Network Based Peer-To-Peer Botnet Detection

Yonas Alehegn[1], Dr. T. Pandikumar[2], Abdulkadir Hassen[3]

[1]*Information System Security Office, Bank of Abyssinia*
[2] *Department of CIT, College of Engineering, Defense University, MoD, FDRE*
[3] *Department of ICT, College of Engineering, Defense University, MoD, FDRE*

---***---

Abstract - *Attacks such as spam, DDoS and phishing are common problems on the Internet nowadays. In the past, attackers tended to use a traditional botnets that used a central communication architecture where all bots connect to command and control servers. In recent years, peer-to-peer (p2p) structured botnets have emerged as a new advanced form of botnets. Compared with traditional botnets, p2p botnets are difficult to detect, because they have no single point of failure. The objective of this research is to study and implement network based peer-to-peer botnet detection by means of network flow analysis. This helps to detect individual peer-to-peer bots in a network. A detection algorithm, which is based on behaviors that isolate malicious from legitimate p2p traffic, is proposed to detect p2p botnet in a live netflow data. These behaviors were identified by analyzing the behaviors of two legitimate p2p applications and Zbot p2p botnet. After the implementation of detection algorithm, the evaluation of result shows 0% false-alarm rate.*

Key Words: Botnet, Netflow, Nfdump, Nfsen, P2P Botnet, Zbot

1. INTRODUCTION

We live in a world where computer technology is ever growing at an exponential rate every two years, according to Moore's law. Besides from computer technology ever growing, security is known to fall behind. The security experts at Mandiant [10] call this the "Security Gap", where criminals are always one step ahead of the security experts. Computers can also be used in a more sinister manner; criminals can use computers to extract money and information out of businesses and computer users. They can use software known as Botnets to accomplish these goals. A botnet is a collection of bots typically controlled by a bot master. A bot is a piece of software that conceals itself on a computer system acting on instructions received or programmed by the bot master(s). Botnets are becoming more elaborate and efficient over time and thus the use of Botnets is growing at an exponential rate, threatening the average user and businesses alike [1].

Botnets may be used in a wide range of applications, including malicious and benign ones. For instance, [13] mentioned one of the original botnets, Eggdrop (1993), was developed to facilitate the Internet Relay Chat (IRC) management. However, typical applications of botnets include Distributed Denial-of-Service (DDoS) attacks, identity theft, proxy, spreading of malware or spamming.

The typical botnet consists of a bot server (C&C server controlled by botmaster) and multiple botclients. Botclients are also referred to as zombies or drones. Botnets are problem which is spread across the Internet, spans individual administrative domains and therefore, a problem that requires a solution which scales for whole Internet [11].

According to many research papers on botnets, botnets can vary from networks of just few infected machines up to tens of thousands. The shape of such networks varies in the same way. As [7], traditionally botnet administrators manage their bots through central Command and Control (C&C) servers. A central architecture has a number of disadvantages. The C&C servers can easily be identified and there are organizations that actively keep track of C&C servers on the Internet. C&C servers of large botnets are often taken down by the police in order to dismantle the botnet. In response to this, some botnets now use a peer-to-peer architecture for their communications.

2. BACKGROUND

2.1 Peer-to-Peer Botnet

Peer-to-Peer botnet is a new approach of botnet that use the advantage of P2P technology to accomplish a certain task. The idea of P2P botnet is that all the bots connect & communicate with to each-other in order to remove the need for a centralized server.

In P2P, each node acts as client-server which provides bandwidth, storage and computational power. Using this approach, bots are able to communicate with each bots by downloading files or commands from other bots' machines and performing different activities. In comparison to IRC structures, everyone can join a peer-to-peer network, thus the more peers acting as bots, and the more powerful the botmaster can be. In addition, it will be hard to detect and shut down the botnet as security people would need to isolate each machine [20]. Fig -1 shows the P2P botnet operation.

There are different types of p2p botnets nowadays which uses for many illegal activities. Among them, the new variant of Zbot is one of the dangerous p2p botnet. Zbot p2p malware was selected for this study due to it's in the list of top dangerous malwares [2].

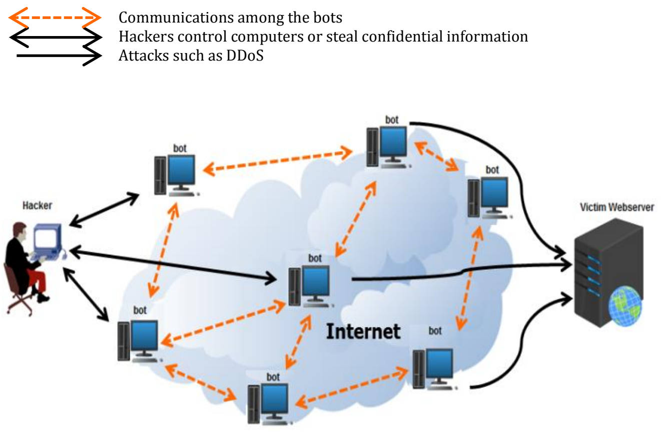

Communications among the bots
Hackers control computers or steal confidential information
Attacks such as DDoS
Hacker
bot
Victim Webserver
Internet

