5-2018

KVM Based Virtualization and Remote Management

Srinath Reddy Pasunuru
St. Cloud State University, spasunuru@stcloudstate.edu

KVM Based Virtualization and Remote Management

by

Srinath Reddy Pasunuru

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

in Information Assurance

May, 2018

Starred Paper Committee
Susantha Herath, Chairperson
Ezzat Kirmani
Sneh Kalia

Abstract

In the recent past, cloud computing is the most significant shifts and Kernel Virtual Machine (KVM) is the most commonly deployed hypervisor which are used in the IaaS layer of the cloud computing systems. The Hypervisor is the one which provides the complete virtualization environment which will intend to virtualize as much as hardware and systems which will include the CPUs, Memory, network interfaces and so on. Because of the virtualization technologies such as the KVM and others such as ESXi, there has been a significant decrease in the usage if the resources and decrease in the costs involved. Firstly, in this Paper I will be discussing about the different hypervisors that are used for the virtualization of the systems, then I discuss about how the virtualization using the Kernel Virtual Machine (KVM) is made easy, and then discuss about the Host security, the access and the security of the KVM virtual machines by the remote management using the Secure Sell (SSH) tunnels, Simple Authentication and Secure Layer (SASL) authentication and Transport Layer Security (TLS).

Table of Contents

Chapter Page

List of Figures

Figure Page

Figure Page

Chapter I: Introduction

Introduction

Virtualization is the one which describes the technology in which an operating system, the applications that are used by the user or the memory storage is abstracted from the software or the hardware underlying it. The major importance of the virtualization technology is the server virtualization, which will use a software layer called as Hypervisor for emulating the hardware underlying it which will include the I/O, network traffic, CPU's memory. The utilization of the CPU on the traditional servers ranges from the 5% to 40% [1]. Then the virtualization can reduce the energy consumption as high as 80% and save the consumption up to 85% [2]. There are different types of hypervisors used in the real-time usage and Kernel-based Virtual Machine (KVM) is one such type of a Hypervisor.

Now-a-days, virtualization has gained much importance and many Companies are using the KVM as the virtualization technologies for the industry applications for reducing the resources they are using and increase in the efficiency. With the help of the KVM hypervisor, one can perform the live migration and move to the presently running virtual machine which are hosted by the physical hosts without interrupting the service. Running the virtual machines remotely includes running them in the cloud. For running the Virtual Machines in the cloud KVM requires the virtualization extensions which are not available in the cloud, so in the cloud there is a technology called as the "nested virtualization" which will act as the hypervisor and gets access to the VM. On the host, every virtual machine is running as the KVM process. This will enable the VM manager to check the static and dynamic status of the VMs and take action on the malicious activities [3].

There are many types of the virtualization such as the platform virtualization, full virtualization, para-virtualization, emulation virtualization, operating system virtualization, application level virtualization, desktop virtualization. The architecture of the full virtualization is as shown below.

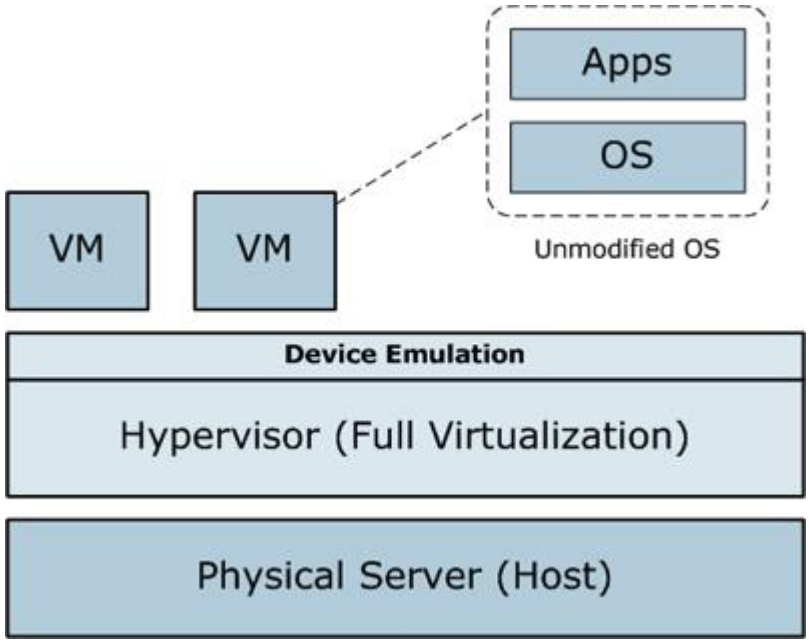


Figure 1. Full virtualization architecture (from www.datamation.com).

In full virtualization, privileged instructions are emulated to overcome the limitations arising from the guest operating system running in ring 1 and VMM running in Ring 0 [4]. Full virtualization was implemented in the initial generation of the X86 VMMs. It mainly depends on the techniques such as the binary translation to trap it and also virtualize some of the sensitive and also the non-virtualized instructions. So, in the binary translation some of the system calls are then interpreted and are written dynamically. The below figure will show how the Guest OS can access the host computer hardware through

the Ring 1 for the privileged instructions and how un-privileged instructions are executed without the involvement of Ring 1 [4].

