IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

Virtualization in High Interaction Honeypots

Jaber Karimpour[1], Maryam Mohammadzad[2]

Department of Computer Science, University of Tabriz, Tabriz, Iran [1,2]

Abstract: Nowadays, High Interaction (HI) Honeypot is the hottest area for research because of the major issues such as monitoring the attacker, and also its detectability in computer network. When we talk about the honeypot technology and its characteristics like capturing data secretly or even when we talk about different instances in data capture tools, we should not forget that how it can achieve in honeypot technology. The virtualization is not a new concept in hardware and software development. Virtual Machine Monitor (VMM) is very useful tool for implementing honeypots. In this paper, first, we review the various implementations of VMM-based monitoring systems that are developed to collect information about attacks on HI honeypots. Finally, a comparison is made between each of them.

Keywords: Computer Network Security, High Interaction Honeypot, Virtualization, Virtual Machine Monitor

I. INTRODUCTION

Honeypot is one of the modern technologies in the area of Intrusion Detection System (IDS) that provides a suitable platform for a successful attack [1]. In fact, the main purpose of the honeypot is to gather information about intruder's activities by luring attackers [2]. Honeypots are designed to provide a tempting target to circumvent the attacker. In general, to achieve the idea of deceiving attackers, network administrators need to create a system with hidden data capture tool but looks like an ordinary system [3].

Virtualization is a technology that provide an interface to monitor internal events of guest Operating System (OS) from outside. Therefore it is very good choice to use in HI honeypot. As they provide a strong isolation between the virtual environment and the underlying real system, Virtual Machines (VMs) can be used to improve the security of a computer system in face of attacks to its network services [4]. In fact, the VM can be used as a honeypot and monitoring tool can be placed in out of the VM (Honeypot).

As mentioned above, virtualization is an interested method to honeypot researchers because of providing a way to monitor internal events of honeypot from outside of the vulnerable system. Some HI honeypots are developed recent years. They use virtualization technology to monitor intruder activities in honeypot. This paper focuses on data capturing process of some important HI honeypots that uses virtualization on their implementation such as VMScope [5], Qebek [6], Xebek [7].

This paper is structured as follows: section II recalls HI Honeypot's concepts; section III introduces Virtual Machine Monitors (VMMs); section IV and V details the VMM-based Honeypots, and section VI provides a comparison for Honeypots introduced in the previous sections.

II. HI HONEYPOTS

The main classification of honeypots is based on level of interaction of attackers with them. In this classification honeypots are divided to Low Interaction (LI) honeypots and HI honeypots [1,3,8-10]. "High-interaction honeypot is a machine with standard software suite installed. Administrator of such honeypot usually installs vulnerable versions of network services so it allows attacker to break in the machine, install malicious tools and complete the attack." [11]. A HI honeypot simulates all aspects of an operating system, whereas a LI honeypots simulates only some parts, for example the network stack [12]. In fact, HI honeypots are designed to get full information about a successful attack by providing a real system for attacker. This kind of honeypot needs to a kernel-based data capture tool for watching to attacker and recording her activities include keystrokes, file system modifications, etc. Since data capture tool lives entirely in kernel-space, it is able to access all decrypted data from kernel, but this is a high risk for data capturing because an intruder witch already gained root access is able to attack to data capture tool.

Some important opportunities are provided by HI honeypots. They are listed as follows:

- Distract attackers from more valuable machines on a network [10]
- Provide a platform for studying the methods and tools used by the intruders [1]
- Study the very motives of the attackers [2]
- Provide early warning about new attack and exploitation trends [10]

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

We have several issues with HI honeypot. The challenges of HI honeypot happened in various domains include achieving reliable data and hiding data monitoring.

A. Data capturing Issues

Data capturing is the main component in the networks that are using Honeypot. Based on the location of data capture tool, Honeypot can be monitored either internally or externally [5]. Traditional methods for external monitoring is to capture intruder's activities by recording the associated network data. This approach places packet analysers to record every network packet flowing in and out of the monitored Honeypot. This is a desirable approach because it can be done in such a way as to be invisible to the intruder [13]. "Existing external honeypot sensors (e.g., network sniffers) could be made invisible to the monitored Honeypot. However, they are not able to capture any internal system events such as system calls executed" [5]. Further, all data captured by network sniffing is useless if the intruder utilizes encryption to connect to Honeypot system [14].

