Proceedings of the 11th Colloquium for Information Systems Security Education
Boston University
Boston, MA June 4-7, 2007

Collecting and Analyzing Bots in a Systematic Honeynet-based Testbed Environment

Napoleon C. Paxton, Gail-Joon Ahn, Richard Kelly, Kevin Pearson, and Bei-Tseng Chu
University of North Carolina at Charlotte

Abstract* - Networks of compromised machines called botnets are one of the most threatening adversaries over the Internet due in large part to the difficulty of identifying botnet traffic patterns. We have witnessed that existing signature-based detection and protection methods are ineffective in dealing with new unknown bots. By slightly modifying the code of an existing bot, bot commanders can bypass most signature based mechanisms. We believe that by analyzing bot traffic for malicious patterns, it is possible to develop a taxonomy of bot characteristics and in turn use these characteristics to develop risks which will ultimately be used in the decision making process of allowing or blocking traffic. In this paper, we introduce our Honeynet-based Bot Analysis Architecture which is the first step towards our Risk-Aware Network-centric Malware Detection and Prevention Framework. We discuss our current architecture and how it could be realized towards identifying unknown bots and other malware. In addition, we discuss our results and lessons learned from this work.

Index Terms – Network Security, Botnet Analysis, Honeynet

I. Introduction

Botnets are one of the largest problems that computers on the Internet face. Commanders of these botnets have their own purposes for their army of compromised machines ranging from spam for hire to distributed denial of service (DDoS) and phishing attacks. Pertaining to corporate networks, the primary problem botnets present is their capability to perform DDoS attacks [1]. Enterprises that offer web services have a difficult time in distinguishing a DDoS attack from a spike in legitimate customers accessing their service sites [2]. Also, enterprises have to protect their corporate networks from becoming part of a botnet through malware propagation. Malware of different types have been around for the decade. However, there has recently been a surge in various ploys that seek user's credentials, credit card numbers or other sensitive information. Malware includes a broad range of techniques that snoop on a user's activity, deploy Trojan horses, exploit with key and mouse logging software, and finally allow an adversary to control compromised machines in use [3, 4, and 5]. In this paper, we describe our Honeynet-based Bot Analysis Architecture which includes collecting bots, running them on an offline simulated network, and installing them on an open analysis system to connect with their command and control center. We use the actual collected bots to connect to their command and control centers instead of simulated attack bots, sometimes called "drones". We use our analysis template to discover characteristics of each individual bot. This template has proved to be an invaluable learning tool for students to interact with malware in the wild. In the future such characteristics will be used to determine relevant risk values of specific network patterns for making signature-based detection more effective.

The rest of the paper is organized as follows. Section II discusses background information and related works. The Honeynet-based Bot Analysis Architecture is presented in Section III. In Section IV, we discuss our analysis method along with the results. Section V concludes this paper with a brief description of future work.

II. Background Technologies and Related Work

Honeynets have been used to learn as much about bots and the attacker sending bots as possible [6]. Even though this approach allows us to gather attackers' footprints, a systematic data analysis method is still needed. In the botnet, the command and control is where the attacker sends commands to the botnet. Currently most malicious bots use IRC to communicate with the command and control. IRC's built-in multicast capabilities make it easy for the commander to send orders to all the bots in the botnet without much effort [7]. A more destructive form of communication for bots is with the P2P protocol. These bots contain P2P clients and can communicate with one another without the use of a central command center. With this type of command and control the attacker can initiate commands by posing as a peer anywhere in the network. Other forms of command and control are also being used to a lesser degree, such as instant messaging and cellular phones. As researchers continue to find ways to protect against IRC based command and control structures, the number of botnets controlled by other

This work was supported, in part, by funds provided by National Science Foundation (NSF-DUE-0416042), Department of Defense (H98230-05-1-0119 and H98230-06-1-0128) and e-Business Technology Institute at UNC Charlotte. All correspondence should be addressed to Dr. Gail-Joon Ahn (gahn@uncc.edu).

protocols will continue to increase. As mentioned earlier, DDoS attacks are extremely difficult to detect. Most existing mechanisms have limitations to properly distinguish botnet traffic from legitimate traffic, generating a high false positive rate [8]. A high false positive rate may be its own denial of service, since legitimate traffic is blocked from accessing the network. Botnets continue to be a growing threat until a trustworthy mechanism is presented that effectively detects and blocks botnet attacks while allowing a very low false positive rate [9, 10].

