Advances in Computer Science and Information Technology (ACSIT)
p-ISSN: 2393-9907; e-ISSN: 2393-9915; Volume 2, Issue 13; July-September, 2015 pp. 82-86

http://www.krishisanskriti.org/Publication.html

# Overhead Analysis between Xen & KVM for Security Primitives

**Shikha Dixit[1] and Rahul Hada[2]**

*[1]School of Engineering & Technology, Poornima University, Jaipur*
*[2]Department of Computer Engineering, Poornima University*
*E-mail: [1]shikhadixit032@gmail.com, [2]rahul.hada@poornima.edu.in*

**Abstract—***In recent years, Virtualization is the fundamental technology for corporate data centre consolidation and cloud computing. Security is a critical issue in Virtualization Technology. To overcome Security Issues such as Authenticity and Integrity, an analysis was carried out between Xen & KVM Hypervisors by creating virtual machines on it using Red Hat Linux Environment and in the last conclusion will be presented that, which virtualization platform (Xen/KVM) provide better performance by reducing overhead after applying SELinux Security Primitive. In this paper some of the works related to the implementation of Security Primitives on Virtual Machines deployed on Hypervisors has been done, which helped to understand the risk that occurs in virtualized environment and improve system reliability by applying SELinux Security Primitives on Xen & KVM Hypervisor. Furthermore, Swappiness & Huge Memory techniques was used for reducing the increased overhead due to security primitives. Experimental work included implementation of various virtual machines having Red Hat as guest OS on hypervisor Xen and KVM to analysis which virtualization platform or hypervisor (Xen & KVM) provide better performance by reducing overhead for security primitives.*

**Keywords***: Xen, KVM, SELinux, Swappiness, Huge Memory.*

## 1. INTRODUCTION

The recent growth in cloud environments has accelerated the advancement of virtualization through hypervisors; with so many different virtualization technologies, it is difficult to ascertain how different hypervisors impact virtual machines and application performance and whether the same performance can be achieved for each hypervisor. Many different Hypervisors (both open source and commercial) exist today, each with their own advantages and disadvantages. This introduces a large number of new and challenging research questions.

The key technology enabling cloud computing is virtualization and the hypervisor is the software layer that implements it. A number of security concerns that needs to be tackle due to the privilege level of the hypervisor.

In recent years, with the rapid development of Internet and the commonly use of computer, there are more comprehensive ways for sharing information resources.

While the information sharing, it is also necessary for us to prevent both non-authorized users and programs access to sensitive information. As we know access control is designed to protect the security of the information which was stored and handled in information system. There are two kinds of access control we usually use nowadays which are discretionary access control (DAC) and mandatory access control (MAC).

Discretionary Access Control is an effective way to protect the computer system resources from being accessed illegally. The owner of the resource specifies whether other users can access the resource. However, it has an obvious drawback that is this control is independent. Although this independence provides users with great flexibility; it also brings a serious security problem.

SELinux affords Linux a flexible, configurable MAC mechanism, which provides a mechanism to enforce the access control based on confidentiality and integrity requirements, besides it offers an effective protection of information security.

In this paper we will describe in what way SELinux uses various models and policies to secure the information safety, especially represent how multilevel security policy and type enforcement provide confidentiality and integrity protection.

## 2. SELINUX

Security-Enhanced Linux (SELinux) is an implementation of a mandatory access control mechanism in the Linux kernel, checking for allowed operations after standard discretionary access controls are checked. It was created by the National Security Agency and can enforce rules on files and processes in a Linux system, and on their actions, based on defined policies.

DAC access decisions are only based on user identity and ownership, ignoring other security-relevant information such as the role of the user, the function and trustworthiness of the program, and the sensitivity and integrity of the data. Each user typically has complete discretion over their files, making it difficult to enforce a system-wide security policy.

Security-Enhanced Linux (SELinux) adds Mandatory Access Control (MAC) to the Linux kernel, and is enabled by default in Red Hat Enterprise Linux. A general purpose MAC architecture needs the ability to enforce an administratively-set security policy over all processes and files in the system, basing decisions on labels containing a variety of security-relevant information.

On Linux operating systems that run SELinux, there are Linux users as well as SELinux users. SELinux users are part of SELinux policy.

## 3. XEN & KVM HYPERVISOR OVERVIEW

### 3.1 Xen Hypervisor

A virtual-machine monitor for IA-32, x86-64, Itanium, and ARM architectures, Xen allows several guest operating systems to execute on the same computer hardware concurrently. Xen systems have a structure with the Xen hypervisor as the lowest and most privileged layer.

The basic components of a Xen-based virtualization environment are the Xen hypervisor, the Domain0, any number of other VM Guests, and the tools, commands, and configuration files that let you manage virtualization. Collectively, the physical computer running all these components is referred to as a VM Host Server because together these components form a platform for hosting virtual machines.

