DISTRIBUTED LOW-INTERACTION HONEYPOT SYSTEM TO DETECT BOTNETS

AHMAD JAKALAN [1] **GONG JIAN** [2] **LIU SHANGDONG** [3]
ahmad@njnet.edu.cn jgong@njnet.edu.cn sdliu@njnet.edu.cn
Jiangsu Key Laboratory of Computer Networking Technology,
China, Nanjing, Southeast University

Abstract:

The objective of this research is to detect the existence of botnets in the monitored network by designing a distributed low-interaction honeypot, and to provide clues from the detection for the threat evaluation by botnets propagation estimation. A distributed framework of nepenthes honeypots is built to collect as more as possible malware samples. These samples are analyzed firstly by features via antivirus scan, then by behavior via two online sandboxes. The configuration of Nepenthes is optimized to improve the capture efficiency.

Keywords: Network security, Botnet detection, Honeypot, Nepenthes.

1. Introduction

Malware is a term that means software or a piece of software that serve malicious purposes. Malware is often used to infect the computers of unsuspecting victims by exploiting software vulnerabilities or tricking users into running malicious code. There are different classifications of malware depending on its propagation method, activity, goals. As most of security researchers classify it as the most "evil-minded" botnet are now the greatest challenge in for researchers. The term *botnet* is used to define networks of infected end-hosts, called *bots* that are under the control of a human operator commonly known as a *botmaster* or *bot-herder*. Botnets are not only a constant threat to the integrity of individual computers on the Internet; the combined power of many compromised machines is a constant danger even to uninfected sites. Botmasters use bots for a variety of attacks. For example carrying out Distributed Denial-of-Service (DDoS) attacks, sending out millions of spam or phishing e-mails, Attacks against infected hosts often hurt their performance and may include capturing private information or credentials for identity theft. It has gone the time that hackers try to demonstrate their technical prominence among others, instead, botnets are predominantly used for illegal activities. Every compromised machine a so called *bot* establishes a connection to a remote control network by which the attacker can issue arbitrary commands. Typical examples for these remote control networks are IRC networks, HTTP servers, and P2P. Malware detection has become difficult with the use of compression, polymorphic methods and techniques to detect and disable security software. Those and other obfuscation techniques pose a problem for detection and classification schemes that analyze malware behavior.

[1] Ahmad Jakalan, Southeast University, Nanjing, China. Tel: 008615366165651. Fax: 00862583694035. Email: ahmad@njnet.edu.cn, jakalan982@hotmail.com.

[2] Gong Jian, Chief of Jiangsu Key Laboratory of Computer Networking Technology, Southeast University. Email: jgong@njnet.edu.cn.

[3] Liu ShangDong, Southeast University, Nanjing, China. Email: sdliu@njnet.edu.cn

2. Types of honeypots

The first step to study malware and its malicious activities is to collect malware samples. The main tool recommended to collect malware in an automated fashion today is so-called *honeypots.* A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource; they are the vulnerable systems waiting for attacks. The idea behind this methodology is to lure in attackers such as automated malware and then study them in detail. Honeypots have proven to be very effective tools in learning more about Internet crime like botnets. There are two general types of honeypots:

1. *Low-interaction honeypots:* They emulate services or operating systems with a low level of interaction. Implementing this type of honeypots tends to be low risk, main intention is to *capture harmful code samples*, Deployment and maintenance tends to be easy. A popular example of this kind of honeypots is *nepenthes.*
2. *High-interaction honeypots* offer the attacker a real system to interact with. The risk of deploying tends to be higher, so it's required to establish precautions and special provisions are to be done to prevent attacks against system. More complex to setup and maintain. The main intention is to understand the attack scene, concerned that the attacks on the process, it requires a strong ability to interact with the attacker. The most common setup for this kind of honeypots is a *GenII honeynet.*

Nepenthes, is a low-interaction honeypot like honeyd or mwcollect. It is designed to run on Linux and it emulates known vulnerabilities in the Windows OS that worms use to propagate. The worm payload used to infect Windows machines are downloaded and stored as binary files for later analysis. It is a scalable honeypot, this is because of its ability to be configured to listen to a numerous number of IP addresses. Nepenthes is useful to capture new malware samples spreading by exploiting old vulnerabilities but still useless for capturing samples of malwares that exploit new vulnerabilities, that is simply because these vulnerabilities are not emulated yet, but at the same time it has the ability to include more vulnerabilities modules.