Defending networks against botnet attacks is an emerging issue in network security and cyber crime research communities. To our knowledge, there are only a few works using risk as a deciding factor such as a newly released McAfee's Advanced Botnet Protection in Intrusion Prevention System [8]. This tool takes a similar approach of our framework in that it uses a proxy to accept or block traffic that appears to be botnet related. It does not use the risk value rigorously but mainly relies on a signature based approach. Our architecture is very similar to the approach as noted by Rajab, Zarfoss, Monrose, and Terzis [11]. Some key differences are that instead of creating "drones" to connect to a command and control, we "install" the actual bot on a honeypot to connect to its command and control. Our correlation system component is also a major difference in that we are keeping track of similarities in the bots and the sources that download the bots. Their approach is also more geared towards discovering the level of activity of botnets on the Internet without discovering characteristics for identifying similar unknown variants of each bot and corresponding botnet traffic. Dagon, Gu, Zou, Grizzard, Dwivedi, Lee, and R. Lipton introduced a taxonomy of botnets to provide a response to botnets by degrading or disrupting them [12]. This method involved discovery and proactive attack to the botnet. In this paper, we focus on a bot taxonomy to build up properties for our risk-aware mechanism. Some earlier works addressed issues on tracking botnets [13]. Such works adopted sensors and honeypots to investigate a pathway to and from botnets. Our approach uses a virtual space such as honeypots to capture bots and track botnets. In addition, we attempt to move one step forward by providing a way to categorize the bots and to record scanning activities targeted for vulnerable services. This allows us to grasp more details of the intent of the adversary and gives us a way to keep track of what services are being attacked the most.

III. Honeynet-based Bot Analysis Architecture

[1]Our honeynet testbed was created to satisfy three major requirements:

- Systematically collect and analyze malware traffic over the Internet
- Comprehensively discover characteristics and unique behaviors of malware
- Dynamically determine associated risks and generate corresponding detection rules.

In this section we discuss the components of the architecture without having any specific tools in mind. Any tool that can perform the tasks described here can be used as part of the architecture. This requirement is to ensure the extensibility of our architecture. Figure 1 is a visual representation of our architecture.

Malware Collection and Network Monitoring: Before we can discover what the risks are in a network, we need to discover how attack code reacts with the system. To realize this goal, a collection system is proposed that collects bots to be dynamically analyzed. Dynamic analysis occurs by ensuring that the collection system emulates each of the services on the network it is protecting. We capture bots by emulating the vulnerable services. Also, this system provides protection against significant involvement in attacks after the bot has been run on the system. It uses firewall and intrusion protection techniques, such as limiting or dropping packets leaving the protected network.

Closed & Open Analysis System Component: This component takes the binary captured in the collection system and runs it on a closed network environment. This is a necessary step to discover certain aspects of the malware before putting it on the open analysis system and opening it up to the network. The closed analysis component has the capability to use attack commands found in the binary and perform simulated attacks using a Perl script. These attacks are only run in the simulated network and will give insight to what the binary is made to be used for. It includes the discovered hard-coded DNS addresses, attack commands, and other functionality of the bot. Eventually more functionality will be identified from the closed analysis such as patterns from the virtually simulated attacks that can be performed within the closed analysis system. The open analysis component of this system allows us to inject a malicious bot into a computer and connect back to its original destination. This enable us to isolate the bot from the network and monitor its traffic in a more controlled way instead of waiting to be infected and then monitoring the

[1] To date we have satisfied the first major requirement and are well on our way to satisfying the other two. More information is given on steps to satisfy the remaining two requirements in section 5.

traffic passively. The strings are pulled from the binary as it is being run in memory, thereby negating any obfuscation techniques.