The Xen hypervisor sometimes referred to generically as a virtual machine monitor, is an open-source software program that coordinates the low-level interaction between virtual machines and physical hardware.

The Dom0 is a virtual machine host environment, also referred to as Domain0 or controlling.

### 3.2 KVM Hypervisor

A virtualization infrastructure for the Linux kernel, KVM supports native virtualization on processors with hardware virtualization extensions.

KVM is a full virtualization solution for x86 processors supporting hardware virtualization (Intel VT or AMD-V). It consists of two main components: A set of Kernel modules (KVM.ko, KVM-intel.ko, and KVM-amd.ko) providing the core virtualization infrastructure and processor specific drivers and a user space program (qemu-KVM) that provides emulation for virtual devices and control mechanisms to manage VM Guests (virtual machines).

The term KVM more properly refers to the Kernel level virtualization functionality, but is in practice more commonly used to reference the user space component. VM Guests (virtual machines), virtual storage and networks can be managed with libvirt-based and QEMU tools. libvirt is a library that provides an API to manage VM Guests based on different virtualization solutions, among them KVM and Xen.

## 4. RELATED WORK

In this work many of the parameters were used to check the performance of Xen & KVM Hypervisors having guest virtual machines in virtualized environment. All the performance parameters are used with and without security primitives. The performance parameters are as follows:

1. CPU Utilization

2. Memory

3. Read/Write Operations

**4.1 CPU Utilization:** CPU utilization refers to a computer's usage of processing resources, or the amount of work handled by a CPU. Actual CPU utilization varies depending on the amount and type of managed computing tasks. Certain tasks require heavy CPU time, while others require less because of non-CPU resource requirements.

**4.2 Memory:** Memory refers to any information or data, often in binary format, that a machine or technology can recall and use. Data in storage memory remains, and the computer accesses it through a hard drive.

**4.1 Read/Write Operation:** Read/write operations are an extremely important component in determining the overall performance of the Memory and hard disk, since they play such an important role in the storage and retrieval of data.

**Read:** Describes the operations carried out by the processor when a memory read is executed. Number of sectors read from the device. The size of a sector is 512 bytes.

**Write:** Sequence of operations write, Number of sectors written to the device. The size of a sector is 512 bytes.

## 5. EXPERIMENT SETUP

### 5.1 Hardware Specifications

| | |
|---|---|
| Processor | Core i5 @ 2.50GHz |
| Memory | 4 GB |
| Virtualization | Hardware Assisted Virtualization Enabled |

### 5.2 Software Specifications

| | |
|---|---|
| Native Host Operating System | Red Hat 6 |
| VMM | Xen 4.1 & KVM |
| Dom-0 (OS) | Red Hat 6 |
| Dom-U (Guest OS) | Red Hat 6 |

## 6. RESULT & DISCUSSIONS

The experiment results are interpreted in the form of statistical graphs and the results are analyzed with various statistical comparisons in accordance to a few of our intriguing cases. All our interpreted results are presented as an average value of

the obtained data after repeating the experiment for a significant number of values.

Table 6.1 shows the performance of Xen & KVM Hypervisor in terms of CPU Utilization without any Security Primitive, with Security primitive (SELinux) and it is observed that to reduce increased overhead, Swappiness has been used at certain value to get minimum utilization.

**Table 6.1: CPU Utilization of Xen versus KVM with or without Security Primitives after reducing overhead**

| Hypervisor | Without Security primitive | With Security Primitive | After Overhead Reduction |
|---|---|---|---|
| | | SELinux | SELinux |
| Xen | 41.45 | 45.53 | 25.44 |
| KVM | 32.87 | 37.64 | 20.26 |

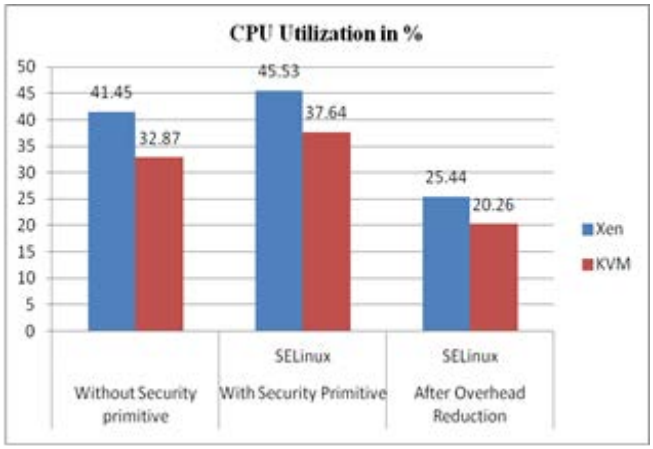


**Fig. 6.1: Performance between Xen and KVM during CPU Utilization without Security Primitives, with Security Primitives and after reducing overhead**

Fig 6.1 shows that the comparative analysis of CPU Utilization performance parameter between Xen & KVM Hypervisor. Swappiness is used to reduce overhead increases on CPU Utilization performance after applying Security primitive SELinux uses 10 to 100 values having interval of 10. It is observed from the results that after applying Security Primitive SELinux on Xen Hypervisors overhead increases in terms of CPU Utilization performance reduced by the value of swappiness at 20. While in case of SELinux Security Primitive on KVM hypervisors the value of swappiness is 40.

Table 6.2 shows the performance of Xen & KVM Hypervisor in terms of Memory Usage without any Security Primitive, with Security primitive (SELinux) and it is observed that to reduce increased overhead, Huge Memory Technique has been used at certain value of Huge Page Size to get minimum utilization.

**Table 6.2: Memory Used of Xen versus KVM with or without Security Primitives after reducing overhead**

| Hypervisor | Without Security primitive | With Security Primitive | After Overhead Reduction |
|---|---|---|---|
| | | SELinux | SELinux |
| Xen | 2248578 | 2928737 | 2559378 |
| KVM | 1837317 | 2875255 | 1547393 |

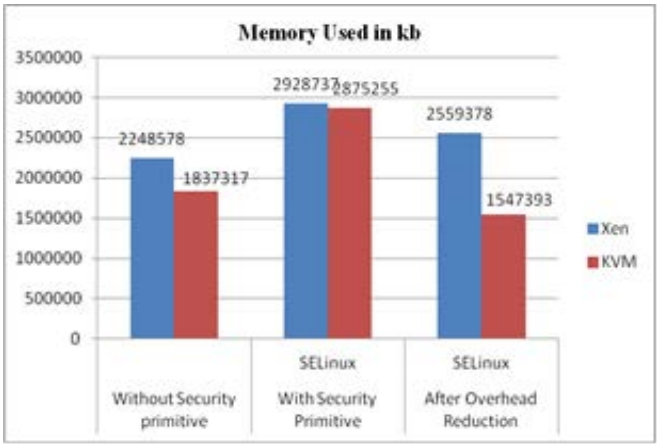


**Fig. 6.2: Memory Usage between Xen and KVM without Security Primitives, with Security Primitives and after reducing overhead**

Fig 6.2 shows that the comparative analysis of used Memory performance parameter between Xen & KVM Hypervisor. Huge Memory technique having huge pages is used to reduce overhead increases on Memory uses after applying Security primitives rithm and SELinux from 500 to 1500 huge page sizes having interval of 500. It is observed from the results that after applying Security Primitive rithm& SELinux on Xen Hypervisor overhead increases in terms of used memory performance reduced by the value of huge pages at 500 & 1000 respectively while in case of KVM hypervisor this value of huge pages is 1000 for both Security primitives rithm and SELinux.

Table 6.3 shows the performance of Xen & KVM Hypervisor in terms of Read Operations without any Security Primitive, with Security primitive (SELinux) and it is observed that to reduce increased overhead, Swappiness has been used at certain value to get minimum utilization.

**Table 6.3: Read Operations of Xen versus KVM with or without Security Primitives after reducing overhead**

| Hypervisor | Without Security primitive | With Security Primitive | After Overhead Reduction |
|---|---|---|---|
| | | SELinux | SELinux |
| Xen | 1725.67 | 808.05 | 1644.71 |
| KVM | 2591.93 | 719 | 2083.66 |


**Fig. 6.3: Performance between Xen and KVM during Read Operations without Security Primitives, with Security Primitives and after reducing overhead**

Fig 6.3 shows that the comparative analysis of Read operation performance parameter between Xen & KVM Hypervisor. Swappiness is used to reduce overhead increases on Read Operation performance after applying Security primitives SELinux from 10 to 100 values having interval of 10. It is observed from the results that after applying Security Primitive SELinux on Xen Hypervisors overhead increases in terms of Read operation performance reduced by the value of swappiness at 30 respectively while in case of KVM hypervisor this value of swappiness is 30 for Security Primitives rithm and SELinux respectively.

Table 6.4 shows the performance of Xen & KVM Hypervisor in terms of Write Operations without any Security Primitive, with Security primitive (SELinux) and it is observed that to reduce increased overhead, Swappiness has been used at certain value to get minimum utilization.

**Table 6.4: Write Operations of Xen versus KVM with or without Security Primitives after reducing overhead**

| Hypervisor | Without Security primitive | With Security Primitive | After Overhead Reduction |
|---|---|---|---|
| | | SELinux | SELinux |
| **Xen** | 10821.5 | 5548.21 | 5817.62 |
| **KVM** | 9514.06 | 4543.54 | 4825.45 |

Fig 6.4 shows that the comparative analysis of Write operation performance parameter between Xen & KVM Hypervisor. Swappiness is used to reduce overhead increases on Write Operation performance after applying Security primitive SELinux from 10 to 100 values having interval of 10. It is observed from the results that after applying Security Primitive SELinux on Xen Hypervisors overhead increases in terms of Write operation performance reduced by the value of swappiness 60 respectively while in case of KVM hypervisor this value of swappiness is 60 for both Security Primitives rithm and SELinux respectively.

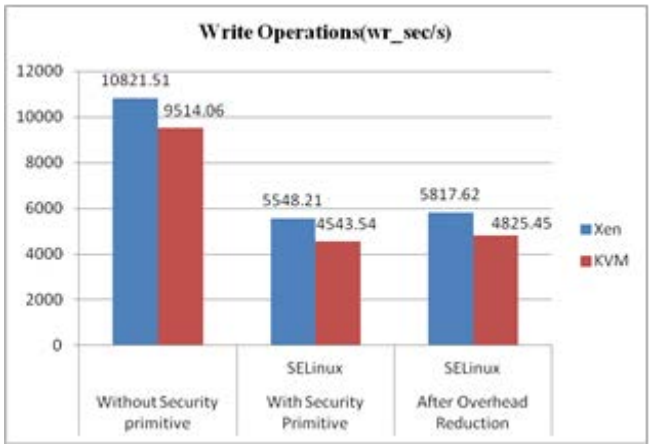


**Fig. 6.4: Performance between Xen and KVM during Write Operations without Security Primitives, with Security Primitives and after reducing overhead**

## 7. CONCLUSION

In this work, implementation of virtual machines have been performed on Hypervisors Xen and KVM to analyze which virtualization platform (Xen/KVM) provide better performance by reducing overhead for Security Primitive in virtualized environment by applying SELinux Security primitives. Results shows that KVM Hypervisor provides better performance by utilizing less CPU & Memory with reduced overhead even after applying Security Primitives in comparison to the Xen Hypervisor in Linux Environment. On the other hand, KVM had higher read operations as compare to Xen while Xen had higher write operations than KVM according to experimental results. So, it has been believed that KVM may have performed better than Xen in terms of CPU Utilization, Memory Usages and Read Operations.

## 8. ACKNOWLEDGEMENTS

I would like to express my deep gratitude and thanks to **Dr. Mahesh Bundele (Coordinator, Research), Poornima University** for giving me an opportunity to work under his guidance for review of research papers and his consistent motivation & direction in this regard. I would also express my sincere thanks to **Mr. Rahul Hada (Associate Professor, CE), Poornima University** for their guidance and support.

## REFERENCES

[1] Sabahi, F., "Virtualization-level security in cloud computing," 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN), pp.250, 254, 27-29 May 2011.

[2] Sailer, R.; Jaeger, T.; Valdez, E.; Caceres, R.; Perez, R.; Berger, S.; Griffin, J.L.; van Doorn, L., "Building a MAC-based security architecture for the Xen open-source hypervisor," 21st Annual Conference Computer Security Applications, pp.10 pp.,285, 5-9 Dec. 2005.

[3] Sahoo, J.; Mohapatra, S.; Lath, R., "Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues," 2010 Second International Conference on Computer and Network Technology (ICCNT), pp.222, 226, 23-25 April 2010.

[4] [Somani, G.; Agarwal, A.; Ladha, S., "Overhead Analysis of Security Primitives in Cloud," 2012 International Symposium on Cloud and Services Computing (ISCOS), pp.129, 135, 17-18 Dec. 2012.

[5] Todd Deshane, Muli Ben-Yehuda, Amit Shah and Balaji Rao, “Quantitative Comparison of Xen and KVM”, Xen Summit, June 23-24, 2008.

[6] Xiao Xu; Chuangbai Xiao; Chaoqin Gao; Guozhong Tian, "A study on confidentiality and integrity protection of SELinux," 2010 International Conference on Networking and Information Technology (ICNIT), pp.269, 273, 11-12 June 2010.

[7] Xianghua Xu; Peipei Shan; Jian Wan; Yucheng Jiang, "Performance Evaluation of the CPU Scheduler in XEN," 2008 International Symposium on Information Science and Engineering, ISISE, pp.68, 72, 20-22 Dec. 2008.

[8] Xiantao Zhang; Yaozu Dong, "Optimizing Xen VMM Based on Intel® Virtualization Technology," 2008 International Conference on Internet Computing in Science and Engineering, ICICSE, pp.367, 374, 28-29 Jan. 2008.

[9] YamunaDevi, L.; Aruna, P.; Sudha, D.D.; Priya, N., "Security in Virtual Machine Live Migration for KVM," 2011 International Conference on Process Automation, Control and Computing (PACC), pp.1,6, 20-22 July 2011.