Deploying Virtual Honeypots on Virtual Machine Monitor

Wira Zanoramy Ansiry Zakaria, Siti Rohaidah Ahmad and Norazah Abd Aziz
Cyberspace Security Center, MIMOS Berhad, Technology Park Malaysia, 57000 Kuala Lumpur
{zanoramy.ansiry, rohaidah, azahaa}@mimos.my

Abstract

Xen is a virtual machine monitor that supports execution of multiple guest operating systems on the same computer hardware at the same time with unprecedented levels of performance and resource isolation. Honeypot is a decoy system, deployed throughout the network, which purposely built to attract and deceive computer intruders and malicious programs. In this paper, we review the combination of both, virtualization technologies of Xen and honeypot concept, in order to deploy and run virtual honeypot on top of virtual machine monitor.

1. Introduction

Virtualization essentially lets one computer do the job of multiple computers, by sharing the resources of a single computer across multiple environments. Virtual servers and virtual desktops let you host multiple operating systems and multiple applications locally and in remote locations, freeing you from physical and geographical limitations. In addition to energy savings and lower capital expenses due to more efficient use of your hardware resources, you get high availability of resources, better desktop management, increased security, and improved disaster recovery processes when you build a virtual infrastructure [1].

Today's powerful x86 computer hardware was originally designed to run only a single operating system and a single application, but virtualization breaks that bond, making it possible to run multiple operating systems and multiple applications on the same computer at the same time, increasing the utilization and flexibility of hardware.

Virtualization is a technology that can benefit anyone who uses a computer, from IT professionals and Mac enthusiasts to commercial businesses and government organizations. Join the millions of people around the world who use virtualization to save time, money and energy while achieving more with the computer hardware they already own [1].

This paper explores the design and architecture of Xen, a virtual machine monitor in order to develop a virtual honeypot. We discuss a bit about a virtual machines and network implementation in Xen as general. Finally, we explain the honeypot in detail and demonstrate our approach.

2. Xen Hypervisor (VMM)

The Xen® hypervisor is a unique open source technology, developed collaboratively by the Xen community and engineers at over 20 of the most innovative data center solution vendors, including AMD, Cisco, Dell, HP, IBM, Intel, Mellanox, Network Appliance, Novell, Red Hat, SGI, Sun, Unisys, Veritas, Voltaire, and of course, Citrix. Xen is licensed under the GNU General Public License (GPL2) and is available at no charge in both source and object format. Xen is, and always will be, open sourced, uniting the industry and the Xen ecosystem to speed the adoption of virtualization in the enterprise.

The Xen hypervisor was created in 2003 at the University of Cambridge Computer Laboratory in what's known as the Xen Hypervisor project led by Ian Pratt with team members Keir Fraser, Steven Hand, and Christian Limpach. This team along with Silicon Valley technology entrepreneurs Nick Gault and Simon Crosby founded XenSource which was acquired by Citrix Systems in October 2007. Xen is an x86 virtual machine monitor produced by the University of Cambridge Computer Laboratory and released under the GNU General Public License [2].

Xen is an open source virtual machine monitor (VMM) based on para- virtualization technology that allows the hardware resources of a machine to be virtualized and dynamically shared between OSs running on top of it [2].

Xen itself the hypervisor since it operates at a higher privilege level than the supervisor code of the guest operating systems that it hosts [2]. In Xen terminology, each virtual machine (VM) known is called Domain. Xen provides isolated execution for each domain, preventing failures or malicious activities

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

in one domain from impacting another domain. Multiple user domains called as DomainU in Xen terminology are created to run guest OSs [3].

Xen virtualization technology available for the Linux kernel and is designed to consolidate multiple operating systems to run on a single server, normalize hardware accessed by the operating systems, isolate misbehaving applications, and migrate running OS instances from one physical server to another [2].

2.1. Xen Architecture

In 2005, XenSource released Xen 3.0. Figure 1 shows the architecture of Xen 3.0 hosting four VMs (Host OS or Domain 0, Guest OS1, Guest OS2, and Guest OS3). The above figure depicts the basic architecture of a virtualization platform using Xen. The Xen Virtual Machine Monitor (VMM) or the hypervisor works as an idealized hardware layer; an abstraction which contains the virtualized instances of the underlying physical hardware interfaces such as the CPU, memory and I/O operations. This layer is crucial in facilitating communications between a virtual machine (VM) and the hardware itself. A host OS or the VM called Domain 0 has the privilege to the control interface of the VMM; the mechanism that enables other guest OS to be created, destroyed and managed. In short, the management and control software runs in the host OS.