3. Infrastructure

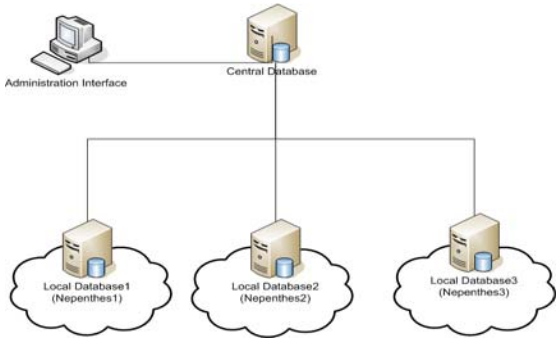

Figure 1: distributed low-interaction honeypot

The deployed system is designed as shown in figure 1; it consists of at least one Linux server machine with nepenthes installed on it. We have installed two machines with nepenthes-0.2.2. The two machines form a distributed farm of low-interaction honeypots each one collects malware samples and submit it to the malware database server. Configurations on the nepenthes servers include adding a range of IP addresses (complete scope C of IP addresses for each honeypot) to increase the probability of catching the spreading malware. Each machine (nepenthes honeypot nodes) has two NIC (Network Interface Card) one is allocated to the local login dedicated for the management, and the other is connected *directly* to the internet without any filtering on the gateway to enable the honeypot to receive as more as possible network attacks, It is configured to use a complete list of class C network IP addresses each one referred to as nepenthes honeypot sensor.

4. Malware collection

Nepenthes is set up to listen to a number of ports which the vulnerability modules expect to receive a worm attack through. It is a passive honeypot, means that it will not invite worms to hack the machine; instead it should wait until one of its vulnerable open ports is scanned by the worms, then the source of infection will send the shell_code which will trigger the machine to download the malware code, at this time the honeypot will log a download attempt of new malware. On the accomplishment of malware download, the honeypot Stores the sample in binaries named with its md5-hash and logs a successful download, then the file with some of useful information are submitted to the malware database server via http submission. The central machine runs the administration website which provides ability to submit the collected malwares to online sandbox for analysis and to receive the analysis reports from them, extract network activity from the analysis report to understand the network behavior of the malware, in addition to the ability to scan the malwares with some known antivirus. The http submission from distributed honeypots to the main server is implemented as a PHP code to be requested by other honeypot nodes by configuring the nepenthes honeypot to request this http code as soon as it has a new malware, the new malware binary code is submitted to the malware database server in addition to the source of infection IP address, honeypot sensor IP address, md5-hash. One malware could be collected many times from the source of infection, or from other sources, so each time the information are logged but only one time the file is submitted to the malware server. For a period of about one year we have collected more that (2500) different malware samples, we had on one of the honeypots (the main honeypot) which is always run more than (215k) download attempts, and more than (21k) successful downloads. But the other honeypot was run for two periods each one of about ten days. In the first run from 2010/11/04 to 2010/11/14 it has collected two different malware samples which are already collected by the first nepenthes honeypot. In the second run from 2010/12/05 to 2010/12/15 it has collected two *new* different malware samples but this time both the new collected malware samples haven't been collected by the first nepenthes honeypot.

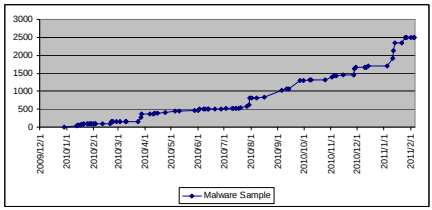

Figure 2: The total number of the collected malware binaries

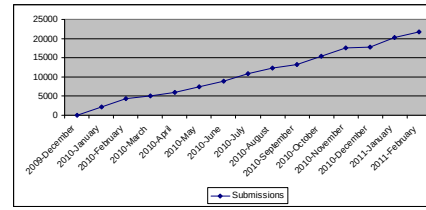

Figure 3: The total number of the successful download of malware binaries

Figures 2 and 3 show comparison between the total number of the collected malware samples and the number of successfully downloaded malware binaries with a percentage of about 1/10, this is because many malware files were submitted many times on different sensors of the honeypot.

