DISTRIBUTED LOW-INTERACTION HONEYPOT SYSTEM TO DETECT BOTNETS

AHMAD JAKALAN[1], GONG JIAN[2]
{ahmad, jgong} @njnet.edu.cn
Jiangsu Key Laboratory of Computer Networking Technology,
Southeast University, Nanjing, China

Abstract: The objective of this research is to design and develop a distributed low-interaction honeypot to detect the existence of botnets in the monitored network, and to provide clues for the threat evaluation by botnets propagation estimation. Running a low-interaction honeypot provides a good feature that there is a big evidence of the existence of botnet. This evidence is the malware (botnet) sample collected by the honeypot. The approach followed in this research for detecting botnets is to analyze all the collected malware samples to know their types, and to detect if any botnets exist between all the collected malware samples. So the first target is collecting as more as possible malware samples, and then analyzing them to spot the botnets samples. The accurate results we get depends on not only one source of analysis, it depends on the multiple sources especially the behavior analysis of the collected malware samples.

Keywords: botnet detection, honeypot, network security.

1. Introduction

Recently broadband Internet connections became very common even for home users, normally these users have little or no information about internet security, and so they are the most desired target for most internet attacks by different types of malware. *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. Malware can exploit and compromise a system in many ways, either attacking operating systems vulnerabilities or remote services through Internet or deceiving the user to execute it by clicking on a fake link or opening an e-mail attachment. 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*. Such malware is not only a constant threat to the integrity of individual computers on the Internet; for example they can bring down almost any server through distributed denial of service, 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

---

[1] Ahmad Jakalan, Master's Degree student in Computer Networks Security, 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, Professor in Computer Networks in Southeast University. Email: jgong@njnet.edu.cn.

identity theft. It has gone the time that hackers try to demonstrate their technical prominence among others, instead of, 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. The objective of this paper is to present our work in ***designing and developing a distributed low-interaction honeypot to detect the existence of botnets in the monitored network, and to provide clues for the threat evaluation by botnets propagation estimation.***

2. Types of honeypots

The first step to study malware and its malicious activities is to collect malware samples. Security researcher have invented and deployed several ways and technologies to collect malware, either by setting up a vulnerable system to wait for attacks, or crawling web pages for malicious code stored on servers. Indeed, the effective method recommended to collect malware samples depend firstly on the spreading model of the malware itself, *Automated Malware* particularly means malware that spreads automatically over the network from machine to machine by exploiting known or unknown vulnerabilities. The main tool recommended for this type 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 a very effective tool in learning more about Internet crime like botnets. There are two general types of honeypots:

1. *Low-interaction honeypots* this type offers limited services to the attacker. They emulate services or operating systems with a low level of interaction which varies with the implementation. The risk of implementing this type of honeypots tends to be very low, because of that its main intention is to *capture harmful code samples,* so usually do not require too much interaction. Deploying and maintaining low-interaction honeypots tends to be easy. A popular example of this kind of honeypots is *nepenthes.* With the help of low-interaction honeypots, it is possible to learn more about attack patterns and attacker behavior.
2. *High-interaction honeypots* offer the attacker a real system to interact with. The risk of deploying this type of honeypots tends to be higher than that of low-interaction honeypots, so it’s required to establish precautions and special provisions are to be done to prevent attacks against system. They are normally more complex to setup and maintain. The high-interaction honeypots 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. Meaning it is not a fully blown Operating System with live running services. Instead Nepenthes is designed to run on Linux and it emulates known vulnerabilities in the Windows OS that worms use to propagate. The emulated vulnerabilities cannot be used to attack the underlying Linux OS, so nepenthes requires low maintenance. The worm payload used to infect Windows machines are downloaded and stored as binary files for later analysis. The downloaded payload can also be sent by e-mail to Norman Sandbox, Anubis sandbox, and CW Sandbox for evaluation. Also nepenthes is a scalable honeypot, this is because of its ability to be configured to listen to a numerous number of IP addresses. Nepenthes is modular. It has modules for resolve dns, emulate vulnerabilities, download handlers, submit handlers, trigger events, shellcode handler...

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. The main focus is to collect the malware binary, download it, store it for the further in-depth analysis, so nepenthes is not designed for any human interaction. Nepenthes does not emulate the full services for the attacker to interact with because the main idea is to offer only as much interaction as is needed to exploit a vulnerability, this can be considered as one of the limitations of low-interaction honeypot because in this case it’s easy for the advanced botnets to detect the existence of honeypot.

