Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Enhancing Honeypot Deception Capability Through Network Service Fingerprinting

To cite this article: R N Dahbul *et al* 2017 *J. Phys.: Conf. Ser.* **801** 012057

Related content

- Analysis Of Using Firewall And Single Honeypot In Training Attack On Wireless Network
  Tengku. Mohd. Diansyah, Ilham Faisal, Adidtya Perdana et al.
- Research on Network Defense Strategy Based on Honey Pot Technology
  Jianchao Hong and Ying Hua
- Network traffic intelligence using a low interaction honeypot
  Tendai Nyamugudza, Venkatesh Rajasekar, Prasad Sen et al.

Recent citations

- Oleg Surnin *et al*
- Detecting indicators of deception in emulated monitoring systems
  Kon Papazis and Naveen Chilamkurti

Enhancing Honeypot Deception Capability Through Network Service Fingerprinting

R N Dahbul[1] **, C Lim**[2] **, J Purnama**[3]

Department of Information Technology, Swiss German University, Edutown BSD City, Tangerang, 15339 Indonesia.

E-mail: [1]rasyid.dahbul [at] student.sgu.ac.id, [2]charles.lim [at] sgu.ac.id, [3]james.purnama [at] sgu.ac.id

Abstract. Honeypot is designed to lure attackers away from the computer resources the attackers are trying to compromise. In addition, honeypot also tracks attacker's activities and helps researchers learn about their attack patterns. However, honeypot can also be identified by attackers using various fingerprinting methods. In this research, we use threat modeling to identify potential threats that reveal its existence which made honeypot ineffective. Various countermeasures are discussed and the proposed countermeasures have proved effective to enhance the deception capability of the honeypots we tested.

1. Introduction

Internet security threats continue to rise at an alarming rate as more and more system and application vulnerabilities continue to increase [1]. These threats are contributed by common Internet services such as Email & Web services and by the rise of mobile devices and Internet of Things usage as well, as demonstrated by the recent Distributed Denial of Service (DDoS) attack [2]. Attackers persist to find new ways to probe, attack and compromise systems and/or applications they are targeting. Firewalls and Intrusion Detection Systems (IDS) have always been used intensively to defense against known Internet security threats and attacks to the systems connected to the Internet. However, these security protection does not protect against unknown threats, that over the time may cause even larger damage to the system or system owner.

Honeypot provides an exceptional way to detect these unknown threats, which may include possible intrusion by attackers [3]. It can gather information useful attack patterns, which may help system administrator to learn in hardening systems to provide better security defense. With the increased usage of honeypots [4, 5], attackers also prepare their ways to defeat them. Attacker often uses fingerprinting techniques to probe system's profiles including system version, open services and their vulnerabilities. Using the same techniques, honeypot system could also be identified and thus make the system useless. In the worst case, the attacker can create a false intrusion to the honeypot causing disturbance to system.

The objective of this research is to explore various fingerprinting techniques at different Open Systems Interconnection (OSI) layers to detect the presence of honeypots. We use information gathered using these techniques as the new threat sources to our threat model, used to analyze these new threats. The new countermeasures are designed and applied to the existing honeypot systems. Security experts are then invited to review and validate the applied countermeasures.

2. Literature Review

2.1. Cyber Deception

Deception is an approach to deceive or mislead users into believing that the given information is beneficial to the user, while at the same time protecting the real assets and the owner's intention [6]. The idea of deception is not new, but with the more sophisticated attacks may bypass the common protection at the perimeter such as Firewall and IDS. The use of Cyber deception, hence, engages attackers using the useless resources and spend most of their time with the provided resources [7]. For the deception to be effective, there are 6 principles of deception to be taken into consideration [6]:

- **Focus** - The deception must have a target and it must have an effect on the target's decision. It can be targeted against groups or individuals.
- **Objective** - The deception must cause the target decision's to be affected and take suggested actions, not just observing the trap.
- **Centralized planning** - The operation must be centrally planned and move the unity to achieve a certain objective.
- **Security** - The creators of the deceptive system must halt any knowledges that may alert the target's of the trap.
- **Timeliness** - The deception requires patience, it needs careful timing and synchronizing.
- **Integration** - The deception needs to cooperate every system that it supports.

The goal is to have the attacker to think that they are in a real system in order for the operation to successfully continue. For cyber deception, there are many ways to deceive attackers, one of the most popular tools to achieve this purpose is honeypot [6].

