UNIVERSITY OF KENT

Early Deliverable

CO880 – Project and Dissertation

Yann Ferrere
ID 15906875
yf57@kent.ac.uk

Summary of the document

The aim of this document consists of performing a review of related work linked to my dissertation, which is a Systematic Analysis of Attacks on a Honeypot.

Table of content

1. Honeypots

In this section, we will review articles concerning Honeypots, a technic allowing security researcher to analyses attacks on Linux server. First of all, we will see articles that achieve an overview of honeypot technology. Then, we will examine articles performing case of studies where authors analyze attacks through a honeypot system.

1.1. Overview

This sub-section present two reviews of articles presenting an overview of honeypots systems installed on Linux machines.

The Honeynet Project: Trapping the Hackers [1].

This paper presents the honeynet project, a security research organization composed of international security professionals. The purpose of this organization is to analyze black-hat community's attacks by examining their tools, tactics and more generally their behaviors. Based on this willingness to collect data from attacks and learn from them they decided to create a network, also called honeynets, in order to perform this task. A honeynet is basically a type of honeypot which consists of putting in place a system (without any production value and that doesn't contain any sensitive data) designed to be attacked and/or compromised. They can be used by an organization to improve its security (production honeypots) or by the researcher to gathering information on attackers (research honeypots). However, authors of this article underline that in all case of usage, a honeypot can replace security technologies because of the risk to contain vulnerabilities and being exploited. Finally, honeynets are research honeypot that doesn't emulate any services, but it implements real systems with "hidden" software allowing to manage and capture data from attacks. Consequently, the risk of this type of honeypot is that attacks have to be limited to the tested environment and don't impact to any other systems (Data control). But it also consists to ensure that data from attacks can be detected and capture even if they are obfuscated or encrypted (data capture). One improvement for this paper could have been to perform a case of study of a honeynet in a real case scenario and detail more precisely what data can be captured and how to interpret them.

Anti-Honeypot Technology [2].

The starting-point of this article is that spammers continually scan servers with public IP address for multiple reasons, but most of the time they use open relays with obscure IP in order to hide the origin of the scan. However, honeypots can be used to collect connection information such as IP address origin and other data about spammer's, helping to unmask them. Consequently, spammers and more generally attackers try to detect them.

This paper also underlines that it exist multiple types of honeypots that emulate minimal servers services to fully functional support. All services provide on those honeypots are more or less well-implemented. Attackers who want to detect a honeypot will send multiple requests to a specific service and analyze responses in order to check if it corresponds to a normal usage of the original implementation of the tested service. Consequently, honeypots have to implement in a perfect manner the emulated service to complexity their detection by attackers.

Finally, the author present a tool, called Honeypot Hunter, used to detect honeypots by checking their implementation of a mail server. However, this article doesn't provide any practical detection demo of this tool in a real case scenario.

1.2. Case of studies

This sub-section present review of articles that put in place Honeypots system in order to collect data from remote attacks on Linux machines.

1.2.1. Attackers profiles and behaviors

In this part, we will see articles with an analysis of their results of their Honeypots' gathered data, focused on attackers' behavior and profiles.

Profiling Attacker Behavior Following SSH Compromises [3].

This paper consists of explaining and analyzing data from Linux server attacks that authors have obtained by using ssh honeypots. Indeed, they put in place 4 identical Linux operating systems and installed a modified OpenSSH service to collect information from attempted logins and other tools such as syslog-ng and strace to get more data about the attack. In order to attract attackers to compromised their servers, they associate simple identifier/passwords for multiple users (root or not) to facilitate the attack. The authors were able to perform an analysis of login and password used to access by the ssh protocol the Linux server. Username such as root and basic password such as "123456" were commonly used to attempt a connection. Consequently, authors recommended to disable root ssh access and apply a strong password policy. Furthermore, authors decided to classify attackers' command lines by using 7 criteria (CheckSW, Install, Download, Run, Password, CheckHW, ChangeConf). However, this paper doesn't perform any analysis of malware downloaded. Moreover, they focus on listing compute statistics but the explanation of why an attacker use specifics technics in order to comprise a server, are not clearly noticed.