3. Infrastructure

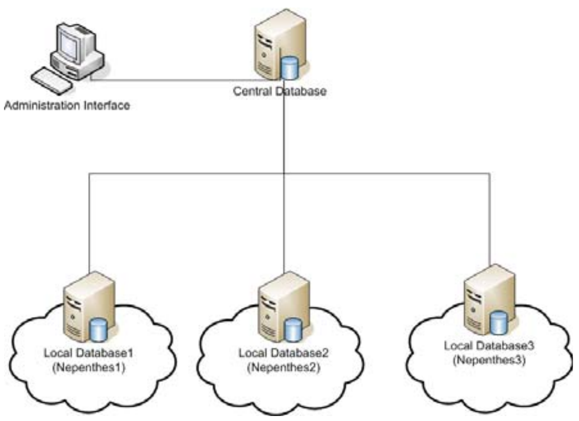
Administration Interface
Central Database
Local Database1 (Nepenthes1)
Local Database2 (Nepenthes2)
Local Database3 (Nepenthes3)

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 the first running Red Hat Enterprise Linux AS release 4, with nepenthes-0.2.2. On the other machine we have installed Linux Ubuntu server with nepenthes-0.2.2 also. The two machines form a distributed farm of nepenthes honeypots each one collects malware samples and send it via http_submission to a central machine we can call it malware samples central 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. It’s possible to face some errors binding ports, that means that the port in listening to the real service for example if you get error binding port 25 that means that sendmail is running and it should be closed to make nepenthes listen to that port.

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. The second NIC 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 Honeypot is set up to listen to a number of ports which the vulnerability modules expect to receive a worm attack through. Nepenthes Honeypot 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 Store the sample in binaries named with its md5-hash and logs a successful download, then the file with the information of some of useful information are submitted to the malware database server.

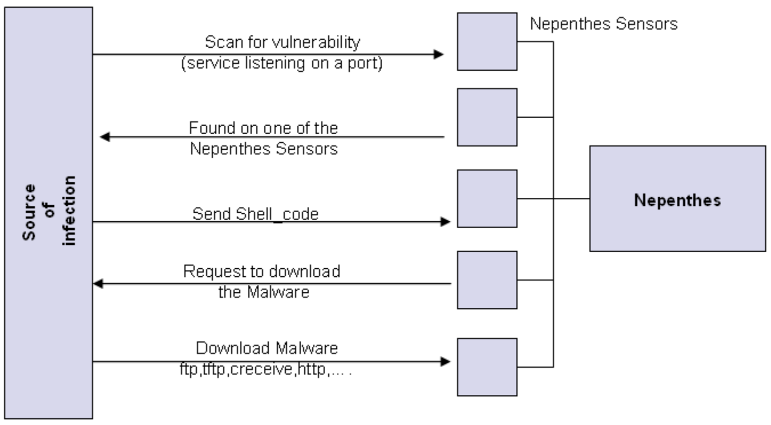
Source of infection
Scan for vulnerability (service listening on a port)
Nepenthes Sensors
Found on one of the Nepenthes Sensors
Send Shell_code
Nepenthes
Request to download the Malware
Download Malware ftp,tftp,creceive,http,... .

Figure 2: Nepenthes honeypot malware collection steps

All nepenthes nodes are connected to the central machine via http_submission, This machine collects all the information from the distributed nepenthes network. The central machine runs the administration Interface and is capable to submit the collected malwares to online sandbox systems for analysis and to receive the analysis reports from them, in addition to scanning the malwares with some known antivirus systems. The http_submission is implemented as a PHP code to be requested by other honeypot nodes by configuring the nepenthes honeypot to request this http page 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 can be collected many times from the source of infection, 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 new 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 also two new different malware samples but this time both the new collected malware samples haven't been collected by the first nepenthes honeypot.

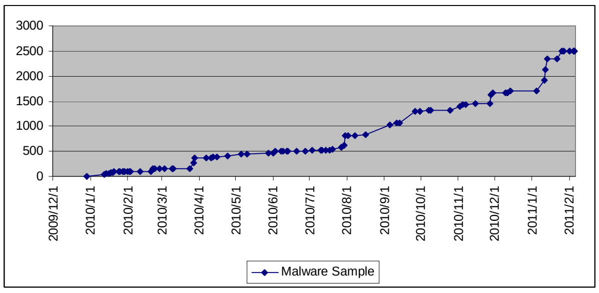

Figure 3: The total number of the collected malware binaries

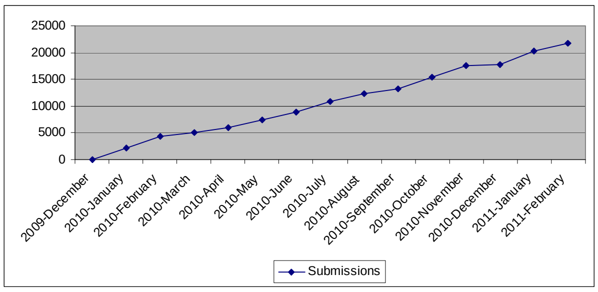

Figure 4: The total number of the successful download of malware binaries

Figures 3 and 4 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. The following table show top-ten downloaded malware samples according to the number of submissions.