2.2. Honeypot

Honeypot is essentially a decoy system that stores many "valuable" information that attackers are looking for and lures them to stay engaged in the system [8, 9]. Honeypot does not block the attackers to explore the system, instead, it simulates network services to interact with the attackers and record all the activities in the system. Since honeypot is intentionally placed in the network, any connections to the honeypot is considered as suspicious [3]. The lesson learned from monitoring attack patterns and distribution of malware during the attack can be very useful to provide real countermeasures.

Even though honeypot has the above mentioned advantages, like any other systems, the attacker can eventually find out the system is a honeypot. Once identified, the attacker will try to compromise the system and use it as a stepping stone to the next system.

3. Threat Modeling

As discussed, one of the counter deception strategy is to use fingerprinting techniques [10] to identify the existence of honeypots. Threat modeling is a process to analyze the security and vulnerabilities of an application [11] and it is commonly used to analyze security issues of an application or network services by identifying system's vulnerabilities [12, 13]. In our research, Threat modeling is used to enumerate all possible security threats by fingerprinting honeypots, as shown on Table 3.

There are 3 models of direct attacks to the honeypots [14, 15]: flooding the honeypots by sending garbage packets, compromising the honeypot to become a bot and using the honeypot as a spying platform. These attacks can be performed once the honeypot is detected through fingerprinting. To counter these threats, we need to evaluate each of the fingerprinting results for each OSI layers' parameters, including TTL (Time-to-live) data in layer 3, TCP or UDP ports on layer 4 and so on.

Table 1. Honeypots Threat Modeling

Attacks	**Attacks Description**	**Root Causes**	**Attacks Surface**	**Possible Mitigation**
Poisoning	Flooding the Honeypots	Honeypot fingerprinting	Multiple connections with same sources with no limit	Using LaBrea Tarpit
Compromising	Staging Internal Attacks	Honeypot fingerprinting	Honeypots that allow any activity entering or leaving the honeypot	Using Honeywall
Learning	Gaining private information	Honeypot fingerprinting	The honeypots are allowed to access the root system	Limit privileges

4. Honeypots Fingerprinting

Attackers fingerprint their targets before committing any intrusion; identifying their target makes it easier to infiltrate using known vulnerabilities. If the honeypots are detected with these attackers, it will be deemed useless as it needs interaction from attackers. Attackers often use port scanning tools such as NMAP or others to detect running network services. In our research, we setup 2 system: the honeypot system and the real system (such as real http server or ssh server) and fingerprinting is performed on both systems. During both system fingerprinting, we are looking for specific information, in particular information from layer 4 and 7 of OSI layers, on each network services, such as network application banner, application version, network protocols, etc. The fingerprinting results become the inputs to our threat modeling, the enhancement is done by comparing the result of real system and honeypot system. The honeypots used in this research are: HoneyD, Dionaea, Kippo, and Glastopf [16]. We chose to install these honeypots on a virtual machine for ease of portability and maintenance. Table 2 describes the overview of the honeypot weaknesses, divided into honeypot groups and focus. Based on these weaknesses, various enhancement is designed and applied.

5. Honeypots Enhancements

In this section, each of the enhancements is being implemented on the relevant honeypots.

5.1. Open Ports Fix

Only open ports that a real server would open will quickly fix the problem. However, opening port should be done carefully since opening port, such as port 17300 on the system may indicate the presence of Kuang2 worm. Opening well-known ports such as port 21, 80 and 143 will make it less suspicious for hackers to attack.

5.2. HoneyD IIS Directory Traversal Exploit Fix

To fix the Microsoft Internet Information Server (IIS) problem, the fix should be done carefully. The problem with the Honeyd IIS scripts is the timestamp itself, hence it is important to modify the timestamp similar to the one of a real server. The modification should also include folder name to make the honeyd appearance looks more closely to the real IIS.

5.3. HoneyD HTTP Service Fix

To fix the outdated and invalid replies of honeyd's HTTP service, we need to find which script that honeyd uses to emulate the HTTP service. The script that honeyd uses to emulate the HTTP service is named "apache.sh" on the scripts folder (`/usr/share/honeyd/scripts/unix/linux/suse8.0`). After full examination of the scripts, there is a problem with the script, it does not implement "POST", "PATCH", "DELETE", "COPY", "LINK", "UNLINK" and "PURGE". This implementation provides an easy way to fingerprint the system as honeyd by the attacker. Comparing and examining the response of the real Apache version 1.3.23, we can recreate the replies on the HoneyD HTTP Services.