<u>*Pattern Correlation System Component*</u>: The pattern correlation system takes input from the open analysis and closed analysis systems and creates an intelligence report to display the alert events that are identified from the bot installed. This intelligence report is used to discover patterns in the traffic and correlations between logs. The goal of the correlation system is to gather as much information (characteristics) about each individual bot as possible and correlate the results with other bots to discover a taxonomy of each bot Each bot taxonomy will have a list of its own characteristics as well as references to other bots that use or have any connection with the bot entry in the taxonomy. This information is referenced by the risk-aware engine. The purpose of the taxonomy is to provide a comprehensive identity for the bot so the characteristics provided by the identity will lead to an accurate assessment of the risks they present. A taxonomy updater is also needed to keep the taxonomy up to date and accurate. When a new correlation is found in a bot, its taxonomy entry will change to reflect the new correlation. All other bot taxonomies that are cross-referenced by that bot will then be updated by the taxonomy updater. The repository is a central collection of all the logs in our architecture. This gives the administrator a macro view of the protection system and provides an aggregated view of the attackers on the network. The repository holds statistics and geographical information on the logs and presents them as input to the risk-aware engine to be used as a factor in the assignment of risk to the traffic.

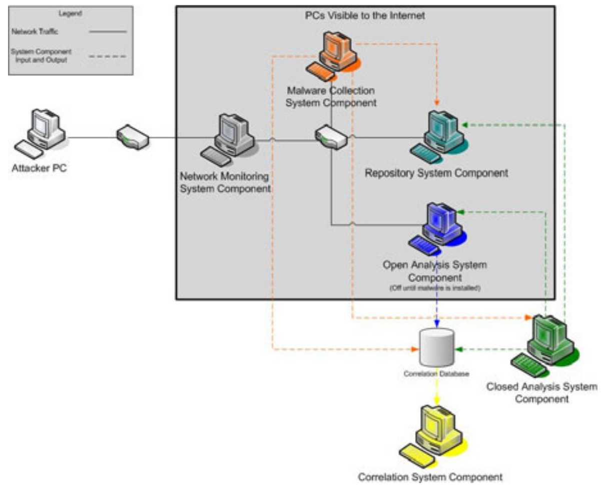

Legend
Network Traffic
System Component Input and Output
PCs Visible to the Internet
Malware Collection System Component
Attacker PC
Network Monitoring System Component
Repository System Component
Open Analysis System Component
(Off until malware is installed)
Correlation Database
Closed Analysis System Component
Correlation System Component

Figure 1: Honeynet-based Bot Analysis Architecture

IV. Analysis Method

In this section, we describe how we have analyzed the bots and what tools were used in our components.

A. Tools Used in Analysis

- Malware Collection: Component for capturing and storing binaries. We used two tools to accomplish this.
    - Nepenthes – A low interaction honeypot for capturing malware [14], [15].
    - MySQL – Our database of choice for storing the malware [16].

- Closed Analysis: Component for analyzing each captured binary offline before allowing it to be run in its native environment. To implement this component we utilize one tool.
    - Sandnet – Sandnet emulates the Internet and gives us the ability to act as the command and control by sending commands found in the strings to a python script that allows us to issue the bot commands [17].

- Open Analysis: Component for analyzing each binary in its native environment. We currently use seven tools to perform this analysis.
    - VMWare – This tool gives us the ability to run our bots on an operating system image that can be quickly restored to the previous system state. This allows us to quickly switch from bots to bots in our analysis process [18].
    - Perileyez – A malware analysis tool that compares snapshots of the system and produces all the changes made. We run this tool before we place the bot on the honeypot and to observe any immediate changes it makes [19].
    - Sebek – A root kit used to collect all the system calls from a client and server. We use this root kit to record all the commands given from the bot master to the bot [20].
    - Wireshark – This tool analyzes network packets. We use this as a learning tool to manually analyze packets [21].
    - Honeywall – It monitors all packets in and out of our architecture. It also provides us with data control, which is our fail-safe shutdown method to avoid being an active participant in a botnet attack [6].
    - Maxmind Database – Tool for displaying the location of an IP on a world map. We use this tool to map the source locations of where the malware was downloaded from [22].

- Norton AntiVirus and ClamAV – We use this tool to determine whether the antivirus signature and categorization for each bot exist [23, 24].