To access to the internal system events, kernel-based data capture tools are developed. This type of monitoring tool reside completely in the Honeypot's host. "To observe intruders using session encryption, researchers needed to find a way to break the session encryption. For many organizations this has proven extremely difficult. In an attempt to circumvent session encryption rather than break it, the Honeynet Project began experimenting with using kernel-based rootkits for the purpose of capturing the data of interest from within the Honeypot's kernel. These experiments lead to the development of a tool called Sebek. This tool is a piece of code the lives entirely in kernel space and records either some or all data accessed by users on the system" [13]. Sebek uses some Rootkit's techniques to gather data from Honeypot's system calls events [5]. However, it encountered a lot of problems. In the next stage, the Honeypot researchers decided to use VMM capabilities for "out of the box" monitoring in HI Honeypots.

B. Anti-detection Issues

The real value of Honeypots is monitoring attackers' activities without being detected. Thus, from the attackers' point of view, detection of Honeypots is vital and they have developed various detection methods that most of which are highly set to particular types of Honeypots [8]. "Being able to detect Honeypots is important to malicious users as well as security professionals. The stealthy-ness of a Honeypot is an important factor to consider in an organization's overall security strategy but more importantly Honeypot developers have few tools with which to test their products" [15].

III. VIRTUAL MACHINE MONITOR (VMM)

VMs can be used to improve the security of a HI honeypot against attacks to its services. The main benefit of this approach is to monitor the honeypot from outside, thus keep the monitoring system safe from intruders. In a non-virtualized system, a single OS controls all hardware platform resources. A virtualized system includes a new layer of software, the VMM [16]. A VM environment is created by a VMM [4].

The machine the VMM runs on is called *host*, the operating system running inside a VMM is called *guest [9]*. There are two classical approaches to build VMs systems. In type I environments, the VMM is implemented between the hardware and the guest systems see Figure 1. Xen is the example of this type. On the other hand, in type II environments, the VMM is implemented as a normal process of an underlying real operating system. QEMU is the example of this type. see Figure 2 [4,9,17].

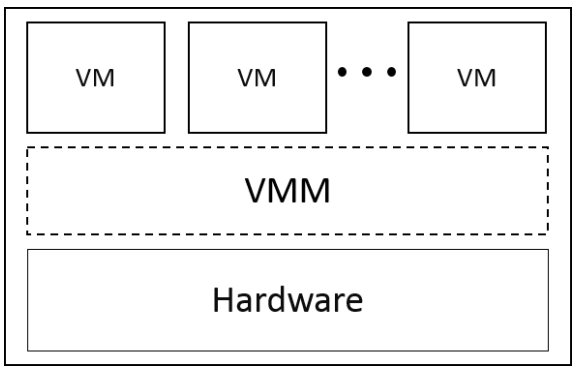

Figure 1: type I VMM

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

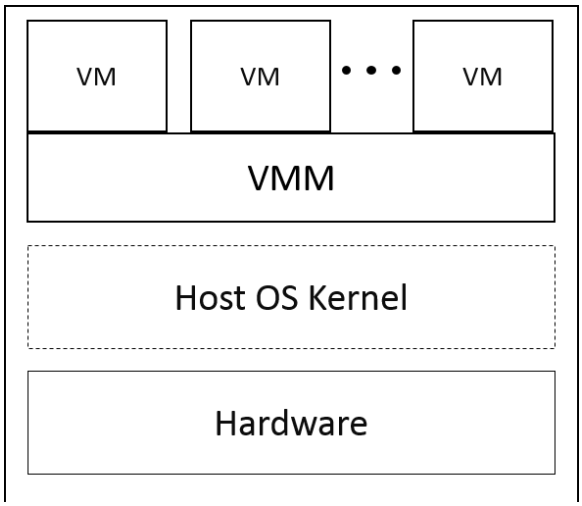

Figure 2: type II VMM

Because the VMM keeps control of the hardware itself, it has a complete view of the guest's system state, including CPU and I/O registers. With this information, a detailed observation and analysis of the guest system is possible. This method is called Virtual Machine Introspection (VMI) [4,9,18].

VMMs have three important properties which makes VMI be possible: *Isolation*: software running in a virtual machine cannot access or modify the software running in the VMM or in the other VM [16,19]. *Inspection*: VMM can access to all the state of a virtual machine: CPU state, all memory, and all I/O device state, this is the concept of Inspection. *Interposition*: VMMs need to interpose on certain VM operations. The monitoring system can leverage this functionality for its own purposes [19].

VMM designers have developed creative solutions that modify guest's source or binaries [16]. *Binary translation* and *para-virtualization* are the names of these solutions that explained in detail in the rest of this section.