5. Malware analysis

Antivirus scans provide a little information about the scanned malware samples; even it may give false results. Antivirus scan depends on Virus Signature. A signature is an algorithm or hash (a number derived from a string of text) that uniquely identifies a specific virus. Most antivirus software are not able to detect zero day spreading malwares, they need to be added to the signatures database before they can be detected, so it's not enough to depend only on antivirus and it's important for security researchers to analyze the new collected malware.

Analyzing unknown executables is divided into two broad categories: *static analysis* and *dynamic analysis*. In static analysis the program's binary code is disassembled first, then, both control flow and data flow analysis techniques can be employed to draw understand the functionality of the program. Dynamic analysis is the process of observing the code during run-time to determine the purpose and functionality of the malware sample. This manner has an advantage that the code is actually executed. Thus, dynamic analysis is immune to **obfuscation** attempts and has no problems with self-modifying programs. But still there is a problem in building the suitable environment in which the binary executable file can be executed safely without affecting the other parts of the network. Running malware directly on a real machine which is part of network or connected to the internet could be disastrous as the malicious code could easily escape and infect other machines. In addition, reinstalling the operating system on the machine after each dynamic test run is not an efficient solution because of the overhead that is involved. To solve these problems, sandbox techniques are used, sandbox is a secured environment which emulates real world environment to enable researchers to execute and observe malicious code securely. Having private sandbox is very useful but building sandbox tends to be very complicated, still there is the ability to depend on many available third party online sandboxes, we used two of them to get reports from different sources for the accurate analysis of malware samples, the first one is *Anubis*, and the other is *Joebox*. Both of them enable submitting the malware sample and they return back the analysis report. Normally analysis reports are divided into many categories like: General Information about the malware sample like file size and time to perform the analysis, File activities, Registry

activities, Services activities, Process activities, in addition to the network activity part which is the most important for the network security researchers. As we have mentioned, our focus is the detection of the existence of botnet, the main characteristics of bots are the networks behavior, so we pay attention on the network behavior section of the report. All the network behavior of the analysis report is collected with a multiple submissions of each malware sample in different times to multiple sandboxes. This will provide more accuracy to our results. Because each malware is executed in the sandbox alone to observe changes on the operating system and the network activity, and the malware can't be executed for a long time, in addition to that botnets are not always active, they are just waiting for the command of the botnet controller, so It's useful to execute the malware sample many times and collect the network activities of the analysis reports of the multiple executions. Most of the existing botnet controllers use IRC to communicate with their zombies. The following table shows a brief description of some randomly selected malwares that show botnet behavior.

Table 1: A brief description of some randomly selected malwares that show botnet behavior