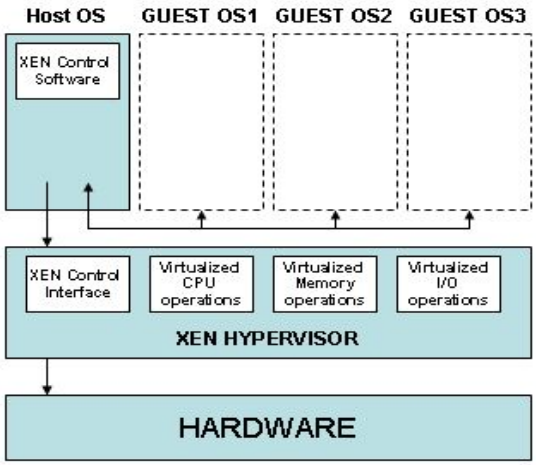

Figure 1. Xen 3.0 Architecture

2.2. Virtual Machines

A virtual machine is a tightly isolated software container that can run its own operating systems and applications as if it were a physical computer. A virtual machine behaves exactly like a physical computer and contains it own virtual (ie, software-based) CPU, RAM hard disk and network interface card (NIC) [1].

An operating system can't tell the difference between a virtual machine and a physical machine, nor can applications or other computers on a network. Even the virtual machine thinks it is a "real" computer. Nevertheless, a virtual machine is composed entirely of software and contains no hardware components whatsoever. As a result, virtual machines offer a number of distinct advantages over physical hardware.

Software running within a virtual machine is called guest software (i.e., guest operating systems and guest applications). All guest software (including the guest OS) runs in user mode; only the VMM runs in the most privileged level (kernel mode). Each guest OS to perform its own paging using its own guaranteed memory reservation and disk allocation [5].

2.3. Network in Xen

By default Xen creates a bridge to which it attaches a physical network interface. It creates a virtual interface for it and attaches the virtual interface to the bridge to keep the host or privileged system (dom0) working. There are three main setup configurations to enable network working in Xen such as bridge networking, routed networking with NAT and two-way routed network. Bridge networking which discuss in this paper is the most simplest and easiest to configure within Xen. This type of networking simply allows the VMs to use a virtual ethernet card to join the existing network. It can be used for a lot of situations.
Typically bridge networking is used where [6]:

- You can freely place a computer/device on your existing network.
- Your existing network uses DHCP or Static IP addresses.
- You want your VMs to be fully visible and available on your existing network, allowing all traffic in both directions.

In this configuration (refer Figure 2), dom0 acts as a virtual hub, forwarding traffic directly. When xend starts up, it will creates a new bridge named "xenbr0", real ethernet interface eth0 is brought down and the IP and MAC addresses of eth0 are copied to virtual network interface veth0. The real interface eth0 is renamed as peth0 and virtual interface veth0 is renamed as eth0. Then, peth0 and vif0.0 are attached to bridge xenbr0 where the bridge, peth0, eth0 and vif0.0 are brought up when a domU stats up.

When the guest operating system is up and running, its virtual network interface (vif) is bridged with the outside network through the help of Domain 0 (the host operating system of the machine). With this, the

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

guest operating system will be assigned an IP address from the subnet's DHCP server. Honeyd will run based on the preconfigured Honeyd script and it deploys virtual honeypots to all unused IP(s) within the subnet. These virtual honeypots will reply to any interaction targeted to the IP(s) that it monitors.