B. Malware Interaction

This section discusses the steps of how the malware interacts with our components including each tool's role in our architecture.

Malware collection is achieved using Nepenthes, a program that emulates Microsoft Windows services to incite automated attacks[1]. When an attack occurs, Nepenthes logs the malicious activities and attempts to download any binaries associated with the attack. The downloaded malware is automatically stored in a MySQL database on the architecture, as well as the originating IP address, and run through two anti-virus engines, *Norton 10 Corporate and ClamAV*. The anti-virus engine results are then stored in the database. Our *Maxmind Database* detects the source of the bot and adds an entry on a map of the world to geographically visualize the location of the bot.

For our *Closed Analysis* we use a simulated environment. Although, Norman Sandbox is the most popular malware simulation environment, we use a tool called Sandnet. Sandnet provides an isolated environment and a virtual network for the piece of malware to execute. The environment consists of two computers, a Sandnet Server and Sandnet Client. After the initial execution of malware, an md5sum file, memory dump file and network traffic logs are sent to the Sandnet Server from the Sandnet Client. Using a specifically designed Perl script we can recompile the memory dump file and running the Linux command (strings –a <*file*>) to obtain the strings off of the malware. The strings allow us to determine commands used by the malware as well as target areas that the malware will be likely to hit. Furthermore, Sandnet is able to simulate various types of servers, the most important being an IRC server since this is the most notorious avenue for sending malware commands.

Our *Open Analysis* provides connection to the internet. The live execution environment is notably more verbose than using Sandnet. To begin, VMWare workstation is used to create a default installation of Windows XP, Service Pack 1. After the image is created, Sebek is installed onto the image. Sebek is a kernel based data capturing tool and captures the processes used by the image, sending them as packets across the network.

To obtain the files added, deleted and changed by the malware the tool Perileyez is run on the image. The initial snapshot of the image is taken once Sebek and Perileyez are installed and after the malware is executed, a second snapshot is taken. By comparing the two snapshots we can identify alterations the malware makes to the image including changes to drivers, DLLs, processes, ports and remote connections as well as any files changed.

Capturing and analyzing network traffic is the final step in running a live execution environment. To capture all network traffic generated by the virtual environment, we use a Honeywall. The Honeywall is able to capture all network packets that are sent and received by the image. These packets are merged into PCAP files and sent to a central server at the end of each day. Currently we have found it useful to separate the PCAP files into four hour segments, giving us six slices for each day. By segmenting the file, it allows us to locate suspicious data more easily. Using the tool Wireshark, we can look at the daily PCAP files and determine the actions of the malware for the previous day. One PCAP file can display IRC conversations, secondary injections attempts, DNS queries, propagation scans and HTTP conversations as well as any other type of network traffic.

C. Analysis Methods and Results

We analyze our collected malware using a predefined method. The list below shows the content of each analysis. Each week information security students share their findings on bot characteristics using this template with other students and faculty. This has greatly increased their competency in analyzing these bots within their native environment.

- Identification: MD5 value and anti-virus engine results
- Source of Infection: network traffic analysis related to the location where the malware downloaded from.
- System Interaction: system state report which includes files added, unloaded drivers, unloaded dlls and so on.
- DNS queries: identification of domain names for command and control servers and corresponding ISP information
- IRC Communications: collection of live IRC conversation and any traffic related to scanning and secondary injection

Most of the malware that we have examined have exhibited similar behavior. Figure 2 is a snapshot from our bot repository. It shows our number of total binaries as opposed to the number of binaries that were actually detected. As we notice, both Norton Antivirus and ClamAV did not detect about 25% of the bots that were downloaded in Nepenthes. When started, at least one and