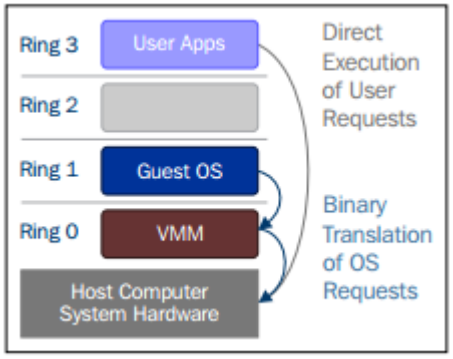


Figure 2. Host computer system architecture for full virtualization.

With the above approach the instructions that are critical can be discovers which can be dynamically or statically at the runtime and are then replaced with some traps in to the virtual machine manager which are to be emulated in the software. A binary translation can be used to generate a large performance overhead when compared to the VM running on the virtualized architectures which are native.

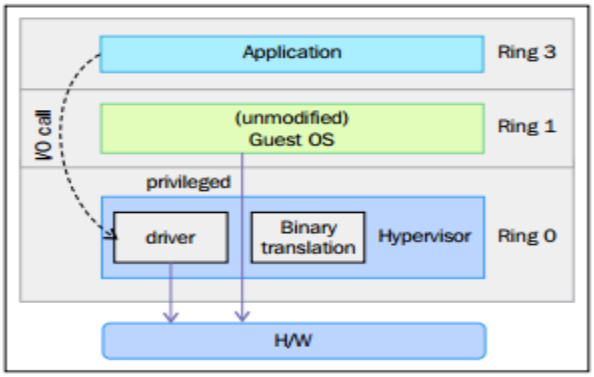


Figure 3. Full virtualization.

As shown in the above figure, the unmodified guest operating systems can be used while utilizing the full virtualization, by which there will be no need of changing the guest operating systems kernel for running the virtual machine manager. When the privileged operations are executed by the guest kernel, then the virtual machine manager will provide the CPU emulation that can be used to modify and handle the CPU operations. But this form of virtualization will cause the performance overhead when compared to the other mode of the virtualization such as the para-virtualization.

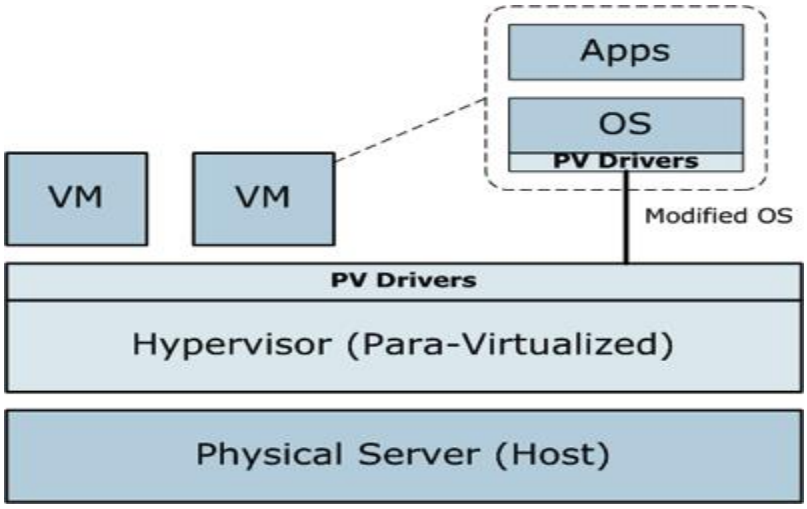


Figure 4. Para-virtualization architecture (from www.datamation.com).

The above figure shows the architecture of the para-virtualization, which is a virtualization technology in which the guest operating system will be directly run on the hypervisor unlike the full virtualization directly running on the hardware of the host [5]. In this type of the virtualization, the hypervisor which is supporting the para virtualization is installed on the host operating system. Here the hypervisor will then act as the virtualization layer, which means the hypervisor here will act as the host on which the

guest operating systems are loaded. Then the guest operating systems will make the requests for the hypervisor to make use of the necessary hardware resources. Some of the examples that are that come under the para-virtualization are Xen-pv and ESXserver.

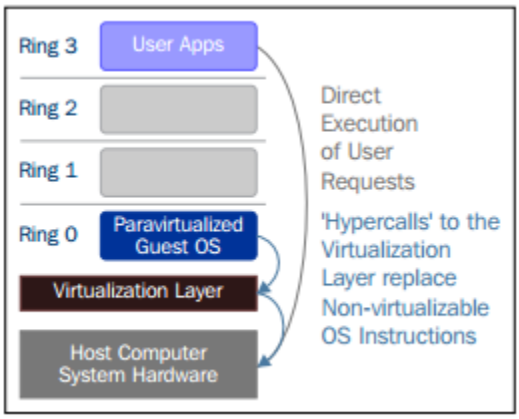


Figure 5. Host computer system architecture of para-virtualization.