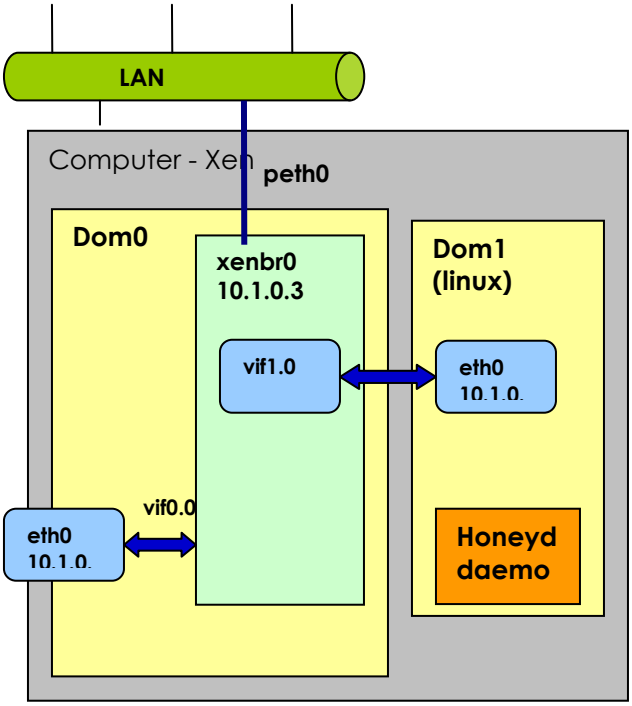

Figure 2. Bridge Networking in Xen

3. What is Honeypot?

The term 'honeypot' is usually being used for representing 'a container (or pot) of honey', where it is often playing off the image of nice sweetness that is being used as a lure. But in the case of computer security, this term is being used to represent a new born computer security concept that is solely based on deception. According to Wikipedia, a honeypot is a trap set to detect, deflect or in some manner counteract attempts at unauthorized use of information systems [4]. Honeypot is a form of digital bait to trap the attacker's tools and activities. Lance Spitzner, the founder of The Honeynet Project organization, defines a honeypots as:

"A honeypots is a security resource whose value lies in being probed, attacked, or compromised [7]."

Honeypots are resources that are meant to have no authorized activity and no production value. Theoretically, a honeypots should not see any traffic and interactions. Any interactions with it are most likely an illegitimate activity or a malicious activity [9]. Since honeypots have no production value, no resource or person should be interacting with them, therefore, any activity arriving at a honeypots is assume to be a probe, scan, or attack. Their value comes from their potential ability to capture scans, probes, attacks, and malicious activity.

The concept of honeypots is not new. Actually, it is already been around about since 16 years ago. Clifford Stoll and Bill Cheswick, whom are icons in computer security, have written good documentations related to this concept in the early 1990's. Stoll's book entitled "The Cuckoo's Egg" and Cheswick's paper "An Evening with Berferd", has already explained the basic concepts of honeypots [7]. At that time, the term 'honeypots' was not coined yet, but the idea can be clearly seen in their writings. Even though their papers are non-technical, but the contribution does gave impact to the emergence of the honeypots technology. Peoples in the honeypots community took their contribution as a launch pad for expanding the interest and developments related to honeypots technology.

Deception Toolkit or better known as DTK, is the first ever honeypots solution created. It was designed and developed by Fred Cohen and was released in 1997 [5]. DTK was constructed by a group of Perl scripts and C codes, where its main task is to emulate various of UNIX vulnerabilities and log the actions of the attacker that interacts with it. DTK simply listens for inputs, provides sensible responses that seems normal and lulls the attacker into a false sense of the host insecurity [8]. In 1998, the first commercial honeypots, CyberCop Sting, was released. At the same time, the concept of virtual systems was first introduced.

4. Honeypot on Virtualization Layer

Honeypots can be built from a physical machine or even by emulation [10]. A physical honeypots is a real machine or real host that has its own valid IP address [9, 12]. For example, a physical Gentoo Linux computer with FTP service running. A virtual honeypots is an emulated machine with its own preconfigured modeled behavior, where this behavior defines how it will respond to network traffic. Perl scripts that emulate a virtual host with emulated Sendmail service is an example of it. Virtual honeypots are very attractive because they require fewer physical computer systems and at the same time reduce maintenance costs [11]. By using virtual honeypots, it is possible to populate a network with multiple host running different types of operating

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

systems and services. One more advantage is virtual honeypots is much more easy to re-create and re-deploy after it has been compromised and exploited [12]. The idea of virtual honeypots made honeypots technology more affordable and made it easier to deploy.

