Fast Deployment of Botnet Detection with Traffic Monitoring

Chung-Huang Yang  
Nation Kaohsiung Normal University  
chyang@nknucc.nknu.edu.tw

Kuang-Li Ting  
Nation Kaohsiung Normal University  
tiserle@gmail.com

Abstract

With the Internet crime growing, such as phishing, money mules, personal data stealing and trafficking, DDoS (Distributed Denial of Service), and other cases often heard by people from time to time. DDoS mostly uses botnet as source of attack, and distributes trojans and worms to infect hosts. Infected hosts become bots, and could be controlled by the botmaster. Botmaster uses command and control server to control bots. Because botmaster servers use dynamic types and encryption methods to communicate with bots, it's difficult to detect bots.

In this research, we designed and developed a system to detect bot-like traffic and deny traffic of who looks like bots. We revised the NTOP program and integrated it with self developed perl programs. Our system will monitor the network layer and transport layer on network activities and send email/SMS to the network administrator to block suspicious botnet.

1. Introduction

With the progress of network bandwidth and computing, distributed computing is increasingly used widely. However, the concept of distributed computing is also used by hackers to utilize the distributed framework to do attack. These type of distributed attacks are increasing in recent years.

In 2003, hackers of Oregon State of U.S. controlled 20,000 botnet hosts and started a DDoS (Distributed Denial of Service) attack on eBay [9]. In 2005, a new type of botnet network virus Zotob started its DDoS to attack many websites of U.S. famous companies [18]. In Feb. 2006, Criminal Investigation Bureau and Microsoft found out that hosts been infected by botnet numbered around 57,783 and up to 88,136 until Sep. 17 after a thorough check. Within only half a year, the number of new infected hosts was up to 30,000 [16].

The statistics from FBI showed that over 75% of network frauds was conducted through junk mail, and the total amount of frauds in 2007 was over 239 millions [14].

According to Ramachandran and Feamster, spam is always sent by botnet [11]. Currently, DDoS mainly used botnet to increase its attacking sources; besides, it has already become the largest source of junk mail. Botnets not only pose a great threat to network security but also cause the waste of network resources. With the feature of being difficult to be detected and prevented, the controllers of botnet use many different methods such as IRC (Internet Relay Chat) or IM (Instant Messaging) or P2P (Peer to Peer) to do communication and encryption for increasing the difficulty of being detection.

In this research, we revised the NTOP [8] open source program, and conducted the botnet detection on network activities. In order to simplify the detection method, we did not monitor the 5th, 6th and 7th layers of 7 layers of OSI (Open System Interface) model because it is harder to monitor these higher levels and it needs more resources and time to get a satisfied result. Besides, if there is an encryption during the communication process, it will take huge amount of time to do it. Therefore, we only monitor on the network layer and the transport layer and used the features of botnet activity to detect whether it exists or not. Then we develop a system to block suspicious network activities to avoid the expansion of disaster and reduce the number of victims. Also, we inform these victims through administrators and do follow-up actions.

2. Botnet Detection

In 1993, Pointer used TCL language to design the Eggdrop program for administrators to use. However, hackers also used the same thinking to design a botnet program for private purposes [1]. In 1999, after version 2.1 of the SubSeven backdoor, Mobman started to use IRC protocol to construct the channel for attackers to control botnet hosts [3,15], which is the first representative botnet program. Hereafter, the programs GTBot, Sdbot, etc. were widely spread [10], hence programs using IRC protocol was starting to become a mainstream.
In 2003, worm technology had become increasingly mature and botnet program also took advantage of this trend. It used the active transmission technology to do a large amount of spread rapidly, and one famous program was Deloder [10]. P2P botnet appeared in 2004. The program itself included the client of P2P and linked to servers adopting Gnutella, and used WASTE [5] to do communication. It allowed every botnet host to easily find out other botnet hosts and do communication [17].

Botnets can be divided into the following types based on its feature: random, centralized, and P2P [2], and therefore they will have different methods of detection and block. Botnet brings about a great danger and threat, and plenty of experts have proposed some methods to detect the activities and spread of botnet in the light of its features and activity type.

