Adaptability of IRC Botnet Detection Method to P2P Botnet Detection

Ji, Yuan  
Department of Electrical Engineering and Computer Science  
University of California, Irvine  
yji1@uci.edu

John, Robin  
Department of Electrical Engineering and Computer Science  
University of California, Irvine  
rjohn@uci.edu

Abstract—This report mainly discusses the adaptability of the IRC-based Bot detection method to be used in the P2P-based Bot detection.

The first section introduces the IRC-based bot and the newly appeared P2P-based bot to see their difference. The second section shows the related work and the traditional method of BOTNET detection. The third section discusses the methodology used by the IRC based Botnet propagation. In the fourth section, we give a theoretical analysis and brief proof of the adaptability. The next section covers some of our Experimentation attempts.

We finally analyse the current modules on the Nepenthes platform and touch little details on current Nepenthes methodology and then justify how similar platform with slight addition can be used to track most of the upcoming prospective P2P self propagating malware.

I. INTRODUCTION

A bot is a computer program installed on a compromised PC which offers an attacker a remote control of the target. Botnet, is the networks of several bots under a common control architecture, pose a threat to the Internet, such as Distributed Denial-of-Service (DDoS) attacks, sending spams, sniffing traffic, phishing, etc.

The common architecture of the Internet Relay Chat(IRC)-based bot acts as follows: The attacker creates an IRC server and opens a specific channel in which he posts his commands. Bots connect to this channel, get the command and behave accordingly.

Today, we are encountering a new kind of botnets, that is, the P2P-based botnets. These botnets do not have a central server that distributes commands and are therefore not directly affected by the traditional botnet tracking methods which we will discuss later. The most famous P2P-based botnets nowadays are like Peacomm, Phatbot, etc. These are all the typical storm worms. This malware is currently the most wide-spread P2P bot observed in the world.

II. RELATED WORK

When we go to the botnet detection and tracking, we have to introduce the Honeynet project. A honeynet is a high-interactive network containing a set of honeypots. It could be divided into two levels: 1) The high level honeypot (actual honeypot) and 2) the low level honeypot (virtual honeypot). The main difference between these two is that the virtual honeypot don’t have the system really affected. They simulate the system with some specific software, pretend to be affected and act corresponding to the command. However, this is easily recognized by the attack nowadays and the information it can collect is limited. Compared to this mechanism, the high level honeypot is a real affected machine. Thus is hard to detect and could collect more sensitive information. The price is that we have to have a honeywall to stop the pot from attacking the victim.

Nepenthes is a Honeypot sample platform which emulates the necessary vulnerabilities and stands in between the low level and high level interaction, as in, it combines the scalability of low-interaction honeypots and the expressiveness of high-interaction honeypots.

The purpose of designing a honeynet is to create an interception tool of attacks and collect the attacking information in a highly controlled circumstance. To create such an environment, we have to satisfy two requirements:

- Data control: we have to make sure that the compromised honeypots will not harm any system outside the honeynet, and such control should not be detected by the attacker.
- Data Interception: we have to make sure to collect all the behaviors happening in the honeynet and data going into and out of the honeynet. Of course, the interception should also be transient to the attacker.

When applied to the botnet detection and tracking, we use a honeypot to act as a ”spy” and connect to the IRC channel, the commands from the attacker will be allowed to go into the honeynet, however, when the spy want to react to the command and execute the attack to the victim, the attacking process will be intercepted by the honeywall and thus will not make the actual attack. From the information we have intercept in both directions, we are possibly able to find out the channel and the commander, such as we could analyze the DNS/IP address.
of the channel.

A typical architecture of a honeynet is shown in the following figure:

![Honeynet Topology](image)

**Fig. 1.** A typical Honeynet Topology.

For more detailed information, refer [2], the "Know Your Enemy" series white paper will be a good resource which focuses on the main idea of psychological method of solving the BOTNET problem in the long run by taking an attacker’s perspective of thinking by knowing more about the attacker’s intentions.