5.4. Dionaea FTP Service Fix

The naming of the FTP service of Dionaea is different from any other real services. To fix this problem, changing the configurations into a more believable services. For example, changing the FTP banner into something else is a start to enhance the honeypot from fingerprinting activities. To make it hidden from service scan, we changed the "Welcome to the ftp service" header in the configuration file into a real Microsoft FTP service header, i.e. "Microsoft FTP Service."

5.5. Dionaea SMBD Fix

NMAP scan results revealed that Dionaea's SMBD service uses particular names on Workgroup and Homeuser names. If the server has a domain name of WORKGROUP and the computer name of HOMEUSER-XXXXXX, where X is any random character, then it is detected as a Dionaea SMBD service. To evade this fingerprint, modification should be performed on the Dionaea configuration file. For example, we change the name OemDomainName into "MIDOMINIO" and the ServerName into "EQIUPO-TEST."

5.6. Dionaea MS-SQL Service Fix

NMAP detects MS-SQL fingerprint by checking the response of a TDS package that the Dionaea responds. Dionaea sends this response to any client when they are connecting to its database. NMAP detects it by comparing the value of the in the field named "Token Type" during the login process into the Dionaea's MS-SQL service. The modification could be performed on the file of the MS-SQL service script is named "mssql.py" in the scripts folder. Changing the "`r.VersionToken.TokenType`" value from "0x00" to "0xAA" will fix the problem.

5.7. Kippo Regex Ping Fix

To fix the problem, the file that emulates the ping function need to be located. It was found that the ping function is emulated by the script named "ping.py". To repair the issue, several lines of codes need to be modified to restrict the ping command only works within the set of network groups. If the target ping is not in the network group, then it is seen as "Unknown host." After the modification, pinging using an impossible IPP address such as "999.999.999.999" returns with an error [17].

Table 2. Honeypot Weaknesses and Enhancements

Honeypot	**Layer**	**Focus**	**Vulnerabilities**	**Enhancements**
HoneyD	4	Open Ports	Suspicious open ports	Opening & Configuring essential ports
	7	HTTP	Invalid replies	Configuring scripts
	7	IIS	Suspicious timestamps	Fixing timestamps
Dionaea	4	Open Ports	Suspicious open ports	Opening & Configuring essential ports
	7	FTP	Detected by port scan	Configuring scripts
	7	MySQL	Detected by port scan	Configuring scripts
	7	SMBD	Detected by port scan	Configuring scripts
Kippo	7	SSH	Commands not working properly	Configuring scripts
Glastopf	7	HTTP	Invalid error reply	Configuring scripts

5.8. *Glastopf LFI Vulnerability Fix*

At first, if any hacker tries to access a folder that is not in the vulnerability group, it will reply "Warning: include(vars1.php)" error, which is a static error message made by the developer. To workaround this fingerprinting exploit, it can be changed to a "Permission Denied" error. It does make more sense to do this, rather just throw a warning error.

Table 2 summarizes the honeypots' weaknesses and their enhancements.

