: https://books.google.co.in/books?id=PTbABAAAQBAJ.

[17] Rafael A Rodriguez-Gomez, Gabriel Macia-Fernandez, and Pedro Garcia-Teodoro. "Survey and taxonomy of botnet research through life-cycle". In: *ACM Computing Surveys (CSUR)* 45.4 (2013), p. 45.

[18] Inc OpenDNS. *Abstracted Timeline and illustrations*. http://info.opendns.com/rs/opendns/images/OpenDNS_SecurityWhitepaper-DNSRoleInBotnets.pdf/. 2016.

[19] Eng Keong Lua et al. "A survey and comparison of peer-to-peer overlay network schemes". In: *IEEE Communications Surveys & Tutorials* 7.2 (), pp. 72–93.

[20] Sylvia Ratnasamy et al. *A scalable content-addressable network*. Vol. 31. 4. ACM, 2001.

[21] Ben Y Zhao et al. "Tapestry: A resilient global-scale overlay for service deployment". In: *IEEE Journal on selected areas in communications* 22.1 (2004), pp. 41–53.

[22] Ping Wang, Sherri Sparks, and Cliff C Zou. "An Advanced Hybrid Peer-to-Peer Botnet." In: *IEEE Transactions on Dependable and Secure Computing* 7.2 (2010), p. 113.

[23] Gao Jian et al. "Research of an innovative P2P-based botnet". In: *Machine Vision and Human-Machine Interface (MVHI), 2010 International Conference on*. IEEE. 2010, pp. 214–218.

[24] Trend Micro. *SDBot*. http://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/sdbot/. 2016.

[25] Trend Micro. *SDBot*. https://www.symantec.com/security_response/writeup.jsp?docid=2002-051312-3628-99/. 2016.

[26] F-Secure. *Agobot*. https://www.f-secure.com/v-descs/agobot.shtml/. 2016.

[27] James R Binkley and Suresh Singh. "An Algorithm for Anomaly-based Botnet Detection." In: *SRUTI* 6 (2006), pp. 7–7.

[28] Sajjad Arshad et al. "An anomaly-based botnet detection approach for identifying stealthy botnets". In: *Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference on*. IEEE. 2011, pp. 564–569.

[29] Anestis Karasaridis, Brian Rexroad, David A Hoeflin, et al. "Wide-Scale Botnet Detection and Characterization." In: *HotBots* 7 (2007), pp. 7–7.

[30] Carl Livadas et al. "Usilng machine learning technliques to identify botnet traffic". In: *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*. IEEE. 2006, pp. 967–974.

[31] W Timothy Strayer et al. "Detecting botnets with tight command and control". In: *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*. IEEE. 2006, pp. 195–202.

[32] Jan Goebel and Thorsten Holz. "Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation." In: *HotBots* 7 (2007), pp. 8–8.

[33] Jignesh Vania, Arvind Meniya, and HB Jethva. "A review on botnet and detection technique". In: *Int J Comput Trends Technol* 4.1 (2013), pp. 23–29.

[34] Evan Cooke, Farnam Jahanian, and Danny McPherson. "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets." In: *SRUTI* 5 (2005), pp. 6–6.

[35] Guofei Gu et al. "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection." In: *USENIX Security Symposium*. Vol. 5. 2. 2008, pp. 139–154.

[36] Guofei Gu, Junjie Zhang, and Wenke Lee. "BotSniffer: Detecting botnet command and control channels in network traffic". In: (2008).

[37] Roberto Perdisci, Wenke Lee, and Nick Feamster. "Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces." In: *NSDI*. 2010, pp. 391–404.

[38] Jae-Seo Lee et al. "The activity analysis of malicious http-based botnets using degree of periodic repeatability". In: *Security Technology, 2008. SECTECH'08. International Conference on*. IEEE. 2008, pp. 83–86.

[39] Jérôme François, Shaonan Wang, Thomas Engel, et al. "BotTrack: tracking botnets using NetFlow and PageRank". In: *International Conference on Research in Networking*. Springer. 2011, pp. 1–14.

[40] Connor Dillon. "Peer-to-Peer Botnet Detection Using NetFlow". In: (2014).

[41] Nizar Kheir and Chirine Wolley. "Botsuer: Suing stealthy p2p bots in network traffic through netflow analysis". In: *International Conference on Cryptology and Network Security*. Springer. 2013, pp. 162–178.

[42] Vixie P et al. *Dynamic Updates in the Domain Name System (DNS UPDATE)*. http://www.faqs.org/rfcs/rfc2136.html/. 1997.