Fig -1: P2P botnet operation

2.1.1 Zbot

Zbot is a malware which infects Windows users and tries to retrieve confidential information from the infected computers. Once it is installed, it also tries to download configuration files and updates from the Internet [2]. It has been designed primarily to steal confidential information from the computers it compromises. It specifically targets system information, online credentials, and banking details, but can be customized through the toolkit to gather any sort of information [18]. Zbot spreads mostly via email but can also utilize autorun capabilities of removable media, or install via a drive-by infection when the user visits a compromised or malicious webpage [12].

Zbot have introduced the concept of p2p C&C network [12]. Unlike the earlier variant of Zbot which uses a central C&C network, the recent variants of Zbot uses decentralized network for their operation. Once a system is infected with this p2p variant of Zbot, each infected system is capable of communicating to any other infected system and is capable of receiving and passing on commands, updates and malware downloads to other infected systems.

2.2 NetFlow

NetFlow is a traffic profile monitoring technology developed by Darren Kerr and Barry Bruins at Cisco Systems, back in 1996. NetFlow data provides important information about network conversations and behaviors. Each unique flow is recorded by the network devices or probes, and the flows are then reported to a data collection server [16]. To monitor suspect Internet botnet's activities by analyzing the source data from the router, NetFlow is helpful for network managers. The data that generates from NetFlow represents the information gathered from the network by sampling traffic flows and obtaining information regarding source and destination IP addresses and port numbers.

Several different formats for flow records have evolved as NetFlow has matured. Since the first development of NetFlow at Cisco, multiple versions were introduced, but not all of them released. Only two versions, NetFlow v5 and NetFlow v9, are more popular.

Version 5 of NetFlow protocol was (and still is) very common protocol for exporting NetFlow data. V9 is one of the most recent versions of NetFlow record [11]. It is template based, providing extensible design but not widely used in the enterprise network as v5. V5 is the most common and usable protocol in different enterprise networks. Therefore, we will use the NetFlow v5 protocol for this research.

3. RELATED WORK

Peer-to-Peer botnet have recently recognized as one of the most threats to the Internet security. Many researches have been conducted to analyze and detect peer-to-peer botnets and the results of these researches contributed to security enhancement and draw new idea on strengthening the protection of botnets from propagating to network. Some of these researches that are related to ours are [14], [7], [5] and [8].

P. Narang, et al. [14] explains about PeerShark, a novel methodology to detect P2P botnet traffic and differentiate it from benign P2P traffic in a network. Instead of the traditional 5-tuple flow-based detection approach, they use a 2-tuple conversation based approach which is port-oblivious, protocol oblivious and does not require deep packet inspection (DPI).

In [7], the author discussed about the detection of individual p2p bots within a network perimeter. The work is done by looking at the communications with their p2p overlay network. The author used NetFlow protocol to gain insight in all traffic within the network. The study was analyzed and test GameOver Zeus p2p malware. In this work, the experiment has a limited access to the external network. Therefore, there were no incoming requests from outside the network. Our work provides a solution for some limitations of this work and considers its future direction.

On the explanation of C. Rossow, et al. [5], a formal graph model to capture the intrinsic properties and fundamental vulnerabilities of p2p botnets are presented. The authors have applied their own model to current p2p botnets to assess the botnets resilience against attacks.

D. Zhao, et al. [8] presents an approach to detect p2p botnet activity by classifying network traffic behavior using machine learning classification techniques. In this research, the authors have been studied the feasibility of detecting botnet activity without having seen a complete network flow by classifying behavior based on time intervals and they have examined the performance of two popular classification techniques with respect to this data.

Since p2p botnets have a distributed architecture, which make them more robust, the detection method should focus on individual p2p bots. This can be done by observing at the communications with their p2p overlay network using the NetFlow protocol. Therefore, our study focuses on this method.

4. METHODOLOGY AND SAMPLE DATA SELECTION

In this section, we will discuss how our work is organized. We provide a detailed description of the materials and methods used to get our results. The next four sub-sections discuss the experimental setup of our study and the tools that we used and the sample data selections of this work.

4.1 Experimental Setup & Tools

Today, any research experiments involving advanced malware and p2p botnets should ensure that the malware or bot does not unintentionally infect the computer outside the experimentation setup. Failure to do so have huge implications on the security and privacy of the concerned persons and organization. Hence virtual machines are preferred to real world machines so that we would be conduct the experiment in a controlled and secure manner. In this experimentation setup, we have made use of VMware [19] to create virtual machines representing real world hosts and GNS3 [9] to connect the virtual machines (VM)together, to monitor and export netflow data. The tools and experimental settings used in this work are listed below.