### III. IRC-BASED BOTNETS

The most prevalent type of self-propagating malware technique, being used in the past for some time now was the IRC Based Botnet. The methodology here involves first compromising an initial node to be affected by the Bot/malware. This can be established by forcing the recipient to store the bot program either through spam mails or other social engineering method. The compromised node then connects to an IRC Server. Once connected to IRC Server, it opens up a new channel with a name based upon the script written in the Bot module. Once in the channel, it sends message to other currently connected clients to join the channel. Once the number of members in the channel increases beyond a limit specified by the attacker or the bot program, all the clients connected in the channel send a SYN or ACK message which overloads the server. This sudden flooding of messages causes Server breakdown. This has been one of the most prevalent forms of DDoS attack. It has been noticed from the current trend that Otherminders now target most of the Sandbox analysis/anti-malware website servers for the DDoS attack. However, they use the P2P Botnet detection scheme. Fig 2 describes the IRC Botnet mechanism and their propagation.

![IRC Botnet Propagation](image)

**Fig. 2.** IRC Botnet propagation.

### A. P2P-based Botnets Description

Before analysis and the proof of the adaption, we first have a close look at the architecture of the P2P-based botnet.

In the propagation stage, the bots tries to spread as far as possible. Since the more machines it could compromise, the more damage it could cause. A common mechanism for autonomous spreading bot to propagate further is to exploit remote code execution vulnerabilities in network services. If the exploit succeed, the malicious code will create a copy of itself and send it to the exploited machine to execute. Thus a successful compromission is completed. Another common way to use the social engineering. The bot sends a lot of spam e-mails, or malicious links and pretend to be "polite" information which tries to attract you to access them.

In the installing stage of the bot, it will create configuration file on the infected system, which contains an encoded form of information about other neighboring peers and maybe even the attacker’s information. The peer is represented via a hash value and maybe the IP address/port tuple for the machine to connect to in the future. This is the information that the bot needs to join the P2P network. Note that this is mandatory work the bot should do, otherwise, it won’t know how to join the network. This is very important for us to apply the traditional bot detection method. Then, the P2P bot will try to synchronize the time of all the peers. This is done by utilizing the Network Time Protocol (NTP) in Storm Worm. And some other bots use a refreshing number to indicate the time period of the commands.

After registering the information it needs to join the P2P network, it will try to communicate to it neighbors to start working. Previously, the bot take use of the P2P distributed hash table (DHT) routing protocol. With the development of technology, the bots have evolved in the communication. They use the distributed shared hash table (DSHT) in stead of DHT currently. The new mechanism use the XOR to compute the distance from one peer to another. And will pick the peer that has the distance lower to a
threshold to be the neighbors that they will communicate.

Therefore, the message types enumerated below remain the same, only the encoding changed. All algorithms introduced in this report and the general methodology are not affected by this change in communication since the underlying weakness - the use of unauthenticated content-based publish/subscribe style communication - is still present.

B. Traditional Bot Detection and Tracking

As we want to extend the traditional IRC-based detection and tracking to the P2P-based botnets, we take a look at the method first.

The standard technique to track IRC-based botnets includes two steps. The first step could be regarded as bot analyzing. In this step, we try to get a copy of the bot and analyze it. The common way of doing this is by using the Honeypots, or some other specific analysis softwares. The analysis will give us the information of the channel for sure since we need to join the channel. Also the information maybe encoded so that we could not read it clearly. From the network gateway, at least we could know which IP address and port it is connecting to. This information will tell us the location of the IRC server.

Now we go into the second step. We re-connect to the botnet to act as the spy. On receiving commands the spy will have to render the response to the commander of their action. Thus, we could obtain the information of the attacker. We should note that, since the spy will not really attack the victim, thus the response won’t be correct. Therefore, we should try our best to avoid being recognized by the commander. This is a hard operation because we don’t know how the response information is formatted corresponding to different commands. However, if we could collect enough information before being identified, we will be able to locate the commander and thus take him down.

C. Tracking Extension

Briefly speaking, we adapt three steps of the traditional botnet tracking method in the extension.

In the first step, we try to obtain a copy of the bot binary. If the botnet propagates through an exploit mechanism, probably we will have to wait for some one to compromise us. This is quite annoying because we are not positive. However, there are a lot of “white” websites that provides the bot binaries. Simply download those binaries will help to achieve this. For the spamming propagation, we could either access the e-mail we get, or download the binary as in the first mechanism.