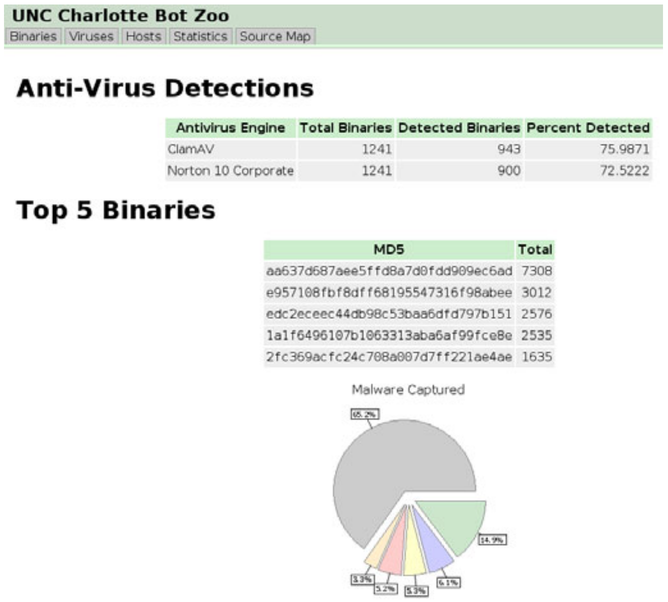
[1] We have currently captured Windows based malware.

as high as fourteen executable were installed on the image. Ports were opened, processes shutdown and/or restarted and new registry keys created. The malware usually restarts legitimate Windows processes so that it may append itself to that process. For example, msmgs.exe is the MSN Messenger process and, by default, is loaded on startup causing the malware to be reloaded every time the machine is restarted. In a high number of instances, the malware "hardens" the system to prevent other bots from infecting the machine with any further attacks and leaves the system still accessible, so that the casual user would not notice much difference. Only a small number of times has a piece of malware completely disabled the image causing it to be unusable.

To use a concrete example, the malware Trojan.Mybot-7663 initially loaded the files lssas.exe and fswinsys.exe, which are registered as the W32.AGOBOT.RL Trojan and Worm.Ircbot.Gen respectively. It furthered its assault by unloading 90 drivers from memory including cdrom.sys, ultimately rendering the CDROM useless. It proceeded to unload 250 DLL files and deleted 77 services, most notably the secondary logon service causing major problems in logging into the image. After opening a few select ports, the malware terminated 16 processes, many system critical. These processes included lsass.exe, winlogon.exe, and services.exe and even though all were eventually restarted it is safe to assume that they were tampered with.

All of the malware that we have actively examined use some type of systematic scan, presumably for propagation. Most of these were TCP SYN scans on a class B subnet. If a TCP SYN scan was not used, ICMP ping scans were used. We have noticed that DNS queries were hard coded into the bots, using the returned IP address to log into an IRC server and obtain secondary injections. Some malware ran had been relatively inactive until the completion of the secondary download in which a propagation scan would ensue. A high number of malware have displayed this behavior allowing us to form the hypothesis that malware writers use other writer's code to ensure a small, compact binary. For example, our Nepenthes sensor captured a process called *fswinsys.exe* for the first time on May 10, 2006 and since have seen numerous hits per day. Upon execution, we realized that *fswinsys.exe* is able to initiate a propagation scan a lot more quickly than most other malware. After this realization we ran numerous other malware that would download the *fswinsys.exe* process as a secondary injection and used for propagation scans. This discovery lead us to our second hypothesis, of which many of the malware writers use previously created malware or copy and paste code from previously created malware. For example, the malware following the md5sum 429d74b465003ddcfd54b586705191cb (classified as a W32.Spybot.Worm) displayed the above mentioned behavior. Its initial execution resulted in PCAP slices ranging from 200K to 600K. Once the secondary injection of *fswinsys.exe* was complete the next slice was 7.8M. The propagation scan had a time limit associated with it so on completion the PCAP slices fell back to its 200K to 600K average. The malware then received a second propagation scan command the following day, but with no time limit and a longer delay resulting in PCAP slices ranging from 1.5M to 6.9M. This malware has become common among our analysis team in which the *fswinsys.exe* process is used to initiate large propagation scans.