- **VMware Workstation Pro**: Four virtual machines were created. Ubuntu Server 14.04 LTS installed on the first VM. Windows XP SP3 installed on the other three virtual machines. Traffic flows were collected and observed on this environment. The system setup of these virtual machines provided in Table 1.

Table -1: System Setup of Virtual Machines

	VM1 (NFC)	VM2 (PC1)	VM3 (PC2)	VM4 (PC3)
Operating System	Ubuntu Server 14.04 LTS (Linux)	Windows XP	Windows XP	Windows XP
Memory	2048 MB	256 MB	256 MB	256 MB
Hard disk	150 GB	40 GB	40 GB	40 GB

- **GNS3**: is a software emulator for networks that allows the combination of virtual and real devices to simulate complex networks. We used this to create a network for the virtual machines in order to enable the NetfFlow cache and export the NetFlow data to the collector. We used Cisco's 7200 router IOS to configure NetFlow and NDE. The screenshot of network setup is shown in Fig - 2.

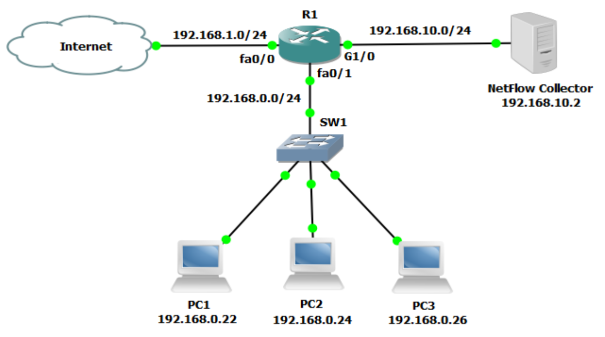


Fig -2: Network Setup

- **Nfdump**: is a set of tools to collect and process NetFlow data. It's fast and has a powerful filter pcap like syntax. It supports NetFlow versions v1, v5, v7, v9 and IPFIX as well as a limited set of sflow. We used this tool to collect and process the exported NetfFlow data from a router IOS.

- **NfSen**: is the web based front end for the Nfdump NetFlow tools. We used this tool to display the NetFlow data in a graphical form within a specific period or protocol.

4.2 NetFlow Data Exporter (NDE)

In this section, we will describe the configuration of Cisco 7200 router IOS on GNS3. A router R1 (rtr-nde) was configured to send NetFlow data to our NetFlow collector. For this experiment, the most important source of statistic was a router which forwards all the traffic on the network. The configuration of Cisco Netflow and NDE was done on a router "rtr-nde" according to [17]. We configured the router to capture the netflow data from the interface fa0/1 and then send these data to the netflow collector (NFC).

4.3 NetFlow Collector

As the name suggests, a NetFlow collector is simply a device that gathers network statistics/data using Cisco's NetFlow network protocol [15]. Depending on how the network is designed and also the size of the network, NetFlow collectors can be one or more. A NetFlow collector can be a device such as a switch, router, server, or even a workstation. In order to test our detection technique, we used one server for NetFlow collector as on Fig -2 shown. NetFlow collectors store data according to guidelines configured by an administrator, so that the data can be accessed for analysis. Depending on the software used, the collection process can be initiated by a couple of commands.

Our study utilizes Nfdump as the flow tool. We used the current version that is nfump-1.6.13. Two of the reasons why this tool was selected are:

- Nfdump is distributed under BSD license
- Nfdump tools support multiple versions of NetFlow: v5, v7 and v9

NfSen, a graphical web-based front end for the nfdump netflow tools is also used. We are using NfSen for displaying NetFlow data and processing data within the specific time periods, and alerting based on various conditions. NfSen has the possibility to extend its functionality by using plugins which we leverage for p2p botnet detection. It is released and distributed under BSD license so we can use it as well as nfdump. For our study purpose, we have modified the Perl and php scripts of the existed NfSen file.

4.4 Sample Data Selection

For experiments to be conducted different legitimate p2p applications and live p2p malware are needed. The sample data sets need to be in the following types.

- **Legitimate p2p file sharing program:** These are legitimate programs such as uTorrent, and eMule.
- **Live p2p malware:** This is a malware in the wild that can generate malicious traffic.

4.4.1 Legitimate traffic

In order to conduct this research, legitimate traffic was generated from Windows XP testing machines in the experiment. The dataset comprises web traffic generated by manual activities. These includes data stream from YouTube, Internet radio, company websites and web-applications. The dataset also includes p2p traffic generated from two file sharing applications on different p2p networks. The overview of the p2p applications are depicted in Table -2.

Table -2: Legitimate p2p applications

Application	Description
eMule	A free peer-to-peer file sharing application for Microsoft Windows.
uTorrent	A program that uses the bittorrent protocol to share files.