Malware md5	Source IP	Kaspersky	Behavior analysis			
			DNS?	DNS Answer	HTTP	IRC server
b313* 6c13	*.198.84.204	Backdoor.Win 32.Rbot.sr	dns.aswend.com	*.107.249.167		*.107.249.167
7f60* 8290	*.83.108.240 *.163.62.37	Net-Worm.Win32. Padobot.n	moscow-advokat.ru			
			coins.dal.net	*.14.236.50		*.14.236.50
			diemen.nl.eu.undernet .org	*.109.20.90		*.109.20.90
1f8a* 173b	37 different sources	Backdoor.Win 32.Rbot.aftu	botz.noretards.com	*.111.73.201		
1085* 60d0	*.160.112.7 *.240.41.10	Trojan.Win32 .Buzus.cvzu	ss.ka3ek.com	*.196.130.50		*.196.130.50
			ss.nadnadzzz.info			*.196.130.50
			ss.MEMEHEHZ.INFO			*.196.130.50
			ss.memehehz.info			*.196.130.50
			go.microsoft.com	*.55.57.251	*.55.57.251	*.55.57.251
			www.ieaddons.com	*.136.35.139	*.136.35.139	*.136.35.139
			www.microsoft.com	*.55.12.249	*.55.12.249	*.55.12.249
			worker-24.seclab.tuwien.ac.at	*.130.56.24		
			ss.nadnadzzz.info	*.43.232.36		*.43.232.36
c4da* 93e2	*.145.120.165	Trojan.Win32 .VB.aizl	http communication without DNS		*.55.57.251	
0368* 6508	*.44.197.72	Trojan.Win32 .Buzus.cvzu	ss.ka3ek.com	*.196.130.50		*.196.130.50
			ss.nadnadzzz.info			
			worker-24.seclab.tuwien.ac.at	*.130.56.24		
2aa6* 764c	*.44.197.72	Virus.Win32. Virut.n	proxim.ircgalaxy.pl	*.133.119.206		
			ss.MEMEHEHZ.INFO	*.196.130.50		
			ss.nadnadzzz.info			
			ku.perfectexe.com	*.170.127.203		
			image.perfectexe.com	*.170.127.203	*.170.127.203	
			kdddaber.com	*.217.162.178	*.217.162.178	
					*.190.222.131	

6. Conclusion

In this paper we have presented our work in detecting the existence of botnet by implementing a distributed low-interaction honeypot. This work show the importance of collecting and analyzing malware and how the behavior analysis shows information can't be obtained by only scanning the malware files by antivirus. Actually detecting botnets depends on suspicion in the malware which is showing activities similar to botnet activities. The information obtained from the proposed system provides clues to understanding the botnets and in divesting the controllers of these botnets. Data stored in database will be also helpful to study the propagation methods and targets of the analyzed malwares.

7. Acknowledgment

This project was supported in part by: State Scientific and Technological Support Plan Project of China under Grant No. 2008BAH37B04, National Basic Research Program of China (973) under Grant No. 2009CB320505, and National Natural Science Foundation of China under Grant No. 60973123.

8. References:

[1] Jos´e Brustoloni, Nicholas Farnan, Ricardo Villamar´ın-Salom´on and David Kyle, Efficient Detection of Bots in Subscribers' Computers, 2009 IEEE 978-1-4244-3435-0/09

[2] B. M. H¨ammerli and R. Sommer (Eds.): DIMVA 2007, LNCS 4579, pp. 109–128, 2007. Measurement and Analysis of Autonomous Spreading Malware in a University Environment.

[3] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, "The Nepenthes Platform: An Efficient Approach to Collect Malware," 9th International Symposium on Recent Advances in Intrusion Detection, RAID06, Hamburg, Germany, September 20-22, 2006, Proceedings, Lecture Notes in Computer Science (Springer, 2006).

[4] André R. A. Grégio1, Isabela L. Oliveira2, Rafael D. C. Santos3, Adriano M. Cansian2, Paulo L. de Geus1. Malware distributed collection and pre-classification system using honeypot technology

[5] Michael Bailey1, Jon Oberheide1, Jon Andersen1, Z. Morley Mao1, Farnam Jahanian1,2, and Jose Nazario2. Automated Classification and Analysis of Internet Malware.

[6] Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: A multifaceted approach to understanding the botnet phenomenon. In: IMC '06: Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, pp. 41–52 (2006)

[7] Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging. In: Proceedings of 1st Workshop on Hot Topics in Understanding Botnets (HotBots '07) (2007)

[8] Saroiu, S., Gribble, S.D., Levy, H.M.: Measurement and analysis of spyware in a university environment. In: Proceedings of Networked Systems Design and Implementation (NSDI'04), San Francisco, California, United States (2004)

[9] Baecher, P., Koetter, M., Holz, T., Dornseif, M., Freiling, F.C.: The nepenthes platform: An efficient approach to collect malware. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 165–184. Springer, Heidelberg (2006)

[10] Nepenthes – Finest Collection. Available at: http://nepenthes.carnivore.it.

[11] http://anubis.iseclab.org/

[12] http://www.joebox.org/

[13] http://www.kaspersky.com/