Chi and Zhao [2] proposed to use ID on Router to identify botnets and also use IDS (Intrusion Detection System) to determine whether there is a botnet attack in the route so that administrators can proceed follow-up actions. Zou and Cunningham [19] discovered that it is easy for hackers to know that they are detected when using Honeypot to detect botnet, therefore they proposed to use double honeypots to make the honeypot implanted bot program first to infect the second one and report to hackers; this way, they will not know the computer they implant bot program successfully is honeypot and a type of worm can be used to construct P2P botnet to prove that their method is able to reduce the possibility that hackers find out the honeypot.

Villamarin-Salomon and Brustoloni [15] found out that bots will access DDNS (Dynamic DNS) Server rapidly, therefore we could check out if there is NXDOMAIN (the domain does not exist) by which botnet could be detected. Choi, Lee, and Kim [3] discovered that it is easily cracked by hackers if we use the traditional method – monitoring DNS traffic – to detect botnet, therefore they proposed monitoring botnet’s group activity on DNS traffic and they also did test in schools.

Kugisaki, Kasahara, Hori, and Sakurai [7] focused on the traffic of IRC botnet. They observed the ports using IRC and clients linked with server and checked the traffic to see if they were members of botnet. Gu, Zhang and Lee [6] constructed a BotSniffer system to judge if there is botnet through the consistence of the behaviors of botnet and used it in the real world. Strayer, Walsh, Livadas and Lapsley [13] used filtering to check the traffic and set up an unharmlful botnet to verify whether the pipelining system they developed could find out the traffic of botnet effectively.

There are already many botnet detection methods based on the features of botnet. In addition, many methods are effectively in detecting the activities of botnet. However, in fact, in our daily lives, we still often hear the news and information about enterprises or governments losses caused by botnet. The reason is that the above methods are effective, but it is not easy to be deployed. In the light of this, this study tried to develop a new method to detect botnet more easily and reduce the difficulty of system deployment.

3. The Proposed Botnet Detection System

Basic operations of the proposed botnet detection system in shown in Fig. 1. The intended system framework is illustrated in Fig 2.

![Flow Chart of the Proposed System](image-url)

We revised the NTOP [8] program and monitored on the network layers and the transport layers to gathered the data related to botnet, and simplify output of the NTOP “Active TCP/UDP Sessions”, make it more easier to count sessions of each network activity. And we developed a program which is written with Perl and shell scripts to count sessions, alert administrators, and block abnormal traffic.
In the proposed system, it first executes NTOP program, then executes the monitoring program developed in this study, then reads blacklist (the IP list to be blocked) and whitelisted (the IP list not to be blocked). The monitoring program will not put the IP already in the whitelisted in blacklist. As the program is running, if it finds out unusually huge amount of session, it will inform the administrator and automatically put the IP into blacklist, and then block the traffic.

We used Cisco Switch and its SPAN (Catalyst Switched Port Analyzer) function [12] which is the so-called mirror port to intercept the traffic of switch without direct interception of packets. Within this framework, NTOP server only takes charge of the function of packet analysis and does not take charge of the roles of router, bridge or NAT (Network Address Translation). This will simplify these functions to reduce the difficulty of administrators' maintenance and it can avoid the network breakdown or paralysis caused by abnormal server and even greater damage.

After analyzing botnet activities, the following types were obtained:

(1) Bot master gives command to bots through C&C (Command and Control) Server. The features of botnet activity are the same source IP, different ports linking to different target IPs, and the same Port.

(2) Bots links to C&C Server, and updates attack code or downloads attack program. The features of botnet activity are different source IPs, different ports linking to the same target IP, and the same Port.

(3) Bots links to DDNS or DNS Server to query the target IP to attack. The features of botnet activity are different IPs, different ports linking to the same target IP, and the same Port.

(4) When bots attack, the features of botnet activity are different source IPs, different ports linking to the same target IP and the same Port.

Therefore, the proposed system will detect these four types of botnet activity. NTOP itself does not have warning function, hence we used Perl to add such warning function and used Perl Module LWP::UserAgent to capture the information of Active TCP/UDP Sessions in the revised NTOP. We also included shell script to provide the function of active warning so that it is more convenient for administrators to use. The main process of this study is as follows:

First, the system has set the upper limit value of sessions, and administrators can also set their own upper limit value. If session number of host is over the upper limit, the system will block the host automatically and inform administrators the abnormality which needs their actions.