The second step is the infiltration of the botnet, which is similar to the first step of the traditional bot tracking method. The difference lies in that, in the IRC-based tracking, we have the information of the IRC channel by inspecting at the gateway. While in the P2P-based botnet, we are not able to see the commander or channel, since there is no parallel channel here. However, we will still be available to find the peers we want to connect to. (If the configuration file is not encoded, we could directly see the peers and even the commander during the communication of the botnet.)

The third step, we continue work in the botnet and tries to find the commander or at least do some mitigation. For the content-based publish/subscriber-style P2P networks, a simple way of mitigation is to create our own commands instead of the actual commands we received from our neighboring peers and spread the new “commands” out. Hope that some peers will receive our meaningless commands before they receive the actual commands. Since there is the time-related protocol in the system, the original commands will not be executed again.

V. NEPENTHES PLATFORM

Nepenthes is a platform to deploy the honeypot modules(vulnerability modules) and more importantly the vulnerabilities can be extended to other operating systems too. Using this Nepenthes platform, we can collect malware that is currently being spread out wildly using the vulnerabilities currently existing in the different services. One example for such a service is the MS Window’s IIS(Internet Information Server) vulnerabilities. No other services have got so many bots based upon like the ones exploiting the IIS loop holes. A vulnerability in MS IIS allows users to cause access violations on a Web Server of this configuration. This causes the Server to stop functioning, a form of Denial of Service and the services can resume only after rebooting. One such example of a vulnerability exploited recently is the SQLcode injection attacks on IIS Web Servers.

Below figure shows the Nepenthes platform with its different modules:

Nepenthes has a good modular design and is continuously updated based on the new vulnerabilities exploited and the new type of BOT scripts encountered. The Daemon acts as the Nepenthes core and handles most of the Network Interface
connections and controls all the actions of the other modules. There are several modules registered to Nepenthes as of now and they are:

- **Vulnerability Modules**: This module takes care of the different vulnerabilities existing in the network services and emulates them. Here it is seen that only the necessary parts of the services are only emulated so that the incoming autonomous spreading malware is lured to exploit our honeypot. Hence the main goal of this module is to trigger the exploitation attempt and receive the actual payload.

- **Shellcode Parsing Modules**: This module analyzes the payload received by one of the vulnerabilities by parsing the shell code received and extracting some information regarding the propagating malware. This involves the use of XOR decoding to extract the shell codes since the codes are encrypted using XOR encoder.

- **Fetch Modules**: This module uses the information extracted from Shellcode parsing module to download and get a copy of the malware from the remote website. The current existing compatible protocols are HTTP, TFTP, FTP, csend & creceive(IRC based submission method).

- **Submission Modules**: This module takes of reporting the malware extracted to a Sandbox(eg: Norman Sandbox) or AntiVirus vendor sites for further analysis. Other options are (i) Saving a copy into configurable location on the filesystem and capable of changing the ownership. (ii) A module that submits the file to another Nepenthes instance to enable a hierarchial structure of Nepenthes sensors. (iii) Storing it in a secure central Database.

- **Logging Modules**: Logs finally analysis information about the extracted malware, emulation process which helps in getting an overview of the patterns in the collected data. All malwares do not spread by downloading shell codes, but by simulating a shell for the attacker directly. Nepenthes offers shell emulation by emulating a basic Windows shell for the attacker. Several commands can be interpreted and batch file execution is supported. Such simulation is also proved to be sufficient to lure the self-propagating malware. Based on the collected information from the shell session, corresponding malware is downloaded. Common method to infect a host via a shell is to write commands for downloading and executing malware into a temporary batch file and then execute it. For this, we need a Virtual File System setup. Scalability is enhanced due to this since files are created only on demand (or Copy-on-Write): when there is an incoming attack trying to create a file, this file is created on demand and subsequently, the attack process can access and modify it. Every shell has its own virtual file system and hence no concurrent sessions interfere with each other. The attacking process is analysed after the attack process and is used to extract more information regarding how to download the malware from the internet.

A. Flexible Deployment

One of the key advantages with Nepenthes is the flexibility to different set ups. One of the option is the LAN set up of the local nepenthes sensor. Most of the information about the malware is stored in the local hard disk. The local sensor in a LAN collects information about suspicious traffic there and the local sensor stores the collected information in a local database and also forwards all information to another nepenthes sensor. whole of fig2. Next option is the Hierarchial set up where in a distributed structure with several levels is build and each level sends the collected information to the next higher level. In this way, load can be distributed among several sensor or information about different network ranges can be collected in a central and efficient way. Internet part of fig4. Third option, traffic can also be routed using a VPN tunnel from a LAN to remote Nepenthes sensor. Right side of fig4.