Malware md5	Number of submissions	First date	Last Date
7d99b0e9108065ad5700a899a1fe3441	4992	2010/2/27	2010/11/15
98eb0fdadf8a403c013a8b1882ec986d	1844	2010/2/11	2010/9/22
1b7012e6f8abd316360694544074033b	1611	2011/1/3	2011/1/4
fb486908b086c67488dab1deb871f706	1235	2011/2/5	2011/2/5
2aa635dda735bbbb560b12a10f6a764c	746	2010/1/31	2010/2/1
fd28c5e1c38caa35bf5e1987e6167f4c	680	2010/6/12	2010/11/3
7d3fcccb077e7fe87e3f5d3483bc6f0f	555	2010/6/13	2010/7/9
1085f60dabfe6df63ec98ae3ad2860d0	491	2010/1/16	2010/2/2
4f86d95b1b57bd0f5f2e288de68547a1	458	2010/1/29	2010/2/4
e269d0462eb2b0b70d5e64dcd7c676cd	433	2010/2/11	2010/8/4

Table 1 : Top-ten malware submissions

5. Malware analysis

Depending on antivirus gives a little information about the collected 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

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 table in the appendix shows a brief description of some randomly selected malwares that show botnet behavior.

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. The future work includes doing the complete work automatically.

7. 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] 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
[4] Michael Bailey1, Jon Oberheide1, Jon Andersen1, Z. Morley Mao1, Farnam Jahanian1,2, and Jose Nazario2. Automated Classification and Analysis of Internet Malware.

[5] 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)
[6] 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)

[7] 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)

[8] 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)

[9] Nepenthes – Finest Collection. Available at: http://nepenthes.carnivore.it.
[10] http://anubis.iseclab.org/
[11] http://www.joebox.org/
[12] http://www.kaspersky.com/

Appendix I: The table shows a brief description of some randomly selected malwares that show botnet behavior.

Malware md5	Source IP	date	Kaspersky	clamav	Behavior analysis					
					DNS?	DNS IP	Country	HTTP	IRC server	Opened listening ports
b3138b807d340e5e daf68e732ceb6c13	193.198.84.204	2010-07-10	Backdoor.Win32.Rbot.sr	Trojan.Agent.ND	dns.aswend.com	70.107.249.167			70.107.249.167	21, 80, 113
7f60162c2c0bd2cc 7531e51328e98290	115.83.108.240	2010-01-14	Net-Worm.Win32.Padobot.n	Worm.Padobot.N	moscow-advokat.ru					113, 3067, 7828, 6255
	115.163.62.37	2010-05-31			coins.dal.net	194.14.236.50			194.14.236.50	
					diemen.nl.eu.undernet.org	194.109.20.90			194.109.20.90:6667	
1f8a826b2ae94daa 78f6542ad4ef173b	37 different sources	2010-01-14 ~ 2010-08-15	Backdoor.Win32.Rbot.aftu	Trojan.SdBot	botz.noretards.com	128.111.73.201				80
1085f60dabfe6df6 3ec98ae3ad2860d0	211.160.112.7 210.240.41.10	2010-01-16 ~ 2010-02-02	Trojan.Win32.Buzus.cvzu	Trojan.Buzus	ss.ka3ek.com	109.196.130.50	RU		109.196.130.50	14999
					ss.nadnadzzz.info				109.196.130.50	
					ss.MEMEHEHZ.INFO				109.196.130.50	
					ss.memehehz.info				109.196.130.50	
					go.microsoft.com	65.55.57.251	US	65.55.57.251	65.55.57.251	
					www.ieaddons.com	120.136.35.139	US	120.136.35.139	120.136.35.139	
					www.microsoft.com	65.55.12.249	US	65.55.12.249	65.55.12.249	
					worker-24.seclab.tuwien.ac.at	128.130.56.24	AT			
					ss.nadnadzzz.info	67.43.232.36	CA		67.43.232.36	
c4daa264460d246e 88991f0aef4a93e2	121.145.120.165	2010-06-02	Trojan.Win32.VB.aizl	unknown	http communication without DNS		US	65.55.57.251		
0368ff583a3f118a 16a72fd6c53e6508	211.44.197.72	2010-02-03	Trojan.Win32.Buzus.cvzu	Trojan.Buzus	ss.ka3ek.com	109.196.130.50	RU		109.196.130.50	
					ss.nadnadzzz.info					
					worker-24.seclab.tuwien.ac.at	128.130.56.24	AT			16549
2aa635dda735bbbb 560b12a10f6a764c	211.44.197.72	2010-01-31	Virus.Win32.Virut.n	W32.Virut.da	proxim.ircgalaxy.pl	83.133.119.206	DE			
					ss.MEMEHEHZ.INFO	109.196.130.50	RU			
					ss.nadnadzzz.info					
					ku.perfectexe.com	222.170.127.203	CN			
					image.perfectexe.com	222.170.127.203	CN	222.170.127.203:80		
					kdddaber.com	91.217.162.178	UA	91.217.162.178:80		
							CN	60.190.222.131:81		