A. Binary translation

If the VMM runs as software on the host machine, it is necessary to convert system calls from the guest system so that the host operating system understands them, especially if the guest and host OS differ. This process is called *binary translation* [9].

Binary translation just modifies binaries of guest's OS and no needs to modification of VM's source code [5]. A VMM that uses this solution can support legacy operating systems by making modifications directly to guest's OS binaries [5,16]. Figure 3 shows binary translation in type I VMMs. Xen uses binary translation to create VMs.

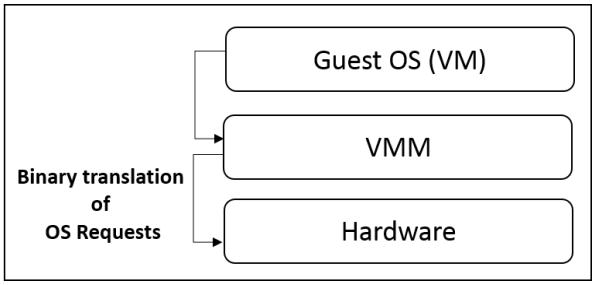

Figure 3: binary translation

B. Para-virtualization

Para-virtualization provides high performance and does not require making changes to guest's binaries [16]. In fact, in para-virtualization, the VMM modifies guest's OS source code to adapt with the host's OS [5,16]. For example, a Linux kernel have to be modified to be able to run as a guest OS in a para-virtualized environment [11]. **Error! Reference source not found.** shows para-virtualization in type II VMMs.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

IV. QEMU-BASED HONEYPOTS

In 2007, Xuxian Jiang and Xinyuan Wang in [5] Have introduced a data monitoring tool for HI honeypot called VMscope. This tool leverages QEMU's capabilities to monitor HI honeypot. After that, In 2010, the Honeynet Project have introduced a HI honeypot called Qebek [6]. In this tutorial, after expressing the problems with HI Honeypots, Qebek is described as a monitoring tool of HI Honeypot that uses QEMU for monitoring and capturing data secretly.

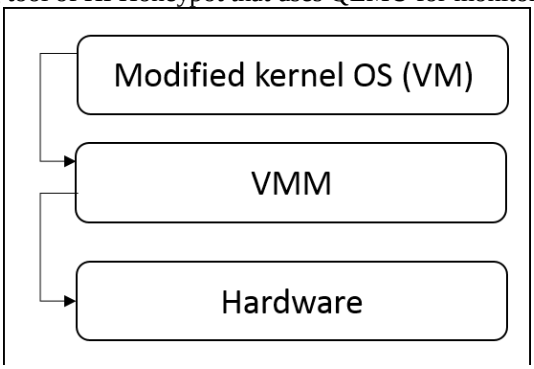

Figure 4: para-virtualization

In the rest of this section, *VMscope* and *Qebek* is explained in details.

A. VMscope

VMscope is able to provide the same deep inspection capability as Sebek, The kernel-based data capture tool presented by Honeynet Project, while still obtaining similar invisibility as other external monitoring tools.

It runs at the VMM layer and is able to observing, recording, and understanding the parameters of various VM-internal system events including system calls. Unlike the traditional approach where the monitoring tools are deployed "inside the box", the placement of VMscope is deployed outside of the monitored VMs. Such "out-of-the-box" placement is desirable because it leverages the isolation property of VMM, even if they are compromised by attackers, it will be hard, to compromise the monitoring tool outside of the VM. It also directly stores the collected log data at the host, which is outside the monitored honeypot. VMscope have implemented based on QEMU. It leverages binary translation capability of this VMM to intercept all system call instructions for logging all system call events.

Interception and *Interpretation* are two concepts that are used in data capturing. How these two process work in VMscope are explained in the rest of the section.

1) Interception

As mentioned above, VMscope tries to obtain all system call events of Honeypot VM. To achieve this, before a system call instruction is executed a call back routine invoke to collect information of this system call then after that instruction completed another call back routine is invoked to obtain returned value.

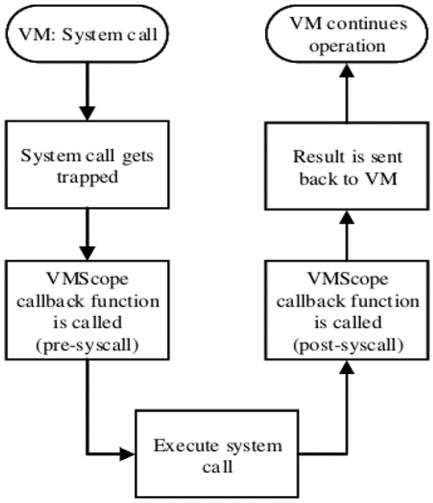

Figure 5: VMscope interception process [9]

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

2) Interpretation

After an intercepted system call completed, the returned value have stored in EAX register. Since VMscope is running on VMM layer it is able to observe content of VM registers. Therefore, to read returned value from EAX register the interpretation code is invoked.

