UNIVERSITI TEKNIKAL MALAYSIA MELAKA

VIRTUAL MACHINE BASED AUTONOMOUS WEB SERVER

MOHD ZAKI MAS'UD
FAIZAL M.A.
ASRUL HADI Y.
AHMAD, N.M.
ERMAN H.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

VIRTUAL MACHINE BASED AUTONOMOUS WEB SERVER

MOHD ZAKI MAS'UD
FAIZAL M.A.
ASRUL HADI Y.
AHMAD, N.M.
ERMAN H.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Virtual Machine based Autonomous Web Server

Mohd Zaki Mas'ud.[1], Faizal M. A.[2], Asrul Hadi Y.[3], Ahmad, N.M.[4], Erman H.[5]

[125]Faculty of Information and Communication Technology
Univeristi Teknikal Malaysia,
Karung Berkunci 1200,
75450 Ayer Keroh, Melaka.

[34]Faculty of Information and Science Technology
Multimedia University
Jalan Ayer Keroh Lama
75450 Ayer Keroh, Melaka

{[1] zaki.masud, [2]faizalabdollah, [5]erman}@utem.edu.my, {[3]asrulhadi.yaacob,[4]nazrul.muhaimin}@mmu.edu.my

Enterprises are turning to Internet technology to circulate information, interact with potential customers and establish an e-commerce business presence. These activities are depending highly on Web server and maintaining good server security has been a requirement for avoiding any malicious attacks especially web defacements and malware. Web server administrators should be alert and attentive to the status of the server at all time. They need to be persistent in monitoring the server in order to detect any attempted attacks. This is an advantage for a web server that is maintained by a big company that has a big budget to hire a knowledgeable web server administrator, for a new established small company it will only burden their expenses. To overcome this problem, this paper proposes a low cost system called Autonomous Web Server Administrator (AWSA) that is fully developed using open source software. AWSA combines several computing concepts such as Virtual Machine, Intrusion Detection System and Checksum. AWSA offers a Virtual Machine based Web server that has the ability to automatically detect intrusions and reconstruct corrupted data or the file system without any human intervention.

*Keywords-**Intrusion detection system, Virtual machine, checksum, web server***

I. INTRODUCTION

Today, the current explosion in Internet trading may have eventually contributed to cast doubt on the trustworthiness of the services. Trust is at the heart of each and every transaction of online businesses. To establish the trust, the company not only needs to persistently declare the infrastructure is secure but also needs to have the tools to let the customers establish trust by themselves.

As online business sites become attractive targets for the customers, Phishing scams, identity fraud and website spoofing have become some of the most lucrative business models of today's cyber-criminals. According to Malaysian Computer Emergency Response Team (MyCERT) the total number of DDOS and intrusion incidents report are increasing from 60 reports in September 2000 to 1661 reports in September 2010 [1]. Nowadays, anyone can attack Internet sites using readily available intrusion tools and exploit scripts that capitalise on widely known vulnerabilities.

Despite stiff challenges from attackers, the influenced of World Wide Web to broaden market potential across geographical boundaries has led major companies or even Small Medium Enterprises (SMEs) to seriously expand their operations and marketing strategies over the Internet. However, a major concern of these companies is to put aside a huge investment on the security system in order to build trustworthiness among customers and to protect its image.

For that reason this paper presents a novel framework of Autonoumous Web Server Administrator (AWSA), a Virtual Machine (VM) based WWW Server, to enhance the trustworthiness of online businesses. It proposes the combination of an Intrusion Detection System (IDS) and file system integrity to detect and prevent attacks against services running on a VM and maintaining the services after the attacks. In summary, AWSA offers the following advantages: (1) the ultimate solution for SMEs to get involved in online business. All AWSA components utilize open source applications which are relatively low cost, scriptable and modifiable, (2) to maintain the services running on a VM from outside (host system), thus keeping the IDS safe, out of reach from intruders, (3) to detect any malicious data modification that the intruders have made through file system integrity checking, and (4) maximizing the state of the VM which can be saved, cloned, encrypted, moved or restored. It allows AWSA to re-construct the corrupted disks or files on the VM by transferring duplicated data from the host system.