4.4.2 Malicious traffic

For our research, one binary of Zbot p2p malware was obtained from public malware site [3]. Its md5 binary is 3d6046e1218fb525805e5d8fdc605361. This malware was downloaded and extracted on windows XP virtual machine in our research experiment. Traffic generated from this machine were collected and stored in netflow collector as nfdump binary format.

5. DATA ANALYSIS AND IMPLEMENTATION

5.1. Data Analysis

This section shows the data analysis of the experiment. Legitimate normal & p2p traffic, and p2p malicious traffic was generated from the three testing PCs in a virtual environment and observed their traffic behavior for more than seven hours in different scenarios. Each scenario has taken one hour. A one hour portion size was chosen, because the Zbot malware generates relatively little traffic. It must thus be monitored for a longer time before its behaviors become visible. More than one hour would unreasonably increase the amount of data that needs to be processed when working with flow data in real time.

At the beginning of the analysis, we used PC1 to produce normal traffic, PC2 to be infected by Zbot p2p malware, and PC3 to generate legitimate p2p traffic. This helps us to identify the behavior of each traffic variants. And gradually, we have infected all three testing machines with Zbot and observe the network flow data.

From this analysis, we have observed the network flow data for more than seven hours. During this time, more than 72,000 flows were generated. The following charts & figure: Chart -1, Fig -3 and Chart -2 shows the whole graph that was displayed on NfSen web during the analysis, the top three IP addresses and the total amount of flows that was generated from each scenario respectively.

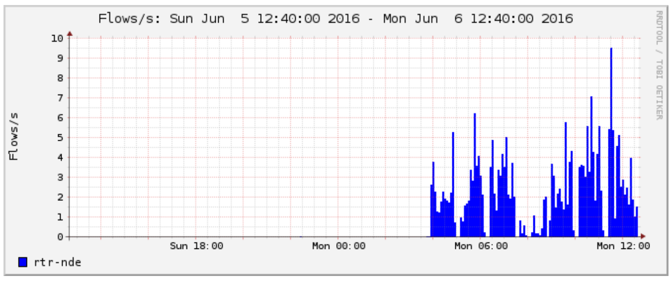
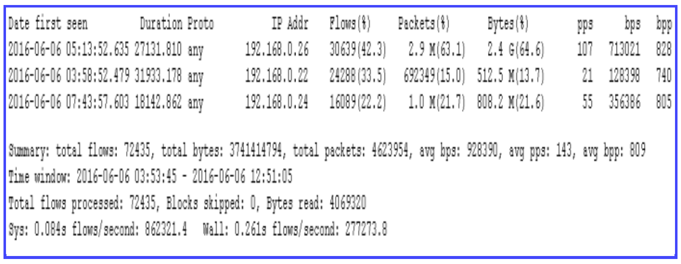


Chart -1: The whole graph displayed on the web during the analysis



```
Date first seen          Duration Proto      IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps      bps   bpp
2016-06-06 05:13:52.635 27131.810 any     192.168.0.26  30639(42.3)    2.9 M(63.1)    2.4 G(64.6)     107   713021   828
2016-06-06 03:58:52.479 31933.178 any     192.168.0.22  24288(33.5)  692349(15.0)  512.5 M(13.7)      21   128398   740
2016-06-06 07:43:57.603 18142.862 any     192.168.0.24  16089(22.2)    1.0 M(21.7)  808.2 M(21.6)      55   356386   805

Summary: total flows: 72435, total bytes: 3741414794, total packets: 4623954, avg bps: 928390, avg pps: 143, avg bpp: 809
Time window: 2016-06-06 03:53:45 - 2016-06-06 12:51:05
Total flows processed: 72435, Blocks skipped: 0, Bytes read: 4069320
Sys: 0.084s flows/second: 862321.4   Wall: 0.261s flows/second: 277273.8
```

Fig -3: Top 3 IP address ordered by flow

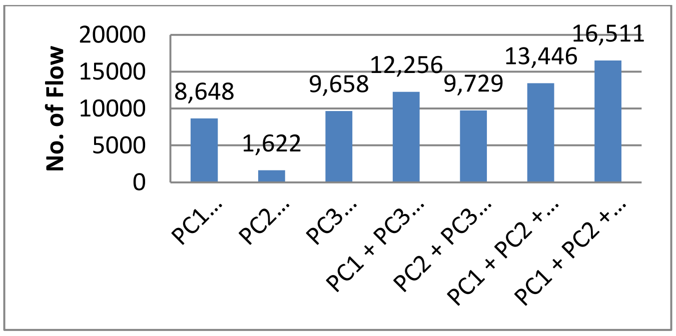


Chart -2: Total amount of flow generated from each scenario

5.1.1 Findings of the Analysis

Based on the data that we got during this analysis, we can easily identify the difference between normal traffic and p2p traffic. Normal or non-p2p traffic which was generated primarily from PC1 has low traffic than p2p traffic that was generated from PC3. This is because p2p traffic generates a high traffic volume when downloading or uploading files.