B. Zero Day Exploits

Zero day attacks are those which use unknown vulnerabilities or those vulnerabilities which has no patch updates yet. Nepenthes platform has the capability to capture even those type of attacks. It is done through the use of the 2 modules: portwatch and bridging. These modules can track network traffic at network ports and can help in the analysis of new exploits. When a new exploit triggers the nepenthes platform, it triggers the first steps of a vulnerability module. At some point, this new exploit will diverge from the emulation and this diversion can be detected and then we perform a switch to either a real honeypot or some kind of taint analysis system. This second system is an instance of the system nepenthes which is emulating vulnerabilities and shares the internal state with it.

VI. EXPERIMENTATION

Our experiment attempts were done on different directions to check the scope of testing from both operating systems and also different Network setup. Virtual LAN Network was set up through the VMWare Virtualization environment. Debian package linux(UBUNTU-7.10) was used as the guest OS on the Virtual machine and Windows Vista as the host environment. Nepenthes honeypot was supported on the Ubuntu machine.
to exploit the self propagating malware. We got the 2 bots, malware.exe and rootkit.exe, supposedly P2P bots from the internet(offensivecomputing.com) for testing. We encountered few issues setting up networking communication between the LAN Guest and host OS.

MS Windows can also be chosen to run on VMWare with Ubuntu running on Guest OS. We also tried with Cygwin shell(Linux shell simulation in Windows) and ran it on Windows environment.

VII. Future Work

This paper has discussed and proved though not verified with logs from experiments and shown the adaptibility of Botnet detection for P2P bot malware using Nepenthes honeypot project. First they can be more exhaustively tested for all IRC bots(which are under control now) if they are all under radar of being detected. Secondly, we don’t verify those P2P bots, working on the mechanism of transferring attacker’s commands among peers in a decentralized Peer-to-Peer network, can be detected with the current Nepenthes modules setup. Hence for this type of P2P BOT also to be diagnosed, there needs an additional module to be incoroporated to the Nepenthes platform.

P2P botnet detection through honeypot has slightly better edge since the commands are transferred among the peer networks. The commander will not need the response from the peers. Otherwise, it is showing itself in front of the peers, which violates the basic purpose of P2P-based botnets. Thus, we don’t need to worry about being identified as the spy in the network.

From our point of view, the adaptability of the extension relies in the similarity of the joining stage of the botnet. No matter if it is an IRC-based botnet or a P2P-based boenet, it should provide the information for the compromised machine to join the network. By using honeypot technology, we will be able to find the server or the peers we will communicate with, and thus tries to mitigate the effect of the bots or locate the commander correspondingly.

The difference is that, in the IRC-based botnet, we will be able to find the commander or at least the IRC server so that we could take down the whole botnet. While in the P2P situation, although it is possible that we find the commander, generally, we will not be able to find it. Thus, we could only try to mitigate under the current known pattern. If a new bot appears, we will need to study it first and act correspondingly afterwards. This is kind of a passive solution which seems not so satisfactory.

Another point we want to mention is that, we are thinking about the developing trend of the botnets. Since the weakness of the current botnet is the information they need to join the network. Will be there be a new architecture of botnets that does not need the information?

VIII. Conclusions

1,800 attacks were registered throughout United States last month, May 2008, almost 20% higher than last month’s. Hence Peer-to-Peer botnets intrusion malware is continuously increasingly growing type of spam in the Internet network. However, Honeypot project has been the most successful detection system for IRC based Botnet. The extension for P2P based botnet detection has a smooth transition provided with the additional module written into Nepenthes platform. The additional module corresponding to the over writing of the attackers commands with irrelevant message to corrupt the Command messages shared among the Peers in the P2P network. Hence the same Nepenthes platform takes care of adaptively detecting the P2P bots as well in addition to standard IRC botnet.

This paper provides the skeleton format for Infocom 2008, in Latex format.

Acknowledgments

The authors would like to thank everyone, especially professor Athina Markopoulou who helped and guided us in framing this paper.

References

[4] "Detecting Peer-to-Peer botnets" by Reinier Schoof & Ralph Koning