[43] Anirudh Ramachandran, David Dagon, and Nick Feamster. "Can DNS-based blacklists keep up with bots?" In: *CEAS*. Citeseer. 2006.

[44] Ahmed M Manasrah et al. "Detecting botnet activities based on abnormal DNS traffic". In: *arXiv preprint arXiv:0911.0487* (2009).

[45] Hyunsang Choi et al. "Botnet detection by monitoring group activities in DNS traffic". In: *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*. IEEE. 2007, pp. 715–720.

[46] Ricardo Villamarin-Salomon and Jose Carlos Brustoloni. "Identifying botnets using anomaly detection techniques applied to DNS traffic". In: *2008 5th IEEE Consumer Communications and Networking Conference*. IEEE. 2008, pp. 476–481.

[47] Vojtech Krmicek. "Inspecting DNS Flow Traffic for Purposes of Botnet Detection". In: *GEANT3 JRA2 T4 Internal Deliverable* (2011).

[48] Mohammad M Masud et al. "Flow-based identification of botnet traffic by mining multiple log files". In: *Distributed Framework and Applications, 2008. DFmA 2008. First International Conference on*. IEEE. 2008, pp. 200–206.

[49] Udaya Wijesinghe, Udaya Tupakula, and Vijay Varadharajan. "An Enhanced Model for Network Flow Based Botnet Detection".

In: *Proceedings of the 38th Australasian Computer Science Conference (ACSC 2015)*. Vol. 27. 2015, p. 30.

[50] Pedram Amini, Reza Azmi, and MuhammadAmin Araghizadeh. "Botnet Detection using NetFlow and Clustering". In: *Advances in Computer Science: an International Journal* 3.2 (2014), pp. 139–149.

[51] Fariba Haddadi et al. "Botnet behaviour analysis using ip flows: with http filters using classifiers". In: *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*. IEEE. 2014, pp. 7–12.

[52] Anna Sperotto et al. "An Overview of IP Flow-Based Intrusion Detection". In: *IEEE Communications Surveys & Tutorials* 12.3 (2010), pp. 343–356. URL: http://doc.utwente.nl/72752/.

[53] M. Graham, A. Winckles, and E. Sanchez-Velazquez. "Botnet detection within cloud service provider networks using flow protocols". In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. July 2015, pp. 1614–1619. DOI: 10.1109/INDIN.2015.7281975.

[54] W Timothy Strayer et al. "Botnet detection based on network behavior". In: *Botnet Detection*. Springer, 2008, pp. 1–24.

[55] ZOHO Corp. *Configuring NetFlow Export on an IOS Device*. https://www.manageengine.com/products/netflow/help/cisco-netflow/cisco-ios-netflow.html/.

[56] Inc. Cisco Systems. *Introduction to Cisco IOS NetFlow - A Technical Overview*. http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html/.

[57] *JFlow*. http://www.juniper.net/us/en/local/pdf/app-notes/3500204-en.pdf/.

[58] *Netstream-Huawei Technologies*. enterprise.huawei.com/ilink/enenterprise/download/HW_201022/.

[59] *RFlow- Ericsson*. http://rbdoc.ufanet.ru/en_lzn7830011_1_r1a/78_1543-CRA1191170_1-V1Uen.A.html/.

[60] Benoit Claise. *Specification of the IP flow information export (IPFIX) protocol for the exchange of IP traffic flow information*. Tech. rep. 2008.

[61] Mark Graham, Adrian Winckles, and Erika Sanchez-Velazquez. "Botnet detection within cloud service provider networks using flow protocols". In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE. 2015, pp. 1614–1619.
[62] *fprobe*. http://fprobe.sourceforge.net/.
[63] *fprobe*. http://manpages.ubuntu.com/manpages/wily/man8/fprobe.8.html/.
[64] *Softflowd*. http://www.mindrot.org/projects/softflowd/.
[65] *nfcapd - netflow capture daemon*. http://manpages.ubuntu.com/manpages/precise/man1/nfcapd.1.html.
[66] *nfdump - netflow display and analyze program*. http://manpages.ubuntu.com/manpages/wily/man1/nfdump.1.html/.
[67] *Nfdump*. http://nfdump.sourceforge.net/.
[68] *zeustracker*. https://zeustracker.abuse.ch/.
[69] *Malwr - Malware Analysis by Cuckoo Sandbox*. https://malwr.com/.
[70] Sebastian Garcia et al. "An empirical comparison of botnet detection methods". In: *Computers and Security Journal,Elsevier. 2 Vol 45, pp 100-123.* (2014).
[71] Mila Parkour. *Contagio Malware Dump*. http://contagiodump.blogspot.in/2013/04/collection-of-pcap-files-from-malware.html. 2015.
[72] Sourcefire VRT. *VRT Labs - Zeus Trojan Analysis*. https://labs.snort.org/papers/zeus.html/.
[73] DeepEnd Research. *CRIME/DeepEnd Research*. https://www.dropbox.com/sh/wje7mxs4nour40k/AAAyl0r16dIW8r4fTEuCPTc_a/PCAPS_TRAFFIC_PATTERNS/CRIME/. 2015.
[74] *Publicly available PCAP files*. http://www.netresec.com/?page=PcapFiles/.
[75] Ismael Valenzuela. *Identifying Malware Traffic with Bro and the Collective Intelligence Framework (CIF)*. http://blog.opensecurityresearch.com/2014/03/identifying-malware-traffic-with-bro.html. 2014.
[76] LBNL/ICSI. *LBNL/ICSI Enterprise Tracing Project*. http://www.icir.org/enterprise-tracing/. 2013.
[77] *CTU-Normal-4-only-DNS*. https://stratosphereips.org/category/dataset.html/.

[78] *CTU-Normal-7*. https://stratosphereips.org/category/dataset.html/.
[79] wikipedia. *Zeus (malware)*. https://en.wikipedia.org/wiki/Zeus_(malware).
[80] Symantec Corporation. *SpyEye Bot versus Zeus Bot*. https://www.symantec.com/connect/blogs/spyeye-bot-versus-zeus-bot.
[81] *Tcpreply*. http://tcpreplay.synfin.net/.
[82] *Tcpreply*. https://linux.die.net/man/1/tcpreplay/.
[83] *µTorrent*. http://www.utorrent.com/intl/en/.
[84] *Frostwire-6.3.5.windows.exe*. http://www.frostwire.com/.
[85] Ting-Fang Yen and Michael K Reiter. "Are your hosts trading or plotting? Telling P2P file-sharing and bots apart". In: *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*. IEEE. 2010, pp. 241–252.
[86] B Soniya and M Wilscy. "Using entropy of traffic features to identify bot infected hosts". In: *Intelligent Computational Systems (RAICS), 2013 IEEE Recent Advances in*. IEEE. 2013, pp. 13–18.
[87] *Skype*. https://www.skype.com/en/.
[88] Gephi.org. *Gephi*. https://gephi.org/. 2016.
[89] Nizar Kheir and Xiao Han. "Peerviewer: Behavioral tracking and classification of P2P malware". In: *Cyberspace Safety and Security*. Springer, 2013, pp. 282–298.
[90] Wikipedia. *Wireshark — Wikipedia, The Free Encyclopedia*. 2016. URL: %5Curl%7Bhttps://en.wikipedia.org/w/index.php?title=Wireshark&oldid=750021585%7D.
[91] Fariba Haddadi and A Nur Zincir-Heywood. "Data confirmation for botnet traffic analysis". In: *International Symposium on Foundations and Practice of Security*. Springer. 2014, pp. 329–336.
[92] Wikipedia. *C4.5 algorithm — Wikipedia, The Free Encyclopedia*. [Online; accessed 12-Nov-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=C4.5_algorithm&oldid=727181226.
[93] *Weka 3: Data Mining Software in Java*. http://www.cs.waikato.ac.nz/ml/weka/.
[94] Fariba Haddadi and A Nur Zincir-Heywood. "Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification". In: ().

[95] *Detecting Cyber Malicious Command-Control (C2) Network Traffic Communications*. https://www.eleceng.adelaide.edu.au/students/wiki/projects/index.php/Projects:2015s1-04_Detecting_Cyber_Malicious_Command-Control_(C2)_Network_Traffic_Communications.

[96] Sebastian Abt, Sascha Wener, and Harald Baier. "Performance Evaluation of Classification and Feature Selection Algorithms for NetFlow-based Protocol Recognition". In: *GI-Jahrestagung*. 2013.

[97] Wikipedia. *Cross-validation (statistics) — Wikipedia, The Free Encyclopedia*. 2016. URL: %5Curl%7Bhttps://en.wikipedia.org/w/index.php?title=Cross-validation_(statistics)&oldid=747269850%7D.

[98] Wikipedia. *Naive Bayes classifier — Wikipedia, The Free Encyclopedia*. [Online; accessed 1-November-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=747258563.

A Appendix A

In this work, two popular classification methods have been used, i.e. Naïve Bayes and Decision Tree.