B. Qebek

Qebek has five important components: the interception module, the breakpoint system, the System View Reconstruction (SVR) helper routines, the introspection module and the output module.

1) The interception module,

The most important reason for this module is to call the *qebek_bp_check* function when the instruction pointer of the guest system changes. Since Qebek only care about function calls, therefore, only intercept instructions that cause control flow **jump** to a new place. This is done by hook the *OPPROTO op_jmp*_T0 function in QEMU.

2) *The breakpoint system,*

Most of exist VMM-based honeypot monitoring tools intercept *sysenter* instruction to hook *syscall* made inside the guest machine. But *sysenter* is not the only way to make *syscall*. In fact, the attacker who gained root access can either install a kernel rootkit that can directly call *syscall* implementation or Register a new *syscall* and make this handler a new *syscall* dispatcher. Any of these tricks will bypass the *sysenter* hooking used by those systems. Therefore, Qebek uses breakpoint to directly hook the *syscall* implementation. Moreover, developers are also able to hook any functions they'd like to.

3) The SVR helper routines,

Helper routines are provided to perform some common tasks like read data from given virtual address. And these routines also hide the details of address translation process, which is VMM-dependent.

4) The introspection module,

This module contains the re-implementation of most functionalities of Sebek. This module is able to monitor events such as the console keystrokes, process creation and network activities.

5) The output module,

After capturing the activities inside the honeypot, Qebek could log the information into database running on the host OS, or use the management network interface to transfer it to honeywall. Further information about Qebek and its architecture can be found in [6].

V. XEN-BASED HONEYPOTS

In 2006, Nguyen Anh Quynh and Yoshiyasu Takefuji in [7] have introduced a data monitoring tool for HI honeypot called Xebek. This tool leverages Xen's capabilities to monitor HI honeypot and tries to solve Sebek's problems.

Xebek consists of three main components: The Xebek device in DomU as a data capture tool (xebekU); the logging recorder in Dom0 as a data collection (xebekd); and utilities in Dom0 including keystroke extractors, database up-loader.

1) xebekU

xebekU is a kernel code is placed in kernel-space of DomU. This code patches the system-calls such as open, close, read, write to gather the data coming in and out of the system. The collected data is then delivered to Dom0 via a shared memory between DomU and Dom0.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

2) Xebekd

xebekd is a logging recorder running in user-space of Dom0 to record data sent from xebekU. This process waits for the notifications on the new data from xebekU. If it detects that the new data arrived, it gets the data from the shared memory between Dom0 and DomU, then store the data into separate logging files for each domain respectively.

3) add-on utilities

Xebek has some utilities to extract interested data from the logging files of xebekd. Xebek tries to provide what Sebek provides with Sebek package, so it is easier for people familiar with Sebek to adopt Xebek. For the time being, a tool to extract keystrokes from logging data and another tool to upload data to a SQL server are available.

Figure 6, shows Xen architecture. Further information about Xebek and its architecture can be found in [7].

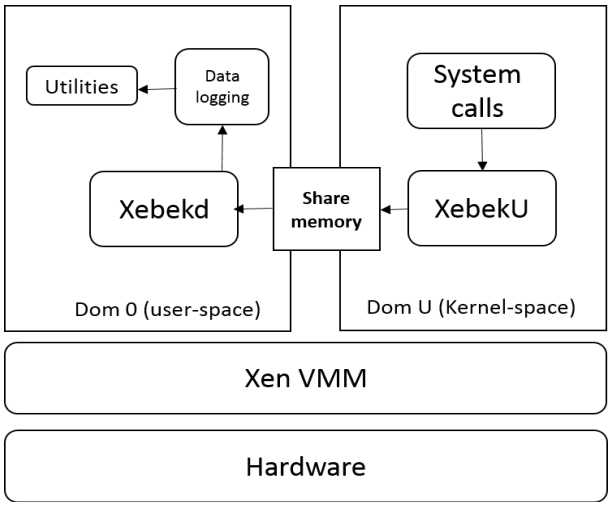

Figure 6: Xebek architecture

VI. COMPARISON

Para-virtualization is the virtualization technique that is used in Xebek and binary translation is the other virtualization technique that is used in VMscope and Qebek tools. Any of them have some features that consider in this section.

In Xebek monitoring tool, Xen is used as VMM that support para-virtualiaztion. Therefore, there are some limitation in OS choosing for Honeypot system. Xen needs to modify guest OS kernel to make it adapted with virtual environment. Unfortunately, some OSs like MS windows cannot be modified.

On the other hand, in VMscope and Qebek, QEMU is used as VMM which support binary translation. QEMU is able to support wide range of OSs because no need to modifying the source code of guest OS's kernel. These tools use binary translation to intercept system calls. This process is very useful to interception of all system calls but cause to low speed in guest OS instruction execution.
All of these features collected in Table 1.

Table 1: features of VMM-based HI Honeypots

Honeypot	Binary translation	Para-virtualization	High speed	High range of OSs
Xebek		●	●	
VMscope	●			●
Qebek	●			●

VII. CONCLUSION

In this paper three VMM-based data capture tool for HI Honeypot reviewed, we can conclude that all of them are able to solve invisibility problems of traditional approach in HI Honeypots. Because of isolation property of VMM they can isolate themselves from Honeypot system, therefore no attack to honeypot can threat the monitoring tool and its data.

REFERENCES

[1] Kuwatly, I., Sraj, M., Al Masri, Z., Artail, H.: A dynamic honeypot design for intrusion detection. In: Pervasive Services, 2004. ICPS 2004. IEEE/ACS International Conference on 2004, pp. 95-104. IEEE
[2] Mukkamala, S., Yendrapalli, K., Basnet, R., Shankarapani, M., Sung, A.: Detection of virtual environments and low interaction honeypots. In: Information Assurance and Security Workshop, 2007. IAW'07. IEEE SMC 2007, pp. 92-98. IEEE
[3] Almutairi, A., Parish, D., Phan, R.: Survey of high interaction honeypot tools: Merits and shortcomings. In: Proceedings of the 13th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting, PGNet2012. PGNet 2012
[4] Laureano, M., Maziero, C., Jamhour, E.: Intrusion detection in virtual machine environments. In: Euromicro Conference, 2004. Proceedings. 30th 2004, pp. 520-525. IEEE
[5] Jiang, X., Wang, X.: "Out-of-the-box" Monitoring of VM-based High-Interaction Honeypots. In: International Workshop on Recent Advances in Intrusion Detection 2007, pp. 198-218. Springer
[6] Song, C., Hay, B., Zhuge, J.: Know your tools: Qebek-conceal the monitoring. In: The honeynet project in Proceedings of 6th IEEE Information Assurance Workshop 2015
[7] Quynh, N.A., Takefuji, Y.: Towards an invisible honeypot monitoring system. In: Australasian Conference on Information Security and Privacy 2006, pp. 111-122. Springer
[8] Bringer, M.L., Chelmecki, C.A., Fujinoki, H.: A survey: Recent advances and future trends in honeypot research. International Journal of Computer Network and Information Security **4**(10), 63 (2012).
[9] Floeren, S.: Honeypot-architectures using VMI techniques. Network **17** (2013).
[10] Mokube, I., Adams, M.: Honeypots: concepts, approaches, and challenges. In: Proceedings of the 45th annual southeast regional conference 2007, pp. 321-326. ACM
[11] Kouba, T.: Virtual honeynet with simulated user activity. (2017).
[12] Provos, N.: Honeyd-a virtual honeypot daemon. In: 10th DFN-CERT Workshop, Hamburg, Germany 2003, p. 4
[13] Enemy, K.Y.: Sebek, A kernel based data capture tool, The Honeynet Project. In. (2003)
[14] Dornseif, M., Holz, T., Klein, C.N.: Nosebreak-attacking honeynets. In: Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC 2004, pp. 123-129. IEEE
[15] Holz, T., Raynal, F.: Detecting honeypots and other suspicious environments. In: Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC 2005, pp. 29-36. IEEE
[16] Uhlig, R., Neiger, G., Rodgers, D., Santoni, A.L., Martins, F.C., Anderson, A.V., Bennett, S.M., Kagi, A., Leung, F.H., Smith, L.: Intel virtualization technology. Computer **38**(5), 48-56 (2005).
[17] Nance, K., Bishop, M., Hay, B.: Virtual machine introspection: Observation or interference? IEEE Security & Privacy **6**(5) (2008).
[18] More, A., Tapaswi, S.: Virtual machine introspection: towards bridging the semantic gap. Journal of Cloud Computing **3**(1), 16 (2014).
[19] Garfinkel, T., Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection. In: Ndss 2003, vol. 2003, pp. 191-206