Second, it provides the function of blacklist and whitelisted to prevent that normal behaviors or important linking of servers providing services for the outside from being blocked. Administrators can set their own blacklist which can be also set as the system runs. System administrators can set their own whitelist.

Third, it provides E-mail or SMS (Short Message Service) to actively remind administrators that there are abnormal sessions in the network, and it can inform users about the upper limit of session and provide the links related to information of Host Activity and Active TCP/UDP Sessions so that administrators can grasp the newest network using of the computer rapidly and deal with it based on the information.

4. Implementation Results

NTOP [8] is a network traffic monitoring program. It is developed by C language and could be used in any Unix-like platform. Its expansion is good and it is easy to be integrated with other program. NTOP uses the libraries of libpcap (also an open source software) to capture the network packets. Because NTOP is an open source software, we can re-write it to satisfy our needs.

Besides, it can also use gd and libpng functions to turn data into pie or column chart for clients to view. Administrators can do remote administration on NTOP through the browsers to avoid increasing the burden of system administration and security problem due to the setup of webpage server.

The use of NTOP can be divided into mirror mode or in-line mode. The former is to do monitoring without destroying the network framework, and the latter is to treat the server of executing NTOP as a node in the network, whose flaw is that when the server is abnormal it will cause a breakdown of the network.
NTOP provides these two modes for users to use it based on their needs.

NTOP has been developed over a decade, a rather stable and powerful program. However, if NTOP is used to detect botnet, because the needed functions are not integrated, users cannot find out the functions they need soon. In order to make it more easy to use, this study added a “Botnet Detection” item on the NTOP main menu and gathered the functions related to botnet in this menu to facilitate administrators’ checking and management. The revised menu is shown in Fig. 3.

Local to Local traffic statistics on the new “Botnet Detection” of the revised NTOP is for administrators to check if there is any behavior of large-scale broadcast or multicast in LAN. If plenty of hosts receive packet of the same size, then it may be the behavior of exploration or malicious attack in the LAN. An example of this function is shown in Fig. 4.

Local to Remote traffic statistics is for administrators to check if the remote host is linked to local host and transmits data of the same size. If yes, then it may be that the controller of botnet is controlling the host been implanted the program or that the host been implanted the program is updating the program or querying DNS. An example is shown in Fig. 6.

IP Traffic on the revised NTOP is to monitor various kinds of protocols that botnet often use, such as DNS, HTTP, IRC, P2P, and IM protocols including MSN and AOL. If the bots have consistent behaviors, for example, querying certain DNS record simultaneously causing that several IP have the same traffic at the same time, then it may be the sign of botnet activity. An example of this IP Traffic is shown in Fig. 7.

---

**Fig. 3. New Botnet Detection on the NTOP Menu**

**Fig. 4. Local to Local Traffic**

**Fig. 5. Local to Remote Traffic**

**Fig. 6. Remote to Local Traffic**

**Fig. 7. IP Traffic**

- Host 1: ip addr 140.127.47.186, df size 17.1MByte, df rate 0.0%, dr rate 0.0%
- Host 2: ip addr 140.127.47.85, df size 41.5MByte, df rate 0.0%, dr rate 0.0%
- Host 3: ip addr 140.127.47.20, df size 2.6GByte, df rate 97.0%, dr size 50.5MByte, dr rate 2.0%
- Host 4: ip addr 140.127.47.6, df size 52.5MByte, df rate 0.0%, dr rate 0.0%
- Host 5: ip addr 140.127.47.23, df size 4.8GByte, df rate 0.2%, dr size 3.5MByte, dr rate 0.1%
- Host 6: ip addr 140.127.47.250, df size 2.2GByte, df rate 0.0%, dr rate 0.0%
- Host 7: ip addr 201.120.120.120, df size 13.1MByte, df rate 0.0%
- Host 8: ip addr 140.127.85.85, df size 12.1MByte, df rate 0.0%
- Host 9: ip addr 140.127.23.23, df size 4.8GByte, df rate 0.2%, dr size 3.5MByte, dr rate 0.1%
- Host 10: ip addr 140.127.47.250, df size 2.2GByte, df rate 0.0%, dr rate 0.0%
- Host 11: ip addr 201.120.120.120, df size 13.1MByte, df rate 0.0%
- Host 12: ip addr 140.127.85.85, df size 12.1MByte, df rate 0.0%
- Host 13: ip addr 140.127.23.23, df size 4.8GByte, df rate 0.2%, dr size 3.5MByte, dr rate 0.1%
- Host 14: ip addr 140.127.47.250, df size 2.2GByte, df rate 0.0%, dr rate 0.0%

<table>
<thead>
<tr>
<th>Host</th>
<th>IP Address</th>
<th>Data Sent</th>
<th>Data Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>cc.nhau.edu.tw</td>
<td>140.127.47.186</td>
<td>17.1MByte</td>
<td>0.0%</td>
</tr>
<tr>
<td>cc.nhau.edu.tw</td>
<td>140.127.47.85</td>
<td>41.5MByte</td>
<td>0.0%</td>
</tr>
<tr>
<td>labs.nhau.edu.tw</td>
<td>140.127.47.20</td>
<td>2.6GByte</td>
<td>97.0%</td>
</tr>
<tr>
<td>vrocker.nhau.edu.tw</td>
<td>140.127.47.6</td>
<td>52.5MByte</td>
<td>0.0%</td>
</tr>
<tr>
<td>stream.nhau.edu.tw</td>
<td>140.127.47.23</td>
<td>4.8GByte</td>
<td>0.2%</td>
</tr>
<tr>
<td>sunshine.nhau.edu.tw</td>
<td>140.127.47.250</td>
<td>2.2GByte</td>
<td>0.0%</td>
</tr>
<tr>
<td>ice.nhau.edu.tw</td>
<td>140.127.47.186</td>
<td>17.1MByte</td>
<td>0.0%</td>
</tr>
<tr>
<td>cc.nhau.edu.tw</td>
<td>140.127.47.85</td>
<td>41.5MByte</td>
<td>0.0%</td>
</tr>
<tr>
<td>labs.nhau.edu.tw</td>
<td>140.127.47.20</td>
<td>2.6GByte</td>
<td>97.0%</td>
</tr>
<tr>
<td>vrocker.nhau.edu.tw</td>
<td>140.127.47.6</td>
<td>52.5MByte</td>
<td>0.0%</td>
</tr>
<tr>
<td>stream.nhau.edu.tw</td>
<td>140.127.47.23</td>
<td>4.8GByte</td>
<td>0.2%</td>
</tr>
<tr>
<td>sunshine.nhau.edu.tw</td>
<td>140.127.47.250</td>
<td>2.2GByte</td>
<td>0.0%</td>
</tr>
</tbody>
</table>
Host Activity is to show, with different colors, the activity frequency of the computer by the unit of hours. The darker the color is, the higher frequency of the computer activity is within the period. Colorless represents 0%, light blue represents 0% ~ 25%, light green represents 25% ~ 75%, and red represents 75% ~ 100%. If some hosts have large amount of outside link in the middle of night or within certain period, then it is possible that these hosts have been implanted botnet program. An example of such Host Activity is shown in Fig. 8.

Active TCP/UDP Sessions is used to record data on each network link, including source IP, source port, target IP, target port, data amount of receiving and delivering, the starting and ending time of link, length of link and the packet information of TCP so that administrators can do checking. We revised the items of Active TCP/UDP Sessions and added another item. In the new items, we deleted unnecessary information, kept source IP, source Port, target IP and target Port, and did follow-up monitoring and processing. One example of such TCP/UDP Session is shown in Fig. 9.

5. Conclusion

The number of botnets increase dramatically in recent years. Botmaster controls a botnet to perform various malicious activities. How to detection such botnets becomes an important issue. In this research, we proposed a botnet detection system which is based on the open source NTOP program with self developed perl programs.

Currently, the proposed system is being tested in National Kaohsiung Normal University. The main purposes of this test were to test its stability and verify whether its functions can meet the needs we first set. After the stable operation of this system is confirmed, we will install it in some elementary and junior high schools and all levels of schools in Kaohsiung county and city to verify its effects.

References

[13]. W.T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, “Detecting Botnets with Tight command and...