Naïve Bayes classification [98] Naïve Bayes classifiers are simple probabilistic classifiers based on Bayes' theorem with strong independence assumptions between the features [98]. It is a conditional probability mode represented by vector $\mathbf{x} = (x_1, \ldots, x_n)$ with n features (independent variables), it assigns to this instance probabilities $p(C_k|x_1, \ldots, x_n)$ for each of K possible outcomes or *classes* C_k.

Using Bayes' theorem, the conditional probability can be decomposed as: $p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}$.

This is same as: $\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$

C4.5 Decision Tree classification [92] Decision Tree learning uses decision tree as predictive model and is a rule based classifier. In this tree structure, leaves represent class labels and branches represent conjunctions of features that lead to those class labels [92].The goal is to create a model that predicts the value of a target variable based on several input variables. C4.5 is an algorithm used to generate a Decision Tree. C4.5 builds Decision Trees from a set of training data using the concept of information entropy.

The training data is a set $S = s_1, s_2, ...$ of already classified samples. Each sample s_i consists of a p-dimensional vector $(x_{1,i}, x_{2,i}, ..., x_{p,i})$, where the x_j represent attribute values or features of the sample, as well as the class in which $s_i$falls. At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

J48 is an open source Java implementation of the C4.5 algorithm in the Weka data mining tool.

B Appendix B

Metric for performance evaluation: When dealing with Dataset that is mixture of more than on one class, then class wise performance evaluation is done. Following are the important metrics considered when dealing with classification performance evaluation:

True Positive Rate True Positive Rate (TPR) means ratio of correctly classified instances under each class.

$$TPR_c = \frac{TP_c}{TP_c + FN_C}$$

Here TPR_c is the class c True Positive Rate (TPR) and TP_c and FN_c are the True Positive and false Positive counts for c class.

False Positive Rate False Positive Rate (FPR) means ratio of incorrectly classified instances under each class.

$$FPR_c = \frac{FP_c}{FP_c + TN_C}$$

Here FPR_c is the class c False Positive Rate (FPR) and FP_c and TN_c are the False Positive and True Negative for c class.

Detection Rate Class wise Detection Rate (DR_c) is similar to TPR_c. To add up the class wise detection rate of classifier across all classes, average Detection Rate is calculated. This is denoted as:

Score = $\frac{1}{|c|} \sum_{c \in C} DR_c$

Interpretation of TPR, FPR, FNR etc. In an ideal scenario, High TPR and Low FPR is desirable. E.g. From bot class point of view, High TPR means more botnet traffic has rightly been detected. High FPR means, high number of traffic has been classified as bot traffic but actually they are not. High FNR means right botnet traffic could not be detected as botnet traffic by the classifier, rather the traffic has been detected as normal traffic.

C Appendix C

Selected Flow attributes			
Haddadi[91]		Zhao [15]	
%td	Duration	SrcIp	Flow source IP address
%sas	Source AS	SrcPort	Flow source port address
%das	Destination AS	DstIp	Flow destination IP address
%in	Input Interface num	DstPort	Flow destination port address
%out	Output Interface num	Protocol	Transport layer protocol or 'mixed'
%pkt	Packets	APL	Average payload packet length for time interval.
%ipkt	Input Packets	PV	Variance of payload packet length for time interval.
%opkt	Output Packets	PX	Number of packets exchanged for time interval.
%byt	Bytes	PPS	Number of packets exchanged per second in time interval T
%ibyt	Input Bytes	FPS	The size of the first packet in the flow.
%obyt	Output Bytes	TBP	The average time between packets in time interval.
%fl	Flows	NR	The number of reconnects for a flow
%tos	Tos	FPH	Number of flows from this address over the total number of flows generated per hour.
%stos	Src Tos	-	
%dtos	Dst Tos	-	
%smk	Src mask	-	
%dmk	Dst mask	-	
%fwd	Forwarding Status	-	
%svln	Src vlan label	-	
%dvln	Dst vlan label	-	
%bps	bits per second	-	
%pps	packets per second	-	
%bpp	Bytes per packet	-	

Table C.1: Feature Set: Haddadi vs Zhao

D Appendix D

Program Name	Time window (in mins)	Number of unique ASNs
Category: Non-P2P bots		
Zeus_1	60	1
Zeus_2	60	1
Zeus_3	60	1
SpyEye_1	60	4
SpyEye_2	60	1
SpyEye_3	60	1
Citadel	60	9
Category: P2P bots		
Zeus Gameover	45	21
NSIS.ay(P2P)	60	573
Category: P2P benign		
benign μTorrent	60	812
benign Frost-wire	30	234
benign Skype	30	73

Table D.1: ASN distribution