Analyzing the process of installing rogue software [4].

This paper has been written in order to report results of an experiment performed to understand and analyze malicious behavior following a remote Linux server attack. In order to realize this analysis, authors of this article have implemented four honeypots. Those honeypots are four Linux target computer using SSH with a simple password, easy to find for an attacker. Moreover, that machine also contains monitoring software such as a modified OpenSSH server, syslog-ng, and strace to gather a maximum of data during a potential malicious access to the computer.

By using this system authors have collected a large amount of data and choose to divide them by using two concepts. The first one is called "session" and define all distinct SSH interaction between an attacker and one of their honeypot. The other one called "action" corresponds to a sequence of command line execute by an attacker to do a specific task such as gathering information or install malicious programs. Furthermore, those honeypots were configured to record all network traffic and consequently, authors were able to collect malware downloaded by attackers on compromised computers. Consequently, they decide to classify them in categories corresponding to their types such as IRCBots, Rootkit, and Flooder. This classification has been realized at 50% by using VirtusTotal and the 50% remaining percent by a manual identification using their source code. However, this paper doesn’t provide more details about the manual malware reversing.

Lessons learned from the deployment of a high-interaction honeypot [5]

This paper consists of present results of an experimental study based on the observation of attackers’ behavior that attack and compromise Linux machine. In order to collect data needed to perform this analysis, they use high-interaction honeypots. Compared to a low-interaction honeypot, that only allow attackers to scan and send requests to a Linux machine, high-interaction honeypots makes possible to analyze behavior one they have compromised the server by allowing SSH connection for example. During their experiment, authors decided to put in place Linux virtual machine (with VMware) and choose to configure an SSH access with weak passwords in order to create a vulnerability easy to exploit by attackers. From collected data, authors of this article were able to compute statistics from SSH connections attempts. For instance, a ranking of a most popular combination of user and password has been realized. Furthermore, this paper provides a classification of SSH connection attempts into three groups. The first of them is IP used to perform the dictionary attack on the SSH port. Then, we can find the group of IP used to realize the intrusion by using credentials provide by the brute force attack. Finally, all other IPs are considered as a scanner and all other “passive” server requests. Finally, authors of this article realize an analysis of attackers’ behaviors in order to determine there if those attacks were performed by human or programs but also quantifying skills of attackers. However, this paper provides a very brief analyze of malicious programs downloaded by attackers without performing any precise static or dynamic reverse engineering of them.

1.2.2. Analysis of collected malwares

In this part, we will review articles with an analysis of their results of their Honeypots’ gathered data that also performed an analysis of downloaded malwares.

Characterizing Attackers and Attacks: An Empirical Study [6].

The starting-point of this article is that network protection's solutions of an organization must reflect the threats. However, it is complex to determine what are the threats and where there are coming from. Consequently, in order to answer this question authors of this article tried to analyze the behavior of hackers and the scheme of their attacks by implementing a honeypot system. The concept of this system consists of using an SSH access, on a public IP address, as an entry point for hackers. This SSH access redirects all attackers into a honeypot, a fake Linux environment in this case. From there, it has been possible for them to collect multiple data such as keystrokes, IP address, network traffic, rogue software... Subsequently, those data has been analyzed by using a notation system in order to identify hacker's profiles. This system uses ten objectives criteria to determine attacker skill. Those criteria are based on four main characteristics such as the attacker's discretion or their rogue software's knowledge.

However, this article specifies that more a hacker execute an action from this list, more he/she is skilled. This point could be interesting to comment because all actions assessed are not always needed to realize a good attack. For example, criteria « Restore Deleted Files », « Check Presence », « Change System » or « Create New User » can be useful in some case but not all the time. Consequently, it is complex to characterize hacker's skills as good or bad only by judging really specific actions and not the entire attack.