The rest of the paper is structured as follows. Section 2 describes the Virtual Machine (VM) concept. Section 3

978-1-4244-8581-9/11/$26.00 ©2011 IEEE

presents the architecture of a self-protected AWSA. Section 4 explains the details of our implementation of AWSA and discusses the prototype performance. Section 5 concludes the paper with a brief summary.

II. VIRTUAL MACHINE

A. Virtual machine environment

A Virtual machine (VM) is defined as an efficient and isolated duplicate of a real machine [2]. This environment is created by using a Virtual Machine Monitor (VMM) which provides a second layer on a machine for another operating system to run on. VMM reproduces everything from the CPU instructions to the I/O devices in software of the operating system it runs on. Virtualization in a VM involves mapping of virtual resources, for example, the registers and memory, to real hardware resources. It also uses the host machines instructions to carry out the actions specified by the VMM. This is done by emulating the host Instruction Set Architecture (ISA).

The underlying operating system is called the host operating system [3]. VMM runs on top of the host operating system. Hence it can hold a second or more operating system on it. The operating system which runs on VMM is called the guest operating system. Fig. 1 illustrates the virtualization concept of VM. The host operating system and the guest operating system can be either the same or different type of operating system. For instance, Windows can run as the host operating system and Linux as the guest operating system.

VM and virtualization research have been going on for nearly thirty years; IBM VM/370 was among the first VMM applications introduced. VMWare[4], Virtualbox[5], and Xen[6] are some examples of more recent VMMs that have been developed. Another close environment which also provides virtualization is called an emulator. The difference is that the emulator only let one guest operating system to run on the host system. An example of an emulator is Qemu [7], which is used in AWSA development.

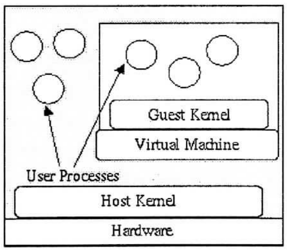

Fig. 1: Virtual machine environment.

B. Advantages of virtual machine

VM technology is widely used to run operating systems or software developed specifically for one platform on another platform. For AWSA, this can be manipulated to increase a server security, others are exploited as below:

Compatibility: VMM can run any operating system on top of it without any modification being.

Isolation: VMM provides a complete layer of virtual hardware to the guest operating system, the guest operating system communicates directly with the hardware without going through the host operating system, thus providing a strong isolation between the host and guest operating system. This indeed will protect the host operating system and any host application from being manipulated by an intrusion on guest operating system.

Security: VMM is written in a few lines [7], which make it a simple program. Due to its small code size, VMM provides a better trusted computing base than the guest operating system itself. This narrow interface restricts intruder actions from invading the host operating system.

Inspection: VMM can have access to any operations in the virtual machine; CPU instructions, all memory actions, I/O device and I/O controllers operations. This feature will let the host operating system to view any operations taking place in the guest operating system via VMM and provide a mechanism for monitoring intruder activities.

C. Related Work

The main objective of this paper is to propose a system that can sustain intruder invasion on a server and fix any corrupted data in the server without any intervention from an administrator. Similar work has been done in *ReVirt*[8] and *Terra*[9]. These two researches manipulate VM as a monitoring mechanism for any intrusion that might occur on the system this is done by modifying some of the code in VM. Hence make it different to the work done in this paper. AWSA only take the advantage of Isolation features in VM without making any changes to the code.

AWSA have a similar concept to *Livewire* [10], where IDS is used to monitor VM from it host operating system. Using this method IDS is place outside the VM which will protect the IDS itself by making it out of reach from intruders attack. The difference that AWSA has over these two is the next actions that take place after the intrusion occur. In their work, IDS is use to only monitor and reporting the incident in a log file or database and the next action will have to depend on the administrator's responsibility for the server, in contrast attack analysis and necessary recovery action take is done automatically in AWSA.

VM provides isolation between the host operating system and guest operating system. Due to its small and narrow code of VMM and also the power of VMM to monitor the process of guest operating system, VM gives an advantage for AWSA and it component (IDS and checksum) to be invisible from the intruder attack. On the other hand, the monitoring and report generated by IDS will alert AWSA to take further action on the intrusion made to the server where cheksum will act as the integrity checker if any of the data is corrupted.

In general AWSA have the ability of isolating potential target with it defense system, monitoring of intruder as well as reporting their activities, data integrity checker and finally reconstruction of compromised data. By considering all this as the features of AWSA, we can compare the difference and similarities of AWSA with the other related work. The comparison of AWSA with *ReVirt*, *Terra* and *Livewire* is summarized in table I.

Table I
The Difference and Similarities of AWSA with Other Related Works

System / Features	AWSA	ReVirt	Terra	Livewire
Monitoring and reporting	✓	✓	✓	✓
Isolation	✓	×	×	✓
Data integrity Checker	✓	×	×	×
Data Reconstruction	✓	×	×	×

In conclusion, AWSA is developed by improving the works that have been done by researchers and developers of the system mentioned in this chapter. In general AWSA is only putting the bit and pieces provided by each element mentioned above into a complete puzzle that reinforces the security of a server.

III. AWSA ARCHITECTURE

A. *AWSA framework*

As online business becomes an attractive target to cyber-criminals, self-protection against attacks is becoming an indispensable attribute of web servers. To achieve self-protection, a web server has to monitor itself, analyze the information obtained, plan a defense against detected attacks, and execute the plan. Therefore, in this work, we adapt these characteristics to design the AWSA. It is a combination of 6 main components: (a)Isolation, (b)Network Monitoring, (c)Logger (d)Analyzer (e)Integrity Checker and (f)Data reconstruction, as shown in Fig 2.

The purpose of isolation in AWSA is to keep the rest of the protection mechanism away from the web server, making them invisible to the attacker's eyes. Thus, it protects the other functions from being manipulated by the attackers. To facilitate this component, a VM is used to encapsulate the web server away from the host operating system. Meanwhile, network monitoring is used to view the incoming data packets that go into the web server. It also acts as a triggering mechanism to warn the other components to start taking action if it detects suspicious data packet going through the network traffic. Intrusion Detection System (IDS) [10] [11] is chosen to equip AWSA as network monitoring tool.

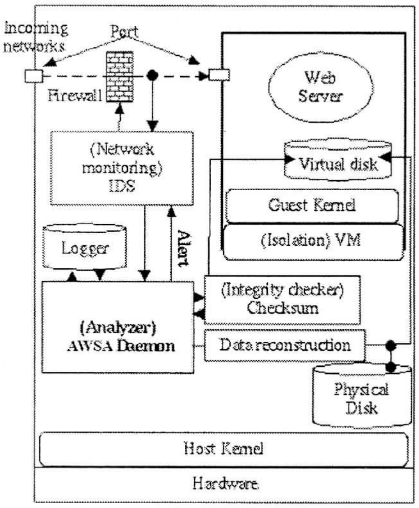

Fig. 2: The system architecture

Logger is the next process executed in AWSA to respond to the alert issued by the previous function. Logger is a database that keeps important information such as date and time of the attack, IP address of the attacker, alert priority generated by the activities, and classification of attack that are captured from the suspicious data packets. The gathered information is used by the analyzer to determine if the data packet is used for intrusion or not, if it is found to be a data packet from an attacker then it will be blocked.

The analyzer also has the ability to compare the original value of the checksum with the generated value from data the integrity function. The integrity checker is able to perform checksum once an intrusion is detected by the analyzer. AWSA's next function is to perform a data reconstruction if the comparison of data integrity by the analyzer found that the data stored is compromised.

B. *AWSA daemon*

Our main focus in this work is to develop the analyzer or simply called AWSA daemon. It is considered as the heart of AWSA and controls the flow of the system. The processes done by AWSA daemon is depicted in Fig 3.

Upon receiving a triggering signal from IDS, AWSA daemon updates the alert into the database. The alert generated by IDS is based on the risk level caused by an attack. Each risk has it own priority. Table I shows the risk.

Once the database updated, AWSA daemon analyzes the data to look for a potential attack, this is done by counting the attack that is originated from similar attackers and similar risk level. The result obtained is then compared with a predetermined threshold value. The threshold is used as an indicator to catch any suspicious attacker with reasonable doubt and at the same time, minimizing the false alarms, which can burden any security system, especially the log file.

Table I
Prioritization of the IDS alert

Priority	Attack Risk Level
Priority 1	High
Priority 2	Medium
Priority 3	Low

Once the comparison value is over the threshold, the suspicious attacker alert is erased from the attacker's log and the attacker's IP address is temporarily blocked from accessing the services offered by the web server. This will prevent a continuous attack from the same potential intruder. It also prevents the attacker log file from getting bigger as well as to reduce the log processing time.

The next process is the execution of the data integrity checker by checksum. Checksumming in the server helps to detect whether important files 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 the AWSA daemon. The checksum process is executed only on the files system of the guest operating system as well as the important data files used by the Web server and not on all the files.

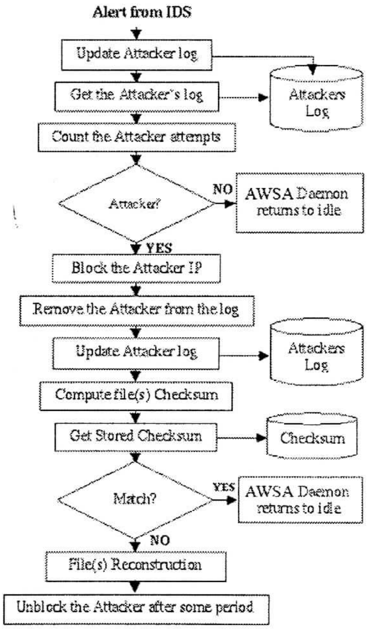

Fig. 3: Flowchart for AWSA daemon

The checksum result is then compared with the original checksum value stored in the AWSA daemon log. If differences are detected, it shows that the files have been tampered with and the next process is activated. When there is a difference between the compared values, AWSA starts implementing its reconstruction mechanism. This is the advantage of utilizing AWSA; it provides automatic recovery of the corrupted files.. Therefore, the file system on the VM can operate without any interruption to the web services and intervention from the web server administrator.

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 web server again, AWSA is ready to repeat all these processes.

IV. IMPLEMENTATION AND RESULT

A prototype was implemented on a Linux platform, using QEMU for virtualization. *Linux Fedora core 3* and *Slackware Linux* were deployed as host and guest operating system respectively. The entire server was being monitored by SNORT 2.0.2 [12], a public-domain IDS tool. Swatch [13], the active log file monitoring tool was used to alert the AWSA. The verification of the data integrity was done by utilizing MD5 checksum. The hardware used in the experiments was a standard PC system (AMD Athlon XP 1600 CPU, 256 MBytes RAM).

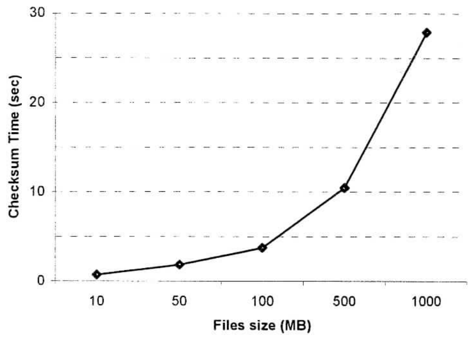

Fig. 4: Integrity checking time

The performance of AWSA is measured by two aspect; the durability in facing continuous web attacks and the response times taken to (a) detect and block attacker's IP address, (b) verify the integrity of data, and (c) reconstruct the compromised data. The durability tests were done by exposing the server to several web attack tools, namely Adore [14][15], Brutus AET2[16], Nmap, Smurf2k and Angry IP scanner. Each of the attacks is successfully detected by AWSA and the attackers IP address was blocked.

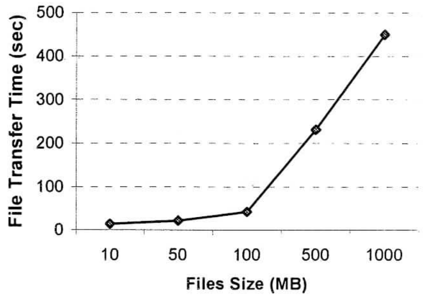

Fig. 5: File recovery time

On average, it took 86 sec for AWSA to impose the access restriction. This Average Blocking Time is taken starting from the time AWSA daemon detects the numbers of attacker's attempts exceed the predetermined threshold until the AWSA daemon alienates the attacker. Fig. 4 illustrates that AWSA daemon took less than 30 seconds to perform the integrity test on 1 Gigabyte of data and the file system of the server. The obtained result shows the time taken by AWSA to perform a checksum on several files with various sizes. It is a good indicator for estimating the period of performing a checksum on the particular files sizes. The final evaluation of AWSA is the file recovery time which indicates the time taken by AWSA to reconstruct the compromised data to its original state. Fig. 5, clearly depicts that files with larger sizes would consume more time in transferring the contents to the guest operating system.

V. CONCLUSION

This paper proposes the idea of Autonomous Web Server Administrator (AWSA), a VM-based Web Server that provides an autonomous attack detection and data recovery system. AWSA 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. IDS is kept isolated from the Web Server, therefore, it is inaccessible to VM processes 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 AWSA 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 AWSA automatically recovers any damaged files by transferring the original contents stored in the host system.

In the near future this work will be extended in finding a better IDS architecture available to be incorporated with AWSA and to extend a research on providing the integrity checker for the virtual machine itself. This research work is also towards finding a process of providing a forensic mechanism in order to investigate how an attack is occurs so that countermeasures can be taken for future attack.

ACKNOWLEDGMENT

This research has been supported by University Technical Malaysia Melaka under the short grant project.

REFERENCES

[1] CyberSecurity Malaysia. (September, 2010). MyCERT Incident Statistics. [On-line]. Available at http://www.mycert.org.my/en/services/statistic/mycert/2010/main/detail/725/index.html

[2] Popek, G. & Goldberg, R *"Formal Requirements for Virtualizable Third Generation Architectures"*, Communications of the ACM, 1974, 17(7):pp. 412-421.

[3] Dunlap, G. W., King, S. T., Cinar, S., Basrai, M. A. & Chen, P. M.. *"ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay."* Proceeding of 2002 Symposium on Operating System Design and Implementation, 2002.

[4] VMware Inc. 2009. (2009, April 05). VMware Workstation. [Online]. Available : http://www.vmware.com/

[5] Sun Microsystem., 2009. (2009, April 05). Virtualbox. [Online]. Available: http://www.virtualbox.org/

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," Proceedings of the nineteenth ACM symposium on Operating systems principles. pp. 164–177, 2003.

[7] Bellard, F. 2009. (2009, January 1). QEMU CPU Emulator. [On-line]. Available: http://fabrice.bellard.free.fr/qemu/

[8] Garfinkel, T., Paff, B., Chow, J., Rosemblum, M. & Boneh, D. 2003. *"TERRA: A virtual Machine-Based Platform Trusted Computing."* SOSP 2003, 2003, pp. 193-205.

[9] Dunlap, G. W., King, S. T., Cinar, S., Basrai, M. A. & Chen, P. M.. *"ReVirt: Enabling Intrusion Analysis Through Virtual-Machine Logging and Replay"*, Proceeding of 2002 Symposium on Operating System Design and Implementation, 2002.

[10] Garfinkel, T. & Rosenblum, M. A, *"Virtual Machine Introspection Based Architecture for Intrusion Detection"*, Proc. Network and Distributed System Security Symposium. 2003

[11] Laureano, M., Maziero, C. & Jamhour, E., *"Intrusion Detection in Virtual Machine Environments,"* Proc. EUROMICRO Conference, 30, 2000.

[12] Sourcefire Inc. (2009, October 1). Snort – The Open Source Network Intrusion Detection System. [On-line]. Available: http://www.snort.org.

[13] Sourceforge. (2009, October 1). Swatch. [On-line]. Available: http://swatch.sourceforge.net.

[14] Wichman, R. (2009, November 21). Linux Kernel Rootkits. [On-line]. Available: http://la-samhna.de/library/rootkits/index.html

[15] (2009, November 21). Packet Storm. Retrieved. [On-line]. Available: http://packetstormsecurity.org.pk/UNIX/penetration/rootkits/index.html

[16] Hoobie Inc. (2009, October 10). Brutus - The remote password cracker. [On-line]. Available: from: http://www.hoobie.net/brutus/