Zbot infected PC, which is PC2, generates traffic within non-fixed time interval. As the behavior of Zbot, it communicates with peers and updates itself from other peers in every 15 up to 25 minutes. From our experiment, a PC infected with this malware generated malicious traffic three times in one hour duration. This indicates that there were communication &/or updates among PC2 and other peers outside the network.

As we have seen above, when we run PC2 together with other PCs or run all testing machines at the same time, we can't identify the Zbot communication separately from the total netflow data or graph on NfSen web. But when we process the netflow which generates from PC2 separately, we can look its data on the NfSen web in text format. So, we can verify that it is running on the network.

One of the differences between Zbot and legitimate p2p applications is traffic volume. Zbot generates very little traffic when we compare with legitimate p2p application. Legitimate p2p applications are generally used for file sharing and thus exchange a huge amount of data, especially with multimedia files. The average bytes per packet (avg bpp) of the legitimate and malicious traffic that we got from the above analysis were 811 and 834. Chart -3 shows this data. But it is clear that relying on the traffic volume alone for detection would result in false positive.

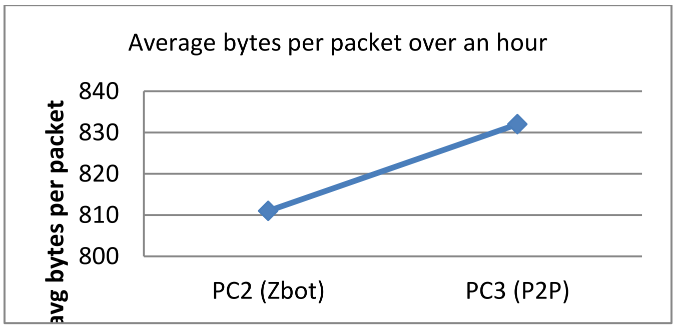


Chart -3: Traffic volume

The other difference is connection failure. From the analysis, we found that three and more number of failed connections occurred on each malicious and legitimate P2P traffic. This was happened cause of peers unreachability due to many reasons such as peers may or available online or placed on behind a firewall. In the normal traffic which was generated from PC1, we didn't observe zero (no) responses or incoming packet for each flows' outgoing packets. For each outgoing packet, there were almost equivalent incoming packets. This was one difference that we found among non-p2p traffic and p2p traffic in our experiment. Table -3 shows some number of failed connections observed from the analysis.

Table -3: Number of failed connection

Source IP	Destination IP	Outgoing Packets	Incoming Packets
192.168.0.26	195.154.109.X	2	0
	176.103.48.X	2	0
	91.200.42.X	4	0
	77.120.115.X	3	0
192.168.0.24	109.169.93.X	3	0
	23.51.251.X	3	0
	184.168.131.X	4	0

In addition, there is also difference in packet symmetry. Packet symmetry is the relation between the outgoing and incoming packets. Analyzing packet symmetry has shown to be effective for detecting high volume DDoS attacks, as these attacks generate a lot of incoming traffic without outgoing traffic [6]. But it can also be useful for detecting malicious traffic in general [4]. The packet ratio is calculated by dividing the outgoing packets by incoming packets. Protocols that rely on TCP for transmitting packets have a packet ratio close to 1, as packets need to be acknowledged. Packet ratios for UDP protocols can be more varying, because UDP does not provide an acknowledgement mechanism. However, most benign protocols that use UDP implement their own mechanism for acknowledgements to ensure reliability.

Chart -4 shows the measured packet ratio for traffic flow that generate from each PC in the analysis. Both legitimate traffics have a ratio that is between 0.5 and 1.0. A high ratio for p2p applications can be explained by the high number of failed connections, as this result in outgoing packets without incoming packets. The malicious traffic has ratio of 0.26, which is close to 0.3. Zbot has a relatively low amount of failed connections which results in a lower packet ratio. Zbot also does not implement an acknowledgement mechanism. So when it requests data from other peers, it receives a high amount of packets with only a single outgoing request packet. This greatly reduces the packet ratio should be balanced out by requests from other peers, because those will result in more outgoing packets.

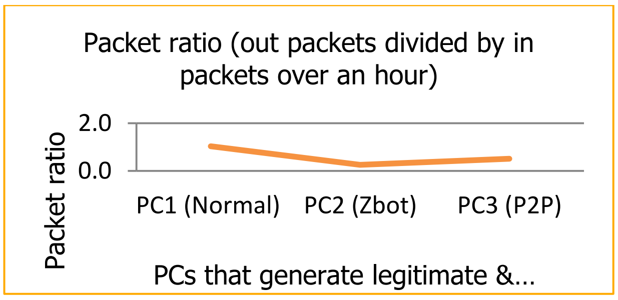


Chart -4: Packet Symmetry

5.2. The Proposed Detection Method and Algorithm

Based on the information gained during studying p2p botnet structures, behavior and gathering of existing approaches, methods for p2p botnet detection were proposed and implemented as described below. Our detection method leverage NfSen plugins.