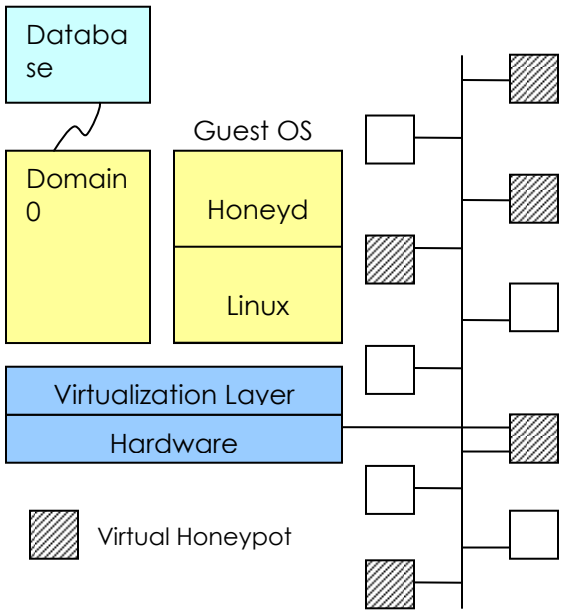

Figure 3. Honeypot on Virtualization

We designed a simple setup for deploying virtual honeypots. Instead of using a real machine to host the Honeyd software, we used a virtual machine, which is running on top of a virtualization layer. The virtual machine, or guest OS, is installed with Linux operating system. Honeyd, an open source low-interaction virtual honeypot daemon is installed after that, within the Linux. With Debian or Ubuntu Linux, we used the default Aptitude package manager to install Honeyd. The virtualization layer is built by using Xen. The physical machine is installed with Xen kernel and the kernel is selected during boot up. Once Xen is run, the virtualization layer is up and taking over the machine. The virtual machine is run after the virtual machine's machine. Even though the virtualization layer can run multiple virtual machines at the same time, this project only runs one virtual machine. One virtual machine is already enough to install and run the Honeyd daemon. Multiple virtual machines could burden the Honeypot machine and it is much harder to maintain and control.

In this setup, Honeyd is responsible in creating and deploying the virtual honeypots throughout the network that the Honeypot Machine resides. Honeyd runs based on the Honeyd script contained in the file honeyd.conf. This configuration file is important, as it defines the workability of Honeyd and the behaviors of all virtual honeypots that it emulates in the network. This low-interaction virtual honeypots emulated by Honeyd is responsible in monitoring all the unused IP(s) inside the network that it monitors. It will reply all communications that try to interact with the unused IP(s). The Honeyd software is controlled and configured from the terminal application inside Linux operating system at the virtual machine. Since Honeyd is run inside a virtual machine, it is more secure from being attack and compromised. This is because the operating system inside the virtual machine itself is a separate compartment from the software that runs below it, which is the virtual machine monitor. So, our virtual honeypot framework is farther from the reach of malicious attackers.

All logs generated by the Honeyd daemon will be routed to a database located on a remote server (refer Figure 3). These collected logs will be used for analysis purposes. This database is located outside of the Honeypot machine because we want to protect the collected data from the risk of losing it. This is because, if the Honeypot Machine is being seriously attacked, compromised or taken down, we still have our data safe and sound inside a database elsewhere.

5. Conclusion

Virtualization is a technology that has a lot of promising values for the deployment of honeypots. We can save a lot of time, energy and cost in honeypot deployment, and at the same time, maintains the original objectives of deploying honeypots. With virtualization, we could bring down the risk of running honeypot host inside our organization. Besides that, this technology also can be used by university students to test and run multiple style of honeypot deployment.

6. References

[1] Virtualization Basic, 2008 [online]. http://www.vmware.com/virtualization

[2] Abels T., Dhawan P. and Chandrasekaran B., 2005. "An Overview of Xen Virtualization".

[3] Bryan.C,Todd.D,Dow.E,Evanchik.S,Finlayson. M,Herne.J,Matthews.J.N. Xen and the Art of Repeated Research. Proceedings of the USENIX Annual Technical Conference 2004 on USENIX Annual Technical Conference: 47 – 47.2004. http://www.clarkson.edu/class/cs644/xen/files/repeated xen-usenix04.pdf.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

[4] Virtualization Basic, 2008 [online]. http://www.vmware.com/virtualization/

[5] Samuel T. King et.al, 2007. SubVirt: Implementing malware with virtual machines.

[6] Xen Networking., 2008 [online]. http://wiki.kartbuilding.net/index.php/Xen_Networking

[7] Lance Spitzner, 2002. Honeypots: Tracking Hackers

[8] Zhang, F., 2004. DTK: Deception Toolkit [online]. Available from: http://www.icst.pku.edu.cn/honeynetweb/reports/dtk.ppt

[9] Kuwatly, I., Sraj, M., Al Masri, Z. & Artail, H., 2004. A Dynamic Honeypot Design for Intrusion Detection [online]. Available from: http://webfea-lb.fea.aub.edu.lb/proceedings/2004/SRC-ECE-04.pdf

[10] Jiang, X. & Dongyan Xu, 2004. BAIT-TRAP: A Catering Honeypot Framework [online]. Available from: http://www.cs.purdue.edu/homes/jiangx/collapsar/publications/BaitTrap.pdf

[11] Niels Provos, 2004. A Virtual Honeypot Framework [online]. Available from: http://www.citi.umich.edu/u/provos/papers/honeyd.pdf

[12] Grimes, R. A., 2005. Honeypots for Windows. Berkeley : Apress.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE