Journal of American Science

Websites:
http://www.jofamericanscience.org
http://www.sciencepub.net

Emails:
editor@sciencepub.net
sciencepub@gmail.com

MARSLAND PRESS
Multidisciplinary Academic Journal Publisher

Another Approach in Building a Secure Server OS, Based on Using Virtualization

Seyyed Mohsen Seyyedsalehi [1], Shahriar Mohammadi [1], Seyyedeh Fatemeh Malek [2]

[1.] K.N.Toosi University of Technology,Tehran, Iran
[2.] Amirkabir University of Technology,Tehran, Iran
seyyedsalehi@gmail.com

Abstract: In this paper the issue of building a secure operating system, as the most important system software, is discussed. Securing operating systems have mainly two traditional approaches: First is reviewing and making secure configurations, and the second is disabling unnecessary services. In this research, these two approaches are reviewed, then the third approach will be presented. In this new approach the important system services will run in isolated virtual machines. Separated services greatly reduce the risk of attack and increase security of operating system. One of the most important benefits of our new approach is the minimal cost of implementation by using the recently available virtualization technologies.
[Seyyedsalehi SM, Mohammadi S, Malek SF. **Another Approach in Building a Secure Server OS, Based on Using Virtualization.** *J Am Sci* 2023;19(6):35-43]. ISSN 1545-1003 (print); ISSN 2375-7264 (online). http://www.jofamericanscience.org 06.doi:10.7537/marsjas190623.06.

Keywords: Secure operating system; virtualization; linux; performance

1. Introduction

Security is important when you have valuable assets that are mentioned to be protected from any damage. In safe operating systems related problems and solutions first you must specify which assets are valuable, and how much they are sensitive. Our valuable properties in this concept are usually made of data and information. In almost all information systems our important assets are placed on OS. As shown in table 1, it can briefly be said:

Table 1. Security of OS & applications

Result	***status***
Insecure system	Insecure operating system + insecure applications
Insecure system	Insecure operating system + secure applications
relatively secure system	secure operating system + insecure applications
secure system	**secure operating system + secure applications**

Looking at a higher level if the operating system wants to provide security, it should have the following controls:

- Control of who can run a program.
- Control of which libraries can called by a program.
- Control of which code into a library program is executed.

And also,

- Control of which data can changed by a program.

Providing above controls in detail isn't possible because of the extent of the issue, but there are useful strategies that will be surveyed. After that, a new idea for more secure operating system is offered.

2. The Problem

Operating system security criteria, as a common practical standard, hasn't been documented yet. Only an effort called *IEEE Baseline Operating Systems Security* ™ in this context was found. The standard had begun to determine the characteristics of a safe operating system, but the project already suspended in IEEE and it's last update has been in 2006. Currently, creating and maintaining a secure operating system by using available technologies is the main security concern of many countries. Doing it from scratch isn't economic therefore there is no choice except Underlying existing products. In this regard, there are two choices among available technologies:

1. The commercial operating systems as the base of secure OS: Due to lack of access to code and functions, we can't work deeply on securing of these OS's. On the other hand this group of OS's because of belonging to a particular company, like Windows that offered by Microsoft company, is not desirable for many countries.
2. The open-source operating systems as the base of secure OS: Since code of these OS's is fully available we can be aware of all details to make them secure. Also there is no restriction in adding new security features to such an open operating systems. The most noteworthy open source OS is Linux. Linux family made of lots of distributions. One of the most popular distributions of Linux is Ubuntu. We choose Ubuntu for implementing the ideas presented in this article. Mint, Mageia, Fedora, OpenSUSE and Debian are other popular distributions that our idea is Implementable on them.

It may come to mind that if an updated anti-virus exists, there is no need to have special security features in operating system. Two replies can be raised for answering these doubts [1]:

1. Zero-day attacks: When a virus is widely reported, the virus definition is added to anti-viruses. But experience has shown that in the gap, caused by an antivirus detection delay, systems have a lot of sacrifice.
2. Targeted attacks: attacks aimed at specific organization as a target couldn't be identified by anti-viruses because the specific virus definition hasn't been published. So it seems that despite all the measures done, still it's necessary to secure your operating system.

SANS, one of the institutions of security training has analytical reports in the field of information security. According to the reports, nowadays the most vulnerable areas of operating systems are: web applications, backup softwares, antivirus softwares, database softwares, messaging softwares, servers provide voice over IP.

3. Related Works

Overally, securing operating system includes two following strategies: First is reviewing all configurations and securing them based on previous attacks experiences. The second is disabling unnecessary services in OS. In continue, these two approaches are briefly explained:

Reviewing all configurations and securing them based on previous attacks experiences: For this purpose, automatic tools can be used for making suitable configurations. A sample of automatic tools is Bastille Linux [18]. The Bastille Hardening program locks down an operating system, proactively configuring the system for increased security and decreasing its susceptibility to compromise. Bastille can also assess a system's current state of hardening, granularly reporting on each of the security settings with which it works. Also the configuration can be done by the professional user instead of automatic tools. In next paragraphs we have a survey of some available secure configurations in different parts of OS:

Securing remote connections: For this purpose we must put away clear text transmission and use SSH, the abbreviation of Secured Shell, in remote connections in order to have a secure transfer of information. SSH uses the RSA algorithm.

Tunneling and securing communication infrastructures: OpenSSL is the open source implementation of the SSL standard by Netscape company. It is suitable to make secure communications regardless of data type. Stunnel is an open-source program, used to provide universal TLS/SSL tunneling service. Tunneling in a simple definition is: packaging data packets of a protocol in packets of other protocols. In this debate usually the first protocol is non-secure and the second protocol is secure.

Securing domain name server (DNS) : One of the main services which servs Internet web sites, is the translation service of common and familiar names to their IP addresses. This service, due to being global and it's recursive property, has been much noticed by hackers [1]. Recursive property is the recursive attempt of main name server to achieve IP of received name if it hasn't it itself. The process result in answering the request which had come to domain name server. For having a secure name server two principles should be considered: Never give additional information to strangers and keep each software package, you use to provide the DNS service, updated. Translation of two above principles to technical language is; Disable or limit recursive property of DNS; Attacks, such as cache poisoning, are placed on this property. Another

hint is using split DNS.

Securing email service: Security of email service like other services on the network is important. Some of the malicious uses of these services reported includes [1]: eavesdropping confidential information which is transmitted via email, sending huge volume of letters to an individual, sending email with forged sender to deceive, publishing virus, illegal access to email server to get other attacks, publishing spam. Implementing an email service needs two agents; MTA and MDA. In each case there are some softwares that we can choose and use them securely.

Securing web service: Table 2 shows the security goals of a web server and their failure factors.

Table 2. Security goals of a web server & their vulnerabilities

Security goals	**Vulnerabilities**
System integrity	Theft of service Pirate servers and applications Password sniffing Rootkit and trojan program DoS targeting or participation
Data integrity	Vandalism, data tampering or site defacement Inadvertent file deletion or modification
Data confidentiality	Theft of personal information Leakage of personal data into URLs and logs
System and network availability	Unauthorized use of resources DoS & DDoS attacks Crash or freeze from resource exhaustion

For each vulnerability of table 2 necessary measures should be done, which is beyond of this article.

Securing database: The most popular database in servers based on Linux is MySQL. Securing database server, in comparison with email server or web server, is less complicated. Security problems that may be occur for the database include: data theft or loss of data or DoS attack on the database. One common attack in databases is SQL injection which the hacker attacks by injecting special statements. The hacker injects codes such as Perl or PHP scripts or any other languages in order to access confidential information. To prevent these attacks some measures can be done. Some of these client-side measures include: check all input variables in the language or limit the illicit characters of the script or check maximum size of inputs. Also we can use intrusion detection softwares in the server side.

Disabling unnecessary services: This approach depends on each case, and a common solution for all cases can't be presented. Here some global guidelines, which are collected from various references, will come in the following:

- Whatever is not explicitly permitted is prohibited.
- None of users should have powers more than what he needs for doing his duty [3].
- None of applications should have powers more than what he needs for doing his task. (This principle is a kind of generalized principle before.)
- Only install necessary softwares. Any other software should be deleted or disabled.
- Prohibit any unnecessary shell access to OS.
- None of services should be publicly available by default. If it is required to enable a specific service for all, it can be done but not by default and for all services.
- Applications which provide service to the public shall be run in the chrooted file system. An executable file without any reason should not have administrator access level.

  Until now we had a survey of existing solutions for securing servers. In next part we explain a different solution.

4. Impelementation

In this approach, we will try to isolate the major services of OS; Therefore, we can restrict any intrusion or damage to any part of operating system with minimum effect on the other parts. This idea has been tested that the creation of virtual machines in a system can prevent from spreading viruses and malicious codes to the whole system [4] .

For making this structure we need some kind of virtualizing. In the overall approach, there are three main virtualization technologies:

- Full virtualization
  - Bare metal (or native) virtualization [20]
  - Hosted virtualization [20]
- Paravirtualization
- Operating system-level virtualization (or

container)

The three techniques differ in complexity of implementation, breadth of OS support, performance in comparison with standalone server, and level of access to common resources. For example, full virtualization have wider scope of usage, but poorer performance. Paravirtualization have better performance, but can support fewer OSs because one has to modify the original OS. Virtualization on the OS level provides the best performance and scalability compared to other approaches. Performance difference of such systems can be as low as 1 to 3%, comparing with that of a standalone server [9]. Comparison of these three technologies is shown in table 3 [5].

Table 3. Comparision of virtualization technologies

virtualization technology	**Flexibility**	**complexity**	**performance**
Full virtualization	Any OS	low	low
Paravirtualization	Any OS with some limitations	high	Medium
OS-level virtualization	Host & guest must be the same (different distributions are accepted)	high	High

For implementing the idea of jail (another name for isolated VM) in Ubuntu operating system, all three above methods have been tested. The results will come in the follow:

Full virtualization: There are two forms of full virtualization. In bare metal virtualization, also known as native virtualization, the hypervisor runs directly on the hardware, without a host OS. In the other form of full virtualization, known as hosted virtualization, the hypervisor runs on top of the host OS; the host OS can be almost any common operating system such as Windows, Linux, or MacOS [20]. The most suitable software for implementing this method in Ubuntu is VirtualBox. It is open source and has stable versions. The result of implementing jail idea by this software was unacceptable. The efficiency of system largely decreased due to high resource consumption and the system almost went down. Some reasons of low performance of full virtualization are:

- A single application has two OSs to traverse; the guest OS and the hypervisor or host OS. More processing equates to slower responses and more overhead.
- Each OS takes space in memory, and memory is always the most constrained resource on a server.
- Hardware support and interoperability for all of the hardware on the market is difficult to emulate well, so it is often a source of slower response times and higher processing overhead.

Paravirtualization: A hypervisor provides the virtualization abstraction of the underlying computer system. In full virtualization, a guest operating system runs unmodified on a hypervisor. However, improved performance and efficiency is achieved by having the guest operating system communicate with the hypervisor. By allowing the guest operating system to indicate its intent to the hypervisor, each can cooperate to obtain better performance when running in a virtual machine. This type of communication is paravirtualization. One of the most suitable softwares for implementing this method in Ubuntu is Xen [7]. This method had problems like previous method with some improvements [8].

Operating system-level virtualization: In this technique the kernel of an operating system allows for multiple isolated userspace instances, instead of just one. OS-level virtualization have been designed to provide the required isolation and security to run multiple applications or copies of the same OS (but different distributions of the OS) on the same server. Considering the above table, the best way to implement Jail idea is operating system-level virtualization. The advantage of this method is neglectable overload on system performance. It means that large number of OS-level virtualized machines, in the role of jail, can be simultaneously performed on one system [3]. Thus, any important service in operating system can have a special machine to run independently. The process of making a secure OS based on our idea is shown in figure 1.

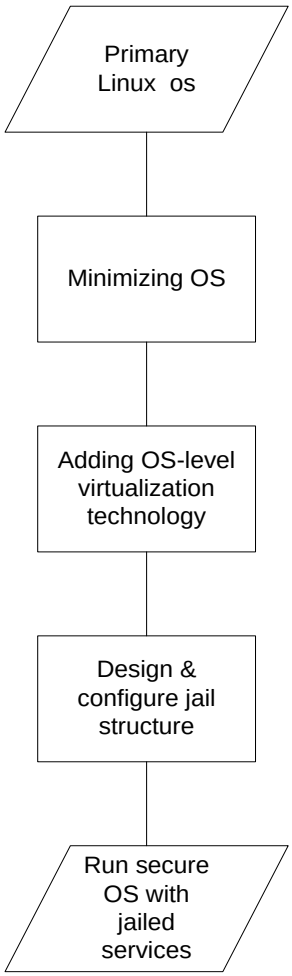

Figure 1. High-level preparation process of the secure operating system

The most suitable software for implementing this method is the OpenVZ [9]. Another software in this context is called Linux-Vserver, but OpenVZ was selected due to a better support and stable versions [10]. Since OpenVZ is the open source version of a commercial software called Virtuozzo, it's support is better [11]. In this study OpenVZ was installed on long-term version of Ubuntu. It should be mentioned that installation of this software, because of changes in the kernel, is difficult and time consuming, but after successful installation the result was really impressive. For example, 100 jail structures run concurrently on an ordinary hardware, and there wasn't any remarkable changes in the system CPU and memory usage. In the following, steps of making Jails will come and then we have a detailed performance analysis:

1. OpenVZ software was chosen for this purpose. It's recommended to use a separate partition for container private directories. We used */var/lib/vz/private/<CTID>*.

2. To install this software first, we must add the relevant data repository address of Linux then get the appropriate packages to Install:

Sudo wget
http://debian.systs.org/dso_archiv_signing_key.asc
sudo apt-key add dso_archiv_signing_key.asc
sudo rm dso_archiv_signing_key.asc
sudo apt-get update
$ Sudo apt-get install linux-openvz vzctl
$ Sudo apt-get remove - purge - auto-remove
linux-image-. server*

3. Then we should do some configurations; End of file /etc/sysctl.conf was edited like this:

 # On Hardware Node we generally need
 # Packet forwarding enabled and proxy arp disabled
net.ipv4.conf.default.forwarding = 1
net.ipv4.conf.default.proxy_arp = 1
net.ipv4.ip_forward = 1
 # Enables source route verification
net.ipv4.conf.all.rp_filter = 1
 # Enables the magic-sysrq key
kernel.sysrq = 1
 # TCP Explict Congestion Notification
 # Net.ipv4.tcp_ecn = 0
 # We do not want all our interfaces to send redirects
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.all.send_redirects = 0

4. Apply the changes in sysctl.conf file:

$ Sudo sysctl -p

5. In this state after restarting and booting from pached kernel, we have jail ready kernel. It's also available out of *uname –r* command. Another confirmation command can be *ps ax | grep vz.* A network interface for jails should be seen in list:

 # ifconfig
venet0 Link encap:UNSPEC HWaddr
00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
UP BROADCAST POINTOPOINT RUNNING NOARP
MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

6. Now, to install a container you can use an OS template. OS templates can be preconfigured for any security issues relating to their special use. For example:

 # sudo apt-get install vzctl-ostmpl-ubuntu-12.04-x86_64

Now the infrastructure of Jail, which is OS-level virtualization technology, is added to our primary operating system, so we can make jail structure by added virtualization technology. Figure 2 shows our status.

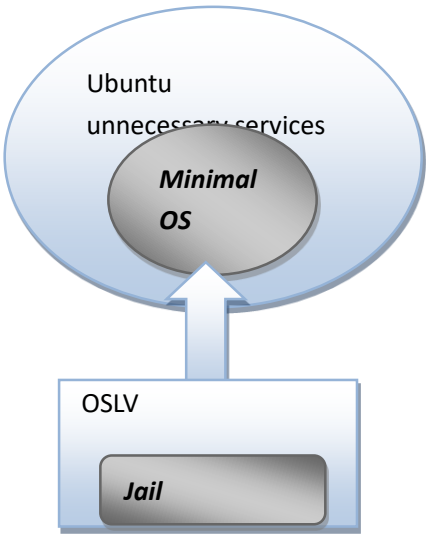

Figure 2. Overview of the secure operating system

Jails can be more secured by some measures;

- A two layer network filter: In addition to IPtable rules in host, we can have IPtable rules per jail.
- Complete isolation: Complete isolation ensures that the jails are secure and have full functional, fault and performance isolation. Isolation is achieved through multiple layers of security designed to ensure that each jail is secure, isolated and unaffected from other jails on the same physical server. An abstraction layer in kernel, mediates activity to the kernel and prevents any single jail from taking the entire server down.
- File system access limitation: File system true configs can ensure that a user cannot access any other jail or part of host server.
- Copy on write technology: Copy on write technology, which makes a local copy of anything unique in the jail can be used for snapshot and backup. Each jail reads from the base environment but write into the jail's private workspace. [24]
- Using hardened OS templates: With templates, many repetitive installation and hardening configuration tasks can be avoided. The result is a fully installed, ready to operate (virtual) server in less time than manual installation [23].
- Monitoring and logging jail: A separated jail for monitoring other jails and keeping their logs. Also logs can be sent to remote machine.

In this step we should plan our jail structure. The plan, depend on the server usage, can be changed. For example, any of these Jail structures can include one of the major services of operating system such as web or mail service. Each jailed service will be run separated from other services. In the other strategy some related services, with the same level of security needed, can be put in one jail. Also we can make one or more honeypot jails. A honeypot is a trap set to detect unauthorized use of information systems. The important result is that the failure of other services doesn't affect on jailed services and vice versa. A sample jail configuration has shown in figure 3.

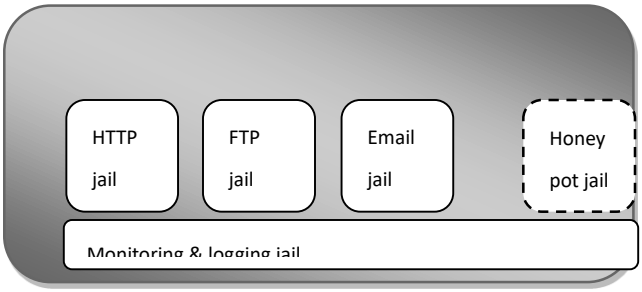

Figure 3. An example of configuring the operating system after creation of jail structure

In the figure 3, each module is independent of other modules. Failure of each one has no effect on others. For example, if the HTTP service is targeted by DoS attack only web service goes down and other services continue their task. Also, HTTP jail can be recovered easily for solving the problem and without any effect on other services. Restarting HTTP jail will be done by only a command from host [12]. A honeypot jail is a system that looks and acts like a production environment but is actually a monitored trap, deployed in a network with enough interesting data to attract hackers, but created to log their activity and keep them from causing damage to the actual production environment [19]. Each jail has its own:

- Files: System libraries, applications, virtualized /proc and /sys, virtualized locks etc.
- Users and groups: Each container has its own root user, as well as other users and groups.
- Process tree: A container only sees its own processes starting from init.
- Network: Virtual network device, which allows a container to have its own IP addresses, as well as a set of netfilter (iptables) and routing rules.

- Devices: If needed, any container can be granted access to real devices like network interfaces, serial ports, disk partitions, etc.
- IPC objects: Shared memory, semaphores, and messages.

5. Results

The result of running jails with this method [5] and its effect on system performance is shown in the table 4:

Table 4. Mean response time (ms)

Number of running threads	Primary system	Jailed system
500	11	12
550	12	13
600	13	13
650	14	16
700	15	20
750	19	25
800	22	30

As we can see in table 4, the mean response time of jailed system is a bit more than primary system and is venial. These results are shown in figure 4, again.

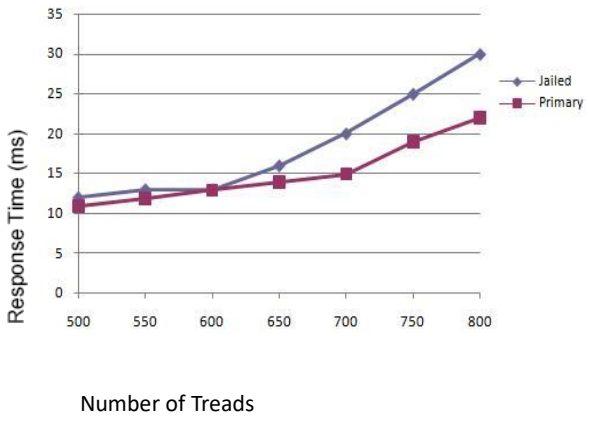

Figure 4. Effect of jail mechanism on response time of system

As shown in figure 4, in a server without jail structure HTTP service is overloaded by a DDoS attack. In this situation database service went down. The attack is simulated by DoSHTTP testing tool version 2.0 on some virtual machines. Requests lost of HTTP flood arrived 99.99% and similar state for database requests. But in figure 5 we see the same scenario on jailed OS.

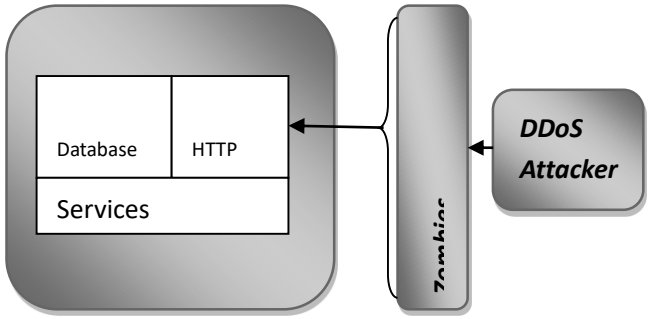

Figure 5. Impact of DDosS attack on all services of server

As shown in figure 5 we have a jailed structure that its HTTP service is targeted by a DDoS attack. The same scenario of previous figure but only HTTP jail is overloaded and database jail continue its service. It means requests lost of HTTP flood arrived 99.99% but database is up. Overloading of HTTP service is limited to jail allocated resources. Also monitoring jail can detect problem and act.

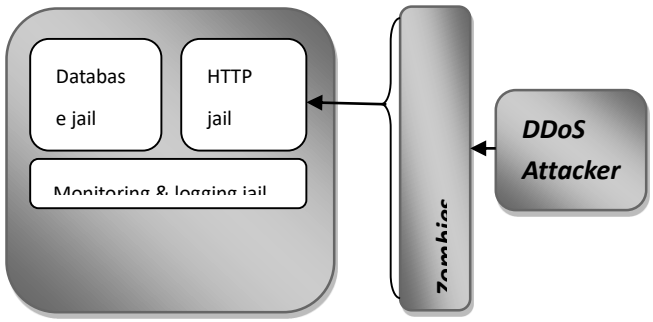

Figure 6. Impact of DDosS attack on targeted service in jailed OS

Finally, if the two described traditional methods combine with jail idea, an acceptable secure product will yield.

6. Conclusions

The propose of this research was improving security of server operating systems. First, available methods for making secure operating system was expressed. Then another approach to build a secure server was introduced. New method was based on creating containers within an operating system and isolation of critical services within the jails.

The importance of this model is in this point: it

improves security factors without any bad impacts on performance of OS. In the other word the performance of our jailed OS is like the primary OS, but CIA (confidentiality, Integrity, and availability) factors are significantly improved. Confidentialrity is improved by partitioned file system. Integrity of data is obtained by limiting access to jailed data. Availability of services is significantly increases by options like backup jail and live migration.

The other important result of our research is improving continuity of services after attacks. In usual OS when an attacker hacks the OS he can access all services and disturb them, but in our jailed OS only hacked service is disturbed and other services continue their tasks. Also disturbed service can be recovered immediately by starting it's back up jail. Some future works that can be offered, include:

- Improving available methods for OS-level virtualizing exactly in tools.
- Providing a conceptual model for finding the optimum form of partitioning the system to jails.
- And a monitoring and logging system specialized for virtualized environments.

References

[1] M. D. Bauer, Building Secure Servers with Linux, O'Reilly, 2002.

[2] Udo Steinberg, Bernhard Kauer, NOVA: a microhypervisor-based secure virtualization architecture, Proceedings of the 5th European conference on Computer systems, April 13-16, 2010, Paris, France.

[3] National Institute of Standards and Technology, http://nist.gov, 2012.

[4] Yoshiki Sameshima, Hideaki Saisho, Tsutomu Matsumoto, and Norihisa Komoda, Windows Vault: Prevention of Virus Infection and Secret Leakage with Secure OS and Virtual Machine, in Pre-Proceedings of the 8th International Workshop of Information Security Applications 2007 (WISA 2007), pp.249-261, 2007.

[5] P. Padala, X. Zhu, Z. Wang, S. Singhal, K. G. Shin, Performance Evaluation of Virtualization Technologies for Server Consolidation, HPL-2007-59.

[6] Mihai Christodorescu, Reiner Sailer Douglas Lee Schales, Daniele Sgandurra, Diego Zamboni, Cloud security is not (just) virtualization security: a short paper, Proceedings of the 2009 ACM workshop on Cloud computing security, November 13-13, 2009, Chicago, Illinois, USA.

[7] XenSource, http://www.xensource.com, 2012.

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. War eld. Xen and the art of virtualization, In Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), October 2003.

[9] OpenVZ: http://openvz.org, 2012.

[10] Linux VServer, http://linux-vserver.org, 2010.

[11] Virtuozzo, http://www.swsoft.com/en/virtuozzo, 2010.

[12] S. Bhattiprolu, W. Biederman, S. Hallyn, D. Lezcano, Virtual servers and checkpoint/restart in mainstream Linux, ACM SIGOPS Operating Systems Review, Volume 42 Issue 5 (July 2008).

[13] Fred Cohen, The Virtualization Solution, IEEE Security and Privacy, vol. 8, no. 3, pp. 60-63, May/June 2010.

[14] Flavio Lombardi, Roberto Di Pietro, Secure virtualization for cloud computing, Journal of Network and Computer Applications, Volume 34, Issue 4, 2010.

[15] Diane Barrett, Gregory Kipper, Virtualization Challenges, Virtualization and Forensics, Syngress, Boston, 2010.

[16] Flavio Lombardi, Roberto Di Pietro, KvmSec: a security extension for Linux kernel virtual machines, Proceedings of the 2009 ACM symposium on Applied Computing, March 08-12, 2009, Honolulu, Hawaii.

[17] Flavio Lombardi and Roberto Di Pietro. Secure virtualization for cloud computing. J. Netw. Comput. Appl. 34, 4. July 2011.

[18] Bastille linux, http://www.bastille-linux.org, 2012.

[19] John Hoopes. Virtualization for Security: Including Sandboxing, Disaster Recovery, High Availability, Forensic Analysis, and Honeypotting. Syngress Publishing. 2008.

[20] Karen A. Scarfone, Murugiah P. Souppaya, and Paul Hoffman. SP 800-125. Guide to Security for Full Virtualization Technologies. Technical Report. NIST, Gaithersburg, MD, United States. 2011.

[21] Yan Wen, Jinjing Zhao, Gang Zhao, Hua Chen, and Dongxia Wang. A Survey of Virtualization Technologies Focusing on Untrusted Code Execution. In Proceedings of the 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS '12). IEEE Computer Society, Washington, DC, USA, 378-383. 2012.

[22] Kun Wang, Jia Rao, and Cheng-Zhong Xu. Rethink the virtual machine template. In Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE '11). ACM, New York, NY, USA, 39-50. 2011.

[23] http://www.vmware.com/pdf/vc_2_templates_usage_best_practices_wp.pdf, 2012.

[24] Yang Yu. 2007. Os-Level Virtualization and its Applications. Ph.D. Dissertation. State University of New York at Stony Brook, Stony Brook, NY, USA. AAI3337611.

[25] Ryan Shea and Jiangchuan Liu. Understanding the impact of denial of service attacks on virtual machines. In Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service (IWQoS '12). IEEE Press, Piscataway, NJ, USA, Article 27, 9 pages. 2012.

[26] Yih Huang and Anup K. Ghosh. Automating Intrusion Response via Virtualization for Realizing Uninterruptible Web Services. In Proceedings of the 2009 Eighth IEEE International Symposium on Network Computing and Applications (NCA '09). IEEE Computer Society, Washington, DC, USA, 114-117. 2009.

[27] Ruo Ando, Zong-Hua Zhang, Youki Kadobayashi, and Yoichi Shinoda. A Dynamic Protection System of Web Server in Virtual Cluster Using Live Migration. In Proceedings of the 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC '09). IEEE Computer Society, Washington, DC, USA, 95-102. 2009.

[28] Ali Peiravi, Mehdi Peiravi, Internet security - cyber crime Paradox, Journal of American Science, 6(1):15-24, 2010.

6/18/2023