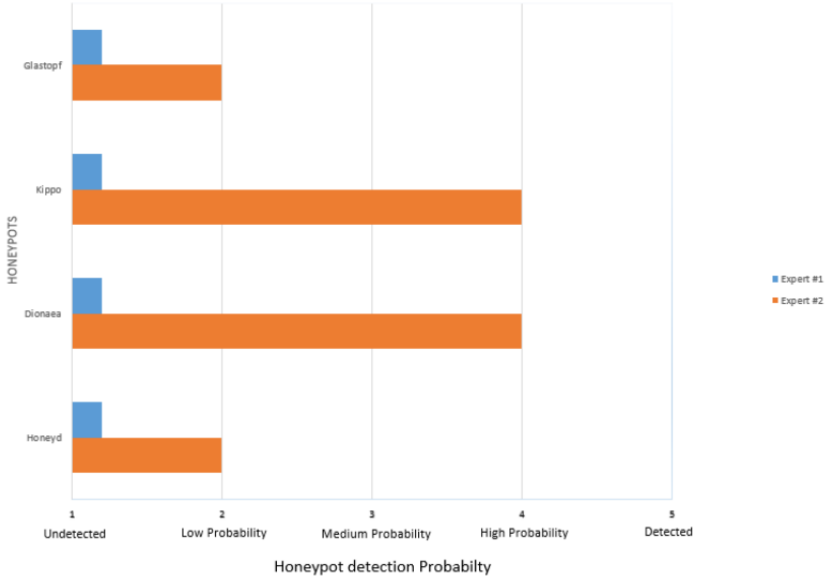

Figure 1. Expert Review results

6. Expert Review

To validate our enhancement results, security experts are invited to review the enhancements. Both of the experts are security professional pen-testers and consultants to various industries, including Banking, Telecommunications and other Enterprises. Security experts were given the freedom to use any security tools to perform fingerprinting on the honeypots. The tests attempted by 2 different experts are using black-box testing method, in which the experts are not given any information about the targets. The result of the tests (detection probability) are divided into 5 categories, i.e. "Undetected", "Low probability", "Medium probability", "High probability" and "Detected". The results show that Glastopf and HoneyD honeypots (with enhancements) were detected with "Low Probability", which indicates that the experts have difficulty in detecting them as honeypots. The other honeypots (with enhancements), Kippo and Dionaea, are detected with "High Probability", which indicates that there are some fingerprinting weaknesses that are need to be addressed. Figure 1 summarizes the security expert reviews on the enhancement of the honeypots.

7. Conclusion

Honeypots being used as a cyber deception tool will continue to evolve. Attackers can detect the presence of honeypot by "fingerprinting" them and inspect the result carefully on different OSI layers. The fingerprinting results of the existing system may indicate whether the existing system is a honeypot, hence, the honeypot should be "hardened" to reduce the chance of the honeypots being discovered by attackers. Threat modeling is then performed to analyze the potential security threats of the fingerprinting results. To enhance its deception capability, system configuration and custom scripts are used. We have successfully enhanced the deception capability of the honeypots to avoid the honeypots being identified by the attackers. Current limitation of our research is still focusing on layer 3, 4 and 7. Our future works may include enhancing the deception capability the system by fingerprinting layer 1 to check for clock skew of the system [18].

References

[1] Rubino D Symantec internet security threat report 2015 URL `https://www.symantec.com/content/en/us/enterprise/othe_resources/21347933_GA_RPT-internet-security-threat-report-volume-20-2015.pdf`

[2] Security K 2016 Hacked cameras, dvrs powered todays massive internet outage URL `https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/`

[3] Spitzner L 2003 *Honeypots: tracking hackers* vol 1 (Addison-Wesley Reading)

[4] Nemade S, Darekar M M A and Bachhav M J 2015 *Journal of Multidisciplinary Studies* **1**

[5] Kao C N, Chang Y C, Huang N F, Liao I J, Liu R T, Hung H W *et al.* 2015 A predictive zero-day network defense using long-term port-scan recording *Communications and Network Security (CNS), 2015 IEEE Conference on* (IEEE) pp 695–696

[6] Bodmer S, Kilger M and Carpenter 2012 *Reverse Deception: Organized Cyber Threat Counter-Exploitation* (McGraw Hill Professional)

[7] Rowe N C 2008 *Cyber Warfare and Cyber Terrorism* 97–104

[8] Provos N *et al.* 2004 A virtual honeypot framework. *USENIX Security Symposium* vol 173 pp 1–14

[9] Sokol P, Kleinova L and Husak M 2015 Study of attack using honeypots and honeynets lessons learned from time-oriented visualization *EUROCON 2015-International Conference* (IEEE) pp 1–6

[10] Aguirre-Anaya E, Gallegos-Garcia G, Luna N S and Vargas L A V 2014 *International Journal of Electrical and Computer Engineering* **4** 848

[11] Shostack A 2014 *Threat modeling: Designing for security* (John Wiley & Sons)

[12] Potteiger B, Martins G and Koutsoukos X 2016 Software and attack centric integrated threat modeling for quantitative risk assessment *Symposium and Bootcamp on the Science of Security* (ACM) pp 99–108

[13] Chang J S, Jeon Y H, Sim S and Kang A N 2015 *International Journal of Distributed Sensor Networks* **2015**

[14] Piller K and Wolfgarten S 2004 *Ernst&Young Risk Advisory Services ppt. Retrieved on November* **16** 2005

[15] Krawetz N 2004 *IEEE Security & Privacy* **2** 76–79

[16] Gorzelak K, Grudziecki T, Jacewicz P, Jaroszewski P, Juszczyk Ł and Kijewski P 2012 Proactive detection of security incidents ii - honeypots Tech. rep. Tech. Rep., ENISA

[17] Oosterhof M 2015 Kippo github pull URL `https://github.com/desaster/kippo/pull/148/commits/19241a374dec3e8d198e7d679ed4e5c6cefe1e2c`

[18] Kohno T, Broido A and Claffy K C 2005 *Dependable and Secure Computing, IEEE Transactions on* **2** 93–108