Plugins allow to extend NfSen for additional functionality. There are two types of plugins: backend plugins and frontend plugins. First type, backend plugins are loaded into background process and can provide several functions. Actions which can be performed are periodic data processing, alerting conditions and alerting actions. Backend plugins must be written as Perl modules.

Frontend plugins, which are second type of plugin available, can display any results of the backend processing. Frontend plugins are PHP scripts. There is a communication channel provided by NfSen for communication between channels. Concept of plugins and their cooperation in NfSen is depicted in Fig -4 [15].

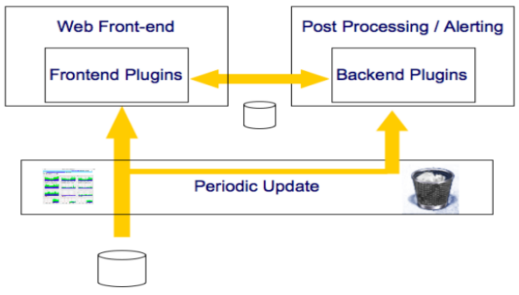


Fig -4: Plugin Concept

For the purposes of our work, we have modified and use the backend and frontend module of NfSen plugin [15] to filter and display suspicious host in our experimental network.

For the detection algorithm, we have used two key differences that we got from the findings of the data analysis, which are the number of failed connection and packet ratio. In the p2p traffic, there were more than 3 failed connections to different destination hosts. We used this difference to filter p2p traffic from non-p2p traffic as it is.

The other one, which is the packet ratio, is close to 0.3 for Zbot infected PC and between 0.5 and 1.0 for the legitimate PCs. In order to increase the detection performance of Zbot infected PC(s) and minimize the possibility of false-negative, we compare the packet ratio of collected traffic to 0.4.

From this ground, we have designed and implemented the detection algorithm which is described below.

1. Get group of flow data
   - Group of flows by source socket and destination socket taken as input
2. Isolate peer-to-peer traffic
   - Source with more than three failed connections to different destination hosts are considered as p2p traffic
3. Detect Zbot p2p botnet based on
   - Packet ratio: If the sum of outgoing packets divided by the sum of incoming packets is less than 0.4, source is detected as bot infected.

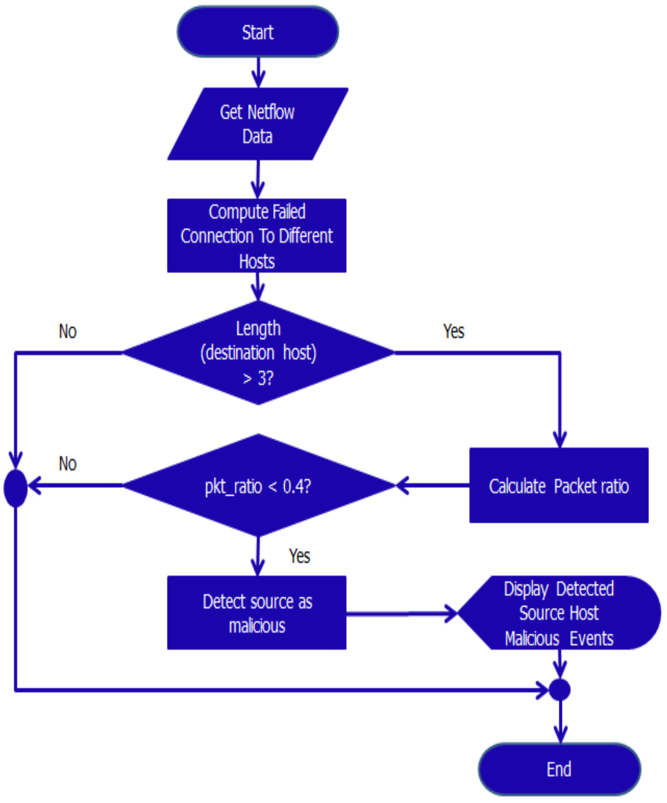

Start
Get Netflow Data
Compute Failed Connection To Different Hosts
Length (destination host) > 3?
No
Yes
Calculate Packet ratio
pkt_ratio < 0.4?
No
Yes
Detect source as malicious
Display Detected Source Host Malicious Events
End

Fig -5: Detection Algorithm Flowchart

5.3 Implementation of the proposed algorithm

The proposed detection method or algorithm was implemented in the backend module of NfSen written in Perl. The backend plugin provides a framework to extend the NfSen usability by adding effective algorithms which can be suitable to detect different malicious activities in a network. Once the nfdump netflow collector deals with live data and keeps a table of flows grouped by source socket and destination socket in memory, NfSen backend plugin uses these data for further processing. When a new flow comes to the plugin, it will filter the p2p traffic from non-p2p traffic. And then, p2p traffic will be processed to identify the Zbot p2p botnet. Once the Zbot p2p malware detected, it displayed the events caused by this malicious traffic on NfSen web, plugin tab.

5.3.1 Isolate P2P Traffic

Get the set of unique destination IPs that have a failed flow. A failed flow is a flow with 1 or more outgoing packets and 0 incoming packets. If the set contains more than 3 IPs, the source is considered p2p.

```
sub p2p_detect{
       my($flows) = @_;
       $noreply = set($flow->{dst_ip}) foreach
my $flow($flows)
       if($flow->{out_pkts} > 0 and $flow-
>{in_pkts}==0)} {
       if(length($noreply) > 3) {
              return True;
              }
              }
}
```

5.3.2 P2P Packet Symmetry Detection

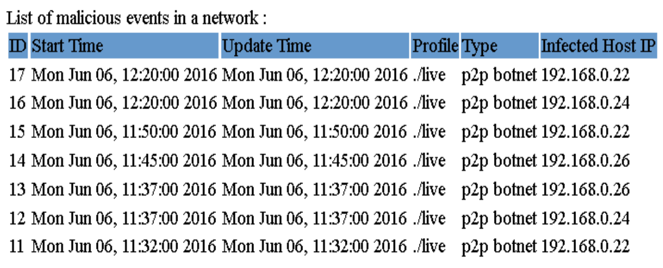
Calculate packet ratio by dividing the sum of all outgoing packets by the sum of all incoming packets. If the ratio is less than 0.4, the source is considered Zbot p2p botnet.

```
sub p2p_ratio_detect{
       my($flows) = @_;
       $out = sum ($flow->{out_pkts} foreach
$flow ($flows));
       $in = sum ($flow->{in_pkts} foreach
$flow ($flows));
       if $out / $in < 0.4 {
              return True;
       }
}
```

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Experimental Results

After the implementation of the proposed detection method, we have done similar activities as we did on the analysis and lastly, we infected all PCs with Zbot malware and made a test of our detection method by running them together at similar time for one hour. During this testing, seven malicious events cause by Zbot from the three PCs were detected and reported on the Events Plugin. These results are shown below on Fig -6.


List of malicious events in a network :

ID	Start Time	Update Time	Profile	Type	Infected Host IP
17	Mon Jun 06, 12:20:00 2016	Mon Jun 06, 12:20:00 2016	./live	p2p botnet	192.168.0.22
16	Mon Jun 06, 12:20:00 2016	Mon Jun 06, 12:20:00 2016	./live	p2p botnet	192.168.0.24
15	Mon Jun 06, 11:50:00 2016	Mon Jun 06, 11:50:00 2016	./live	p2p botnet	192.168.0.22
14	Mon Jun 06, 11:45:00 2016	Mon Jun 06, 11:45:00 2016	./live	p2p botnet	192.168.0.26
13	Mon Jun 06, 11:37:00 2016	Mon Jun 06, 11:37:00 2016	./live	p2p botnet	192.168.0.26
12	Mon Jun 06, 11:37:00 2016	Mon Jun 06, 11:37:00 2016	./live	p2p botnet	192.168.0.24
11	Mon Jun 06, 11:32:00 2016	Mon Jun 06, 11:32:00 2016	./live	p2p botnet	192.168.0.22

Fig -6: Results of detected Zbot p2p botnet from all PCs

6.2 Evaluation of the detection method

We have evaluated our detection approach how it performs accurately its task. The accurate performance of this detection method was measured in terms of its false alarm rate, i.e., false-positive and false-negative rates. A false-positive (FP) is defined as a legitimate host mistakenly identified as Zbot infected host and a false-negative (FN) means Zbot infected host fails to be detected or identified as a legitimate host.

Table -4 depicts our evaluation results where the average number of FP or FN hosts is calculated during the entire period of evaluation. The average FP or FN rate is the number of FP hosts divided by the total number of legitimate hosts, which were 3, or the number of FN hosts divided by the total number of Zbot infected host(s).

Our detection approach performs well in terms of false-positives. It identifies all Zbot infected hosts, which means 100% and almost didn't detect uninfected hosts wrongly. The highest false-positive rate was 33%. There was only PC3 which detected mistakenly as Zbot infected host when all PCs run together. We verified that this PC didn't have malicious activity and failed to form suspicious filtering. When the number of Zbot infected PCs increased, the false-positive rate becomes 0%.

Table -4: False alarm detection rate

Traced Testing Machines (VMs)	Average FP	Average FN	Duration	Description
PC1	0	0	1 HR	Legitimate non-p2p traffic was generated
PC2	0	0	1 HR	Traffic from Zbot infected PC was generated
PC3	0	0	1 HR	Legitimate p2p traffic was generated
PC1 & PC3	0	0	1 HR	Legitimate non-p2p & p2p traffic were generated
PC2 & PC3	0	0	1 HR	Legitimate & malicious p2p traffic were generated
PC1, PC2 & PC3	0.33	0	1 HR	PC2 was the only infected by Zbot
PC1, PC2 & PC3	0	0	1 HR	All PCs were infected by Zbot

7. CONCLUSION

P2P Botnets are considered as the biggest threat to the internet security today. Day by day millions of computers are compromised on the internet. In this research, we have studied and implement network based solution for Zbot p2p botnet. Our contribution might be nothing for the internet security but has a good significance for the small organizations and home network which has no firewall or no more security to detect this p2p malware. Our contributions in this work can be summarized as follows.

1. Proposed and implement a detection method for Zbot malware.
2. Extend the netflow collector functionality by incorporating our detection approach on the existed plugin.

As the previous discussion on the analysis, Zbot behaves differently from legitimate p2p applications. Number of failed connection and packet symmetry were the two main differences that we found from the analysis. So, based on these findings, the algorithm has designed to detect Zbot malware. First, number of failed connection used to filter the p2p traffic from non-p2p traffic. This reduces the scope of the traffic and the chance of false positive. Then, the output of packet symmetry used to filter and detect the Zbot malicious event. This algorithm has implemented on the netflow collector plugin.

The detection method implemented as a result of this research verified that it can able to detect malicious events in a network. The result was displayed on NfSen web plugin tab. To determine how it performs well, its performance was evaluated by measuring in terms of false-alarm rate. From the evaluation, its false positive rate was 0% when three machines were running after infecting them and the false negative rate was 0%.

To sum up, we put two future research directions which needs a more extensive investigation from the point of this work. This study was made on and evaluated by limited number of testing machines. Evaluating the performance of this detection approach with a high number of nodes in a real network will be interesting. In addition, the scope of this study is limited to detection. Therefore, study and implementation of a prevention mechanism for Zbot and other p2p botnet with a similar behavior needs further investigation.

REFERENCES

[1] A. Shaikh, "Botnet Analysis and Detection System," Nov. 2010.

[2] A. Zaharia, "The Top 10 Most Dangerous Malware That Can Empty Your Bank Account," Blog, 2016.

[3] Avcaesar. Retrieved from https://avcaesar.malware.lu/

[4] C. Kreibich, A. Warfield , J. Crowcroft, S. Hand, I. Pratt, "Using packet symmetry to curtail malicious traffic," (n.d.).

[5] C. Rossow, D. Andriesse, T. Werner, B. Stone-Gross, D. Plohmann, C. J. Dietrich, & H. Bos. "Sok: P2PWNED-modeling and evaluating the resilience of peer-to-peer botnets," 2013.

[6] C. Dillon. & M. Berkelaar. "OpenFlow (D)DoS Mitigation," Feb. 2014.

[7] C. Dillon. "Peer-to-Peer Botnet Detection Using NetFlow," Jul. 2014.

[8] D. Zhao, I. Traore, A. Ghorbani, B. Sayed, S. Saad, W. Lu, "Peer to Peer Botnet Detection Based on Flow Intervals," (n.d.).

[9] Graphical Network Simulator-3. (Documentation). Retrieved from https://www.gns3.com/support/docs/

[10] Mandiant. M-trends. "Attack the security gap, Threat Report, 2013.

[11] M. Benes. "Botnet detection based on network traffic classification," 2015.

[12] McAfee. "McAfee Labs Threat Advisory PWS Zbot," 2014.

[13] P. Marques. "Botnet Detection Using Passive DNS," 2013/2014.

[14] P. Narang, S. Ray, C. Hota, & V. Venkatakrishnan, "PeerShark: Detecting Peer-to-Peer Botnets by Tracking Conversation," IEEE Security and Privacy Workshops. 2014.

[15] REN-ISAC Projects, 2013. Retrieved from https://code.google.com/p/renisac/wiki/

[16] S. Choudhary & B. Srinivasan, "Usage of Netflow in Security and Monitoring of Computer Networks," International Journal of Computer Applications, Vol.68, No.24. 2013.

[17] Solarwinds. "How-To Configure NetFlow v5 & v9 on Cisco Routers," TechTips, (n.d.).

[18] Symantec. Trojan.Zbot. Retrieved from https://www.symantec.com/security_response /writeup.jsp?docid=2010-011016-3514-99

[19] VMware. VMware Workstation 12 Pro Documentation. (n.d.).

[20] Y. Al-Hammadi & U. Aickelin, "Behavioral Correlation for Detecting P2P Bots," (n.d.).