Malware use IRC channels to receive commands for propagation scans and secondary download. Throughout the life of our Honeynet, these bots have shown an interesting similarity in the type of commands received. A main focal point for all malware is the use of a propagation scan. The common command for a propagation scan has been .advscan <port#> <threads> <delay> <time> <switches>. For example, the command .advscan lsass_445 200 5 0 –r –b –s would correspond to a randomized (-r switch), class B (-b switch) subnet scan on port 445 using 200 threads with a 5 second delay for an infinite amount of time. Rarely does a piece of malware designate a time for the scan to finish so the 0 is used to express an infinite amount of time. Furthermore, the –s switch is a silent switch that bots will use to keep their status from being broadcast across the IRC channel.



Anti-Virus Detections

Antivirus Engine	Total Binaries	Detected Binaries	Percent Detected
ClamAV	1241	943	75.9871
Norton 10 Corporate	1241	900	72.5222

Top 5 Binaries

MD5	Total
aa637d687aee5ffd8a7d0fdd909ec6ad	7308
e957108fbf8dff68195547316f98abee	3012
edc2eceec44db98c53baa6dfd797b151	2576
1a1f6496107b1063313aba6af99fce8e	2535
2fc369acfc24c708a007d7ff221ae4ae	1635

Figure 2: Bot Repository

V. CONCLUSION

In this paper, we have discussed our Honeynet-based Bot Analysis Architecture. We have shown that our testbed has been an invaluable learning tool for our students and allows them to directly observe interactions of malware in the wild. Our approach has provided us with the analysis results necessary to move forward to our next steps which are to determine the characteristics of incoming malware. For the future work, our goal is to develop a risk aware framework that will examine incoming network packets and use malware characteristics to determine whether network packets are suspicious or not.

VI. Acknowledgements

We would like to thank the anonymous reviewers for their comments and suggestions.

VII. References

[1] Phishing Reaches an All-time High in March. Available at www.it-observer.com

[2] S. Kandula, D. Katabi, M. Jacob, and A. Berger, “Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds,” In Proceedings of NSDI, 2005

[3] R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. Symp. On Usable Privacy and Security, 2005.

[4] S. Saroiu, S. D. Gribble, and H. M. Levy. Measurement and Analysis of Spyware in a University Environment. Proc. NSDI, 2004.

[5] E. Skoudis and L. Zeltser. Malware: Fighting Malicious Code. Prentice Hall, 2004.

[6] The Honeynet Project & Research Alliance. Know Your Enemy: GenII Honeynets. Available at www.honeynet.org/papers/

[7] J. Li, T. Ehrenkranz, and G. Kuenning, “Simulation and Analysis on the Resiliency and Efficiency of Malnets,” In the Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation, 2005

[8] J. Dunn. McAfee launches first bot-killing system. Available at www.techworld.com

[9] The Honeynet Project & Research Alliance, "Know Your Enemy: Tracking Botnets,” Available at www.honeynet.org/papers/

[10] T. Holz, “A Short Visit to the Bot Zoo,” Security & Privacy Magazine, IEEE 2005

[11] Moheed Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis, “A Multifaceted Approach to Understanding the Botnet Phenomenon,” In the Proceedings of ACM IMC, 06

[12] D. Dagon, G. Gu, C. Zou, J. Grizzard, S. Dwivedi, W. Lee, and R. Lipton,”A Taxonomy of Botnets,” Available at www.math.tulane.edu/~tcsem/

[13] F. Freiling, T. Holz, and G. Wicherski, “Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks,” In the Proceedings of the 10th European Symposium on Research in Computer Security, 2005

[14] Baecher et al, "The Nepenthes Platform: An Efficient Approach to Collect Malware," In Proceedings of RAID '06.

[15] Nepenthes-Finest Collection, Available at nepenthes.mwcollect.org/

[16] MySQL, Available at www.mysql.com

[17] Truman – The Reusable Unknown Malware Analysis Net, Available at www.lurhq.com/truman/

[18] Vmware GSX Server, Available at www.vmware.com

[19] Perileyez, Available at www.digitalninjitsu.com/downloads.html

[20] The Honeynet Project, Know Your Enemy: Sebek--A kernel based data capture tool, November 2003, Available at www.honeynet.org

[21] Wireshark, Available at www.wireshark.org

[22] Maxmind, Avalible at www.maxmind.com

[23] Norton 10 Antivirus, Available at www.symantec.com

[24] ClamAV, Available at www.clamav.net