Network Performance Testing on VM based Autonomous Web Server

M. Zaki Mas’ud, Asrul H. Yaacob, Nazrul M. Ahmad
FOSEE, Faculty of Information Science & Technology,
Multimedia University
Jalan Ayer Keroh Lama
75450 Melaka, Malaysia
{zaki.masud, asrulhadi.yaacob, nazrul.muhaimin}@mmu.edu.my

ABSTRACT—As online services increasingly play vital roles in modern society, the possibilities and opportunities offered are limitless, unfortunately, so too are the risks and chances of malicious intrusions. Intrusion Detection Systems (IDSs) has been widely used as an important component in protecting online service towards web attacks and evasions. Yet, today’s architectures for intrusion detection force the IDS designer to make a difficult choice to place IDS, so that it can protect itself from a direct attack. To address these challenges, this paper introduces a novel framework to safeguard IDS from a direct attack. Simply called Zero Administrative Server (ZAS), the system incorporates IDS in a Virtual Machine (VM) environment. VM offers strong isolation for IDS from the monitored services and provides significant resistance to malicious attacks. Moreover, this VM based WWW server has the ability to monitor the network traffic to the running services; analyse the information obtained and detect the intrusion; alienate the intruder from the services; and reconstruct the corrupted data or damaged files caused by the evasion. In this paper, we demonstrate ZAS by exposing it to several attacking tools as well as to show the effects it takes on the network performance in terms of TCP throughput and application-to-application round trip time.

Keywords - Intrusion Detection System, Virtual Machine, WWW Server, Checksum

1. INTRODUCTION

The World Wide Web has become an important way for enterprises to publish information, interact with potential customers and establish an e-commerce business. Thus, this scenario directly makes Intrusion Detection System (IDS) as an essential component of computer and networks defensive system. The IDS acts as an alert mechanism. It identifies threats through listening to the network traffic and looking for anomalies in the packet. IDS can be deployed either on the host which can have an excellent view on events that take place in the host, or reside anywhere in the network to view the whole network traffic. Choosing between these two configurations led to the trade off between better visibility and IDS protection.

Virtualisation provides strong isolation to the IDS from the WWW server and offers good visibility for IDS into the state of the monitored host. This solution significantly offers resistance to both intrusion and attack. Virtualisation is a technology that combines or divides computing resources to present one or many operating environments using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation, time-sharing and others [1]. There are several levels of abstraction where virtualisation can take place: instruction set level (emulator), Hardware Abstraction Layer (HAL), and operating system.

Having a well maintained server which is routinely checked by server administrator is another key factor in preventing intrusion. However, with the advancement and the complexity of security measures taken to reinforce web server from intrusion, it require skillful administrators to detect anomalies that could compromise the security of the web server [2]. Optimising the performances manually, even by skilful personal, can be time-consuming and error-prone [3]. A self-managed system is needed to reduce the work of administrator, thus avoiding error due to the complexity of the system [4].

This paper presents a novel framework of Zero Administrative Server (ZAS), a virtualisation at HAL based WWW server, to increase the trustworthiness of the online services. It proposes the combination of Intrusion Detection System (IDS) and file system integrity to detect and prevent attacks against services running on Virtual Machine (VM) and maintaining the services after the attacks.

The rest of the paper is structured as follows. Section 2 discusses the background of the technologies; Virtualisation, Intrusion Detection System (IDS) and IDS in VM. Section 3 presents the architecture of ZAS. Section 4 elaborates on the implementation and results. Finally, section 5 concludes and discusses the future directions of this work.

1.1 Motivation

This is part of the current work of the Virtual Machine Clustering Group at FIST, Multimedia University. The ability of ZAS has been tested in the virtualisation at the instruction set environment [5]. In this environment, ZAS is implemented by using an emulator, which tries to execute instruction issued by VM based WWW server by translating them into a set of native instructions and then execute them on
the available hardware. This paper presents ZAS at different level of abstraction. Part of the motivation of this study is to perform the virtualisation at HAL and then to investigate the network performance of ZAS.

2. BACKGROUND

ZAS incorporates VM and IDS into its core function to provide isolation between the networks monitoring mechanism from the web server, thus preventing it from direct attack. VM provides virtual environment for multiple operating systems to run on a single machine without interfering with each other processes [6]. This section discusses the concept of VM and IDS, as well as its architecture and the advantages of combining these two systems in creating a secure environment.

2.1 Virtual Machine

VM environment allows two or more operating systems (OSs) to run on a single machine by providing a virtual copy of the underlying machine. VM is defined as an efficient and isolated duplicate of a real machine that uses virtualisation technology [1]. It combines or divides computing resources by applying methodologies, such as hardware partitioning or emulation of a machine [2], [6]. In order to create VM environment, an abstraction mechanism is required to provide a virtual layer to run the multiple OSs. This abstraction mechanism is called the Virtual Machine Monitor (VMM) and it can be achieved in three abstraction levels namely, System Call Interface or OS level, Instruction Set Architecture (ISA) level, and Hardware Abstraction Layer (HAL) level. Even though it can be created in different levels of machine, it still partitions the lower level resources using some techniques.

OS level abstraction of virtualisation works on top of OS as a layer called system call interface. The interface can be used to manage and control the user space processor. OS level abstraction needs lower resources to execute, so it gives much better performances and a faster manipulation. However it exposes the underlying OS that host the VM. Hence, it compromises the isolation factor of other OS in the VM and can potentially affect the host OS if the guest OS is corrupted as it is shares the same kernel. Ensim [7] and Jail [8] apply this procedure in providing VM.

Virtualisation at ISA level runs at a lower performance because it has to emulate the whole instruction set in software. This include emulating the processor, memory chips, buses, hard drives and etc., making the guest OS seems running totally on a different machines. QEMU [9] and Bochs [10] are the examples of the emulator that provide virtualisation using this technique.

HAL provides a virtualisation by introducing VMM between the real machine hardware and the OS, giving a virtual hardware to the guest OS. VMM helps the VM by mapping the virtual resources to physical resources and execute privileged instruction. VMM can create multiple VMs in a single machine and each of them can run similar or different OS inside it. VMWare [11] and Virtual PC [12] are the examples of VMM using HAL virtualisation techniques.

Using HAL virtualisation in the most popular x86 machines will lower the system performance because of its architecture does not support full virtualisation. This is one of the drawback of full virtualisation in HAL because some supervisor instructions in OS must be handled by VMM for correct virtualisation or it could cause a inconvenient trap due to insufficient of privileges level of guest OS [13]. To avoid this limitation, a new approach in HAL virtualisation is introduced, called as Para-virtualisation.

2.2 Para-virtualisation

Para-virtualisation created a VM that is not an exact image of the real machine. The objective of Para-virtualisation is to improve the performance and scalability of VMM by developing a new VM interface. For instance, Xen project [13] implements several new techniques to improve efficiency and performance of VMM such as, new lightweight event mechanism to replace traditional interrupt and asynchronous I/O ring for data transfer between VM and VMM.

Xen operations have a higher privilege level than the supervisor code of the guest OS system than its host. Thus, preventing the instruction from guest OS from being silently fail during execution by VMM. Each guest OS in Xen perform its own paging for support performance isolation and it can support up to 100 instances OSs. In Xen, the host that run on the real machine is called Domain0 and it is responsible for hosting the other guest OS. Domain0 is created during boot time and can create, terminate and control the resources of other domains that encapsulates guest OS.

Although it does require modifications to the guest OS, there is no modification required on the Application Binary Interface (ABI). Hence, similar application that resides in the guest OS can still be used. As the VM created by Para-virtualisation achieve better performance and still kept the traditional VM features, it can be an essential component in our autonomous WWW server system. Besides providing good system performance, it can also keep the protection mechanism out of intruders reach as it isolates the target system and the security components in the host OS.
2.3 Intrusion Detection System

Intrusion Detection is defined as the process of monitoring the events in a computer network and analyzing any attempt to bypass the computer security mechanism on a network in order to jeopardize the integrity of data stored in a computer. Intrusions are caused by an unauthorized user who wants to access a network system, or by an authorized user who misuse the privileges given to him. Intrusion Detection System (IDS) is a software or hardware that automates the monitoring and analyzing process. Nowadays, IDS has been considered as an important element in protecting a network system. Although firewall is widely used as a barrier mechanism to prevent entry to some kind of network traffic, it alone cannot withstand an attack to the system. Therefore, IDS is used to reinforce firewall by monitoring activities that take place within the network.

Intrusion detection falls into two basic categories: signature based intrusion detection and anomaly detection. Attacker normally will leave behind some clues and has a certain signature, which can be detected using specific software. This signature and clues can be found inside the data packet sent over the internet or private network. This signature is then compared with known signatures define by the IDS. Whenever the packet with anomalies is found, IDS will log suspicious activity and generate alert.

IDS can be categorized based on how they gather the source information of the intrusion. Generally, it is classified into 3 types [14] such as Network based IDS, Host based IDS and Distributed based IDS.

Network based IDS (NIDS) is a type of intrusion detection system that detect invaders by monitoring the activities over the network. It is done by capturing and analyzing network packets. NIDS is placed on strategic network segment where it monitors all network traffic for each device in the network segment. NIDS usually runs in stealth mode in order to hide from intruders. NIDS listens to any packets in the network and compares the captured packets with defined signatures stored in the database or log. Normality deviations are then logged and signaled as threats.

Host based IDS (HIDS) operates in individual host device. This gives HIDS more advantage when compared to NIDS: HIDS monitors the activities happening in the host in real time with more reliable and accurate log. Furthermore, HIDS can closely observe the effects on the data files and system processes soon after intrusion attempts.

Distributed based IDS (DIDS) is a combination of several NIDS placed on several network segments and connected to a centralized management station. Each NIDS monitors its respective segment and forward the log to a centralized station. DIDS is more complex to implement due to it centralized station and can be costly.

2.4 Intrusion Detection System in Virtual Machine

The ideal choice for deploying an IDS is to have great visibility of the host system like HIDS, but still providing strong isolation for the IDS as done in NIDS. To realise the idea, we consider the work proposed by Garfinkel and Rosenblum [15], using the concept of Virtual Machine (VM). The idea is to encapsulate the system to be monitored inside a VM as a guest system, which is closely monitored from outside (host system). Hence, the approach isolates the IDS from the VM, giving it a high degree of attack resistance and allowing it to continue observing and reporting with integrity even if the VM has been corrupted. Even though the IDS is placed outside the guest system, it is still keeping the same visibility as the HIDS and it has the privilege to observe the local activities in host and guest system.

3. ZAS ARCHITECTURE

Our main focus in this work is developing an autonomous WWW server that can (1) isolates security tools from the target system, (2) detects intruder, (3) blocks potential attacker from using the service (4) checks any corrupted files or data and (5) reconstructs any modified files or data without any human interventions. These objectives are realisable by combining several existing systems namely VM, IDS, Checksum and together with a new developed script-controlled daemon called Zero Administrative Server (ZAS) as the controller of the system. The architecture of ZAS is depicted in Fig 1.

We used Para-virtualisation technique to create VMs. VM is responsible to isolate IDS, checksum and ZAS daemon from the WWW server. Hence, it keeps the protection mechanism away from the target system and prevents it from being exploited by the intruder. The IDS is a network monitoring tool used to view the incoming data packet that goes into the machine. It also acts as a triggering mechanism to warn the whole system that an intrusion has occurred and for the other component to start taking actions.

The alert generated by IDS contains the date and time of the attack, IP address of the attacker, alert priority generated by the activities, and classification of attack. This information is then updated into an attacker log and then analysed by ZAS. Based on the respective priority level, any attacker’s IP address having number of attempts made exceed the predetermined threshold is blocked temporarily from accessing the services offered by the WWW server. Table I shows the priority level and the risk it bring to the system. Once blocked, the information of the attackers is then deleted from the log to avoid the log file from getting bigger as well as to reduce the log processing time.
Fig. 1: Autonomous WWW server architecture.

The next process is the execution of data integrity checking by checksum. Checksumming in the server helps to detect whether important file have been tampered with. Any changes made by any malicious programs installed illegally or unauthorized modification to the file without the knowledge of file system is checked by checksum soon after the detection of suspicious attack by ZAS daemon. The checksum process is executed only towards the files system of the guest operating system as well as the important data files used by the WWW server and not to the whole files.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>Prioritisation of the IDS Alert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Attack Risk Level</td>
</tr>
<tr>
<td>Priority 1</td>
<td>High</td>
</tr>
<tr>
<td>Priority 2</td>
<td>Medium</td>
</tr>
<tr>
<td>Priority 3</td>
<td>Low</td>
</tr>
</tbody>
</table>

The checksum result is then compared with the original checksum value stored in the ZAS daemon log. If differences are detected, it shows that the files have been tampered and ZAS then starts implementing its reconstruction mechanism. Finally, after certain duration of time, the blocked IP address is removed from the list, and the user can access the web services again. However if they attempt to attack the WWW server again, ZAS is ready to repeat all these processes.

4. IMPLEMENTATION AND RESULT

This section discusses the implementation of ZAS prototype and provides discussion on the results gained from the testing. The tests focus on impact on network performances caused by VM.

A prototype was implemented using standard PC having Pentium 4, 3GHz with 512MBytes of RAM. In Domain0, XenoLinux with Fedora Core 3 base system is used. A total of 128MBytes of RAM is reserved for Domain0. On the guest side, Slackware Linux was deployed. The client is a standard PC having AMD Athlon XP 1.2GHz with 256 MBytes RAM. The client is connected to server using a 3Com 100Mbps switch.

To measure the effectiveness of ZAS towards intrusion, the web server was exposed to several web attack tools namely, Brutus, AET2, Nmap, Smurf2k and Angry IP scanner. Each of the attacks is successfully detected by ZAS and the attackers IP address was blocked. This shows that ZAS is able to detect anomalies in the network and blocked any attempts of intrusion.

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>Bulk Data Transfer Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Size (bytes)</td>
<td>Transfer Rate (Mbps)</td>
</tr>
<tr>
<td>plain</td>
<td>VM</td>
</tr>
<tr>
<td>256</td>
<td>88.81</td>
</tr>
<tr>
<td>512</td>
<td>88.80</td>
</tr>
<tr>
<td>1024</td>
<td>88.82</td>
</tr>
<tr>
<td>2048</td>
<td>88.85</td>
</tr>
<tr>
<td>4096</td>
<td>88.25</td>
</tr>
<tr>
<td>8192</td>
<td>88.06</td>
</tr>
<tr>
<td>16384</td>
<td>88.07</td>
</tr>
<tr>
<td>32768</td>
<td>88.06</td>
</tr>
<tr>
<td>65536</td>
<td>88.04</td>
</tr>
</tbody>
</table>

The deployment of web server in the VM environment incurs some performance or processing overhead including CPU overhead, I/O overhead, etc. In this paper, we measure the processing overhead introduced by VM and then discuss the effect of the overhead on the network performance of the web server. We use Netperf[16] network performance tool to measure the overhead. Netperf is designed around the basic client-server model. The tool consists of two executables – netperf, which represents a client process and netserver, which represents a server process. The options for traffic patterns are set on the system running netperf, while the netserver is invoked on the server system.
Two tests have been conducted: bulk data transfer and request-respond performance. The first test is an indication on overhead for transferring data from client to server as in uploading files during FTP session. The second test gives an indication on overhead for short-live data transfer as in HTTP session.

As shown in Table II, for the same workload the network throughput of the web server running inside a VM is comparable with the network throughput of the web server running directly on a physical machine. We notice a decrement of less than 2% in bulk transfer performance. Besides that, Table III depicts the request-response performance tests. The overhead varies depending on the size of the messages being transferred. When the data size increases, as in real WWW transactions, the overhead continues to decrease. However, the deployment of web server in VM degrades the request-response per sec performance as compared to the physical based web server. To exemplify this scenario, we evaluate the TCP application-to-application round trip time for both machines. For request-response size of 1 byte, a performance of 10814.40 request-response per sec has been observed for physical based web server, corresponding to an application-to-application round trip time of $\frac{1}{10814.40} = 0.092\text{ms}$ . On the other hand, round trip time of $\frac{1}{8069.73} = 0.124\text{ms}$ has been recorded for VM based web server. For a more realistic request size (128 bytes) and response size (8192 bytes), the round trip time for physical and VM based web server are 1.002 ms and 1.013 ms respectively. Therefore, it can be deduced that the WWW transactions over VM based web server experiences a slightly latency than the physical based web server.

As conclusions, even though a decrease in network performance was noted, such decrement is fully justified. In a physical environment, CPU utilisation plays a significant role in reaching acceptable network performance such as throughput and application-to-application round trip time. To process higher level of throughput, more CPU resources are needed. The effect of CPU resource availability on network performance of VM based web server is even more significant. This is because all the elements of the networking from physical to the applications layers are virtualised including the network connection between the guest and host systems. Furthermore, the ZAS system utilises some of the resources to protect the web server. Therefore, processing of WWW transactions is somewhat more expensive for VM based web server than running web server directly on the physical platform.

5. CONCLUSION

This paper proposes the idea of Zero Administrative Server (ZAS), a Para-virtualisation based WWW Server that provides an autonomous attack detection and data recovery system. ZAS is a script-controlled daemon and mainly consists of Intrusion Detection System (IDS) and file integrity checking. The IDS is used to detect any malicious attacks from the intruders. The IDS is kept isolated from the WWW Server via VM. Therefore, it is inaccessible and cannot be subverted by intruders; offers high attack resistance to IDS itself; and provides great visibility to the monitored system. File system integrity checking is deployed by ZAS to detect any attacker’s signatures left in any files if they successfully break-in into the system. Once the corrupted data is identified, the ZAS automatically recovers any damaged files by transferring the original contents stored in the host system.

The current IDS used in this research is a rule-based approach. The rule-based approach relies on the set of predefined rules that has been legislated against the intruders and stored inside the rule database. These rules must be updated regularly to give a good detection for the IDS. In addition, human intervention is needed to do the updating task. Therefore, we believe a more promising approach can be done to enhance the IDS by making it more intelligent. It means that the IDS can learn a new pattern of attacker without any interventions from human especially administrator.
This will help to reduce the workload for the administrator and increase the performance of the IDS. The IDS also has limitations in detecting the intruders. Sometimes, IDS cannot distinguish between the attacker and the authorized user. There is a possibility in the ZAS architecture, the daemon cannot detect the intruders if they are successfully bypassing the IDS. Therefore, a forensic technique can be introduced for further research. A forensic agent will act like a patrol to monitor all the traffics and activities inside the system. This eventually can help to improve the security mechanism for existing systems.

REFERENCES