An experimental study of SSH attacks by using Honeypot decoys [7].

This paper consists of an analysis of collected data from SSH honeypots configured with weak usernames and passwords, intended to be found by attackers using brute force attacks. Authors of this article decided to use an SSH honeypot called "Kojoney SSH honeypot" in its version 0.0.4.2 and all data accumulated during this experiment were stored in a central database.

By using data stored in this database, they were able to perform an analysis of two main aspects of a Linux server attack. First of all, authors of this articles performed a general overview of attackers' behavior by analyzing gathered data from their honeypots. These parts consists of determining where come from attacks (by using geographic location of IP addresses), what are the most common username/password combination, what are the OS used by the attackers and what is the global activity of attempted connection on the SSH ports (ratio successful and failed authentication by their source IP). The second asset aspect concern the activity inside the Honeypot after a successful connection on the SSH port. In this part, authors were able to determine favorite's commands used by attackers but also collect malware downloaded by them. After a more detail analysis of that malware they authors retrieve their "antivirus names", a platform which is necessary to run them and sometimes botnets information (channels, username, password,…).

Although, that the analysis of gathered data was detailed authors doesn't explain what are the steps used to perform all of their malware analysis.

Analysis and visualization of SSH attacks using honeypots [8].

This paper present results of a research honeypot's operation consisting to trap attackers who target an SSH service to compromise a remote server. First of all, authors realize a brief overview of what types of honeypots exist. Indeed, a honeypot can be used by the company to serve a production system of a company to help system administrator to examine attackers and detect weakness in their system. However, honeypots can also be used for the purpose of collecting and analyzing recent attacks in a real context. Moreover, honeypots can also be classified in two other groups, low or high interaction level. The first one define a honeypot only uses to analyze requests send to a server and the second one allow connection on the machine to perform a detailed analysis of the attack.

Furthermore, the authors of this paper introduce an easy manner to visualizing data collected by all sensor set on the honeypots via Kippo-Graph. This tool consists of generating a web page containing multiple schemas based on the data stored in a MySQL database. By using scheme generate with this tools, we are able to easily determine useful information about attacks on the honeypot such as top 10 of usernames and passwords attempted, overall success ration of connection attempted, the frequency of connection… This programs also display on a map IP addresses with their respective country codes a useful information in order to classify and understand attacks. Moreover, authors of this article used

Kippo Honeypot to perform this experiment. This powerful tools allowed them to collect command lines used by the attackers and downloaded malware. Based on those gathered malicious programs, there were able to determine the objectives of those programs (IRCBot, Bouncer, Flooder,…) and associate recover their name (psyBNC, FloodBot,...) to perform some research and finally, understand how they work. However, this process to associate downloaded malicious programs to their name and the technics used to analyze them are not specified precisely in this paper.

2. Malwares

In this section, we will review articles concerning Malwares, malicious software which are used by attackers to compromised machines and perform actions remotely.

2.1. Overview

This sub-section present a review of an article presenting an overview of malicious softwares used by attackers.

Introduction Linux based malware [9].

The aim of this paper is the analysis of malware on Linux operating system. Indeed, as explain by the authors, Linux systems are particularly vulnerable to malicious software because of their minimal defenses against them. Moreover, those systems are increasingly used in particular in IoT (“Internet of Things”) embedded devices and servers. In this paper malware are defined as “Code used to perform malicious actions”. Moreover, authors underline that malware needs to exploit a vulnerability in order to compromise a machine. This article also specifies that malicious programs can embed, in many cases, several functions such as propagation mechanisms and command and control functions. In addition, authors define the notion of the payload as a code to exploit a particular vulnerability, generally present on malicious programs in order to infect the targeted system.

In order to reduce the number of malware attacks, several systems can be put in place to filter traffic to and from the machine to block and/or detect malware. For instance, web application firewall (WAF) configured as an HTTP reverse proxy can protect a web server from potential attacks. In the same way, network-based monitoring programs such as SNORT are necessary to assure a minimal protection of a system.

However, this article could be improved by describing malware classification technics in order to report and monitor in the best manner potential malware attacks on its Linux system.

2.2. Case of study

This sub-section present review of an article that performed an analysis of a known malicious software called Psyb0t.

Psyb0t Malware: A step-by-step Decompilation case study [10].

The authors of this paper perform an analysis of the malware PsyB0t by using a decompilation technic which consists of retrieving a readable code based on the machine code of a binary. However, usage of decompilation tools on malware is a challenging task. Indeed, a malware doesn't respect application's standards because of their objectives to be complex to reverse engineer. For instance, a malware has usually stripped symbols, its code is obfuscated and can contain polymorphic code. Moreover, malware targets multiple platforms and consequently can be compiled for each of them. During the analysis of the PsyB0t malware, authors were able to unpack it by using UPX tool. Then, they process the whole executable to retrieve strings (printable characters that terminated by a zero byte '\0') and some symbols by analyzing data sections of the binary. Next, authors used the address of the program's main by using its internal computer-specific database or using a heuristic detection. Based on that they were able to create a control-flow graph that examines all branch instructions to recognize conditional and unconditional branches, functions calls and returns from functions. At this steps the assembly code contains many redundant instructions, consequently, authors performed optimizations such as detecting multiple instructions that correspond to a known function and replaces them by the function name. Finally, the last step is to use the produced input of the last step to produce a code in the specified language syntax (C or Python like). By analyzing the code authors were able to detect functionalities of this malware (DDoS, brute-force, downloading of files,…) and much other useful information.

Finally, this paper produces a complete analysis of the PsyB0t malware, however, it would have been interesting to perform an overview of decompilation tools and justify their choice to use a retargetable decompiler that is being developed within the Lissom project.

2.3. Analysis methods

This sub-section present review of articles that examine methods in order to perform analysis of malicious softwares by security researchers.

2.3.1. Static and dynamic analysis

In this part, we will review an article concerning static and dynamic reverse engineering methods allowing security researcher to analyze malwares.

Automating Linux Malware Analysis Using Limon Sandbox [11].

This paper focuses on the presentation of Limon Sandbox, an automating Linux Malware analysis tool developed by Monnappa K A. This sandbox, written in python, allow the user to automatically collects, analyzes, and generates reports containing information of the processed malware. Moreover, this software aggregate knew malware analysis tools such as YARA, VirusTotal, and Volatility memory forensics framework. Limon sandbox performs three common types of malware analysis. First of all, a static analysis of the malware is operate without executing the malware. This step can able the user to learn multiple useful information about the malware such as the file type, its cryptographic hash, Strings embedded within the file, obfuscation methods,… Then, a dynamic analysis is performed. This step consists of executing the malware sample in a safe environment (a virtual machine without any personal data) and where it is possible to monitor as it runs. By using these technics a security researcher will be able to get more information impossible to gather with a static analysis such as process, file system and network activity. Next, Limon sandbox allows users to realize a memory analysis of the previously ran virtual machine. This step allows users to extracting forensics information such as running processes, network connections, loaded modules, API Hooking,…

Finally, these tools generate a report in text format that contains all previously collected information. This paper provides a complete explanation of Limon Sandbox tool, however, it might be interesting to detail how can be interpreted results generate by the report.

2.3.2. Unpacking

In this part, we will review an article concerning packing method that malicious softwares developers used to bypass antivirus detection.

Revealing Packed Malware [12].

The starting point of this article is that during the past few year malware threats have increased significantly. In order to protect users against this threats multiple company's developed antivirus software to detect and block malicious programs that try to compromise a computer. However, malware authors use packers in order to evade their detection by antivirus. The author defines a packer as a "binary tools that instigate code obfuscation". Currently, modern malware can completely bypass personal firewalls and antivirus scanners by implementing this technic. A malware can use a packer software in order to compress and encrypt itself. Then, when the packed files are loaded into the memories it is able to restore the original executable and run its malicious part. Consequently, antivirus that tries to match the signature of the packed malware with the known virus does not detect any risk.

Packers can be classified into four categories:

- Compressor: that shrink file size with little or no anti-unpacking tricks (i.e. Upack UPX).
- Crypter: that encrypt and obfuscate the original executable and prevent to be unpacked without any compression (i.e. Yoda's crypter).
- Protectors: that combine features from compressors and crypters (i.e. Armadillo).
- Bundlers: that pack a software package of multiple executable and data files into a single bundled executable file (i.e. PEBundle).

In order to answer to this threat of packed malware, security researchers and antivirus editor use unpacking technics to inspect the original executable signature. One manual technics is to use debugger tools such as Ollydbg. However, automate packers detection exists and are usually used by antivirus that develops static unpackers. This type of tool consists of dedicated routines to decompress and decrypt executable packed by specific packers without executing the suspicious programs.

3. References

[1] L. Spitzner (2003). The Honeynet Project: Trapping the Hackers. [Online]. Available from: IEEE Xplore. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1193207 [Accessed 5 March 2016].

[2] Krawetz, N. (2004). Anti-honeypot technology. [Online]. IEEE. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1264861 [Accessed 19 June 2016].

[3] D. Ramsbrock, R. Berthier, M. Cukier (2007). Profiling Attacker Behavior Following SSH Compromises. [Online]. Available from: IEEE Xplore. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4272962 [Accessed 5 March 2016].

[4] R. Berthier, J. Arjona, M. Cukier (2009). Analyzing the process of installing rogue software. [Online]. Available from: IEEE Xplore. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5270293. [Accessed 5 March 2016].

[5] E. Alata, V. Nicomette, M. Kaaniche, M. Dacier, M. Herrb (2006). Lessons learned from the deployment of a high-interaction honeypot.[Online]. http://homepages.laas.fr/kaaniche/documents/Alata2006/alata-EDCC6.pdf. [Accessed 5 March 2016].

[6] G. Salles-Loustau, R. Berthier, E. Collange, B. Sobesto, M. Cukier (2011). Characterizing Attackers and Attacks: An Empirical Study. [Online]. Available from: IEEE Xplore. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6133079. [Accessed 5 March 2016].

[7] Kheirkhah, E., Mehdi, S., Amin, P., Sistani, H. A. and Acharya, H. (2013). An experimental study of SSH attacks by using Honeypot decoys. [Online]. Available from: http://www.indjst.org/index.php/indjst/article/download/43618/35034 [Accessed 19 June 2016].

[8] Koniaris, I., Papadimitriou, G. and Nicopolitidis, P. (2004). Analysis and visualization of SSH attacks using honeypots. [Online]. IEEE. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6624967&tag=1 [Accessed 19 June 2016].

[9] An Introduction to Linux-based malware (n.d.). [Online]. SANS Institute. Available from: https://www.sans.org/reading-room/whitepapers/malicious/introduction-linux-based-malware-36097 [Accessed 19 June 2016].

[10] Lukášˇ durfinaJakub, Křoustek, P. and Zemek (n.d.). Psyb0t Malware: A step-by-step Decompilation case study. [Online]. Available from: https://www.computer.org/csdl/proceedings/wcre/2013/9999/00/06671321.pdf [Accessed 19 June 2016].

[11] Monnappa, K. A. (n.d.). Automating Linux Malware Analysis Using Limon Sandbox. [Online]. Available from: https://www.blackhat.com/docs/eu-15/materials/eu-15-KA-Automating-Linux-Malware-Analysis-Using-Limon-Sandbox-wp.pdf [Accessed 19 June 2016].

[12] Yan, W., Zhang, Z. and Ansari, N. (2008). Revealing packed Malware. [Online]. IEEE. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4639028 [Accessed 19 June 2016].