From the above figure, we can see that the guest kernel is modified to run on the virtual machine manager. Defining in the other terms, the guest kernel will know that it has been virtualized. The instructions or the operations which are privileged are supposed to be run on the ring 0 have been replaced with the calls which are known as the hypercalls, which will talk to the virtual machine manager. Then the hypercalls will invoke the virtual machine manager to perform the task on behalf of the guest kernel. In this case, as the guest kernel has the capability to communicate with the virtual machine manager through the hypercalls, the performance of the will be much better than the full virtualization. But, this technique requires a guest kernel which is specialized in knowing about the para virtualization technique and will come with the needed software support.

Hardware assisted virtualization:

Intel and AMD realized that full virtualization and paravirtualization are the major challenges of virtualization on the x86 architecture due to the performance overhead and complexity in designing and maintaining the solution. Intel and AMD independently created new processor extensions of the x86 architecture, called Intel VT-x and AMD-V respectively. On the Itanium architecture, hardware-assisted virtualization is known as VT-I [4]. Hardware assisted virtualization is a platform virtualization designed to use the full virtualization efficiently with the hardware capabilities. There are different names that are used for this technology such as the hardware virtual machine, accelerated virtualization.

For the better support for the virtualization, the intel and the AMD have introduced the virtualization technologies such as the VT and Secure virtual machine (SVM), respectively, as the extensions for the instruction set. These virtualization extensions will help the virtual machine manager or the hypervisor to run as the guest Operating system which is expected to run in the kernel mode. These extensions allow the VMM/hypervisor to run a guest OS that expects to run in kernel mode, in lower privileged rings. Hardware assisted virtualization not only proposes new instructions, but also introduces a new privileged access level, called ring -1, where the hypervisor/VMM can run. Hence, guest virtual machines can run in ring 0. With hardware-assisted virtualization, the operating system has direct access to resources without any emulation or OS modification. The hypervisor or VMM can now run at the newly introduced privilege level, Ring -1, with the guest operating systems running on Ring 0. Also, with hardware assisted virtualization, the VMM/hypervisor is relaxed and needs to perform less work compared to the other techniques mentioned, which reduces the performance overhead [4].

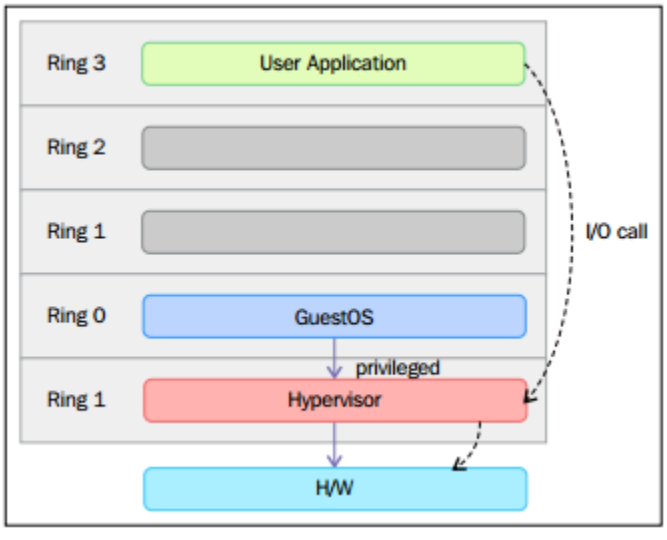


Figure 6. Hardware assisted virtualization architecture.

To put in the easy terms, the virtualization which is aware of the hardware will provide the support of building the Virtual machine manager and make sure that the guest operating system is isolated. This will be helpful in achieving the better performance and tries to avoid the complexity of designing the virtualization solution. In the modern techniques of virtualization, they use this feature for providing the virtualization.

Advantages and Disadvantages of KVM :

- If anyone want to start with KVM as a starter as for the personal use or any other use, it is bit difficult to start with. Unlike the VMware or the virtual box, KVM is the command line tool. The performance of the KVM is better but considering it in terms of the Graphical user interface (GUI) management. KVM makes most sense in terms of the business environment, as it is the command line tool and can provide great productivity and more control in the long run.

- Furthermore, if the CPU used does not support the virtualization, then using the KVM will be of no use and will be running in an extremely slow and the inefficient emulation mode.
- KVM is the good hypervisor to be used if anyone is looking for the open source modern virtualization which is having an unlimited usage mode and no need of paying the extra fees and having to use the powerful command line interface. If there are no virtualization extensions provided to the CPU then it is waste of using the KVM, in that case there are other solutions that can be used such as the VMware server, ESXi or the virtual box.

This paper will initially focus on discussing how to create a virtual machine using the Kernel-based virtualization (KVM), how to take the snapshots of the Virtual Machines created, how to do the Virtual Machine migration. Further, this paper deals with VM security and the issues related to the KVM and how the host can connect to the Virtual Machines which are located remotely using the Secure Shell (SSH) tunnels, using the Simple Authentication and Security Layer (SASL), using the Transport Layer Security (TLS).

Problem Statement

Till now this paper mainly discussed on what is virtualization, and what does the KVM do. As most of the companies are moving to cloud and virtualization is one of the main important thing using the cloud, and accessing these Virtual Machines in the cloud may be vulnerable to some of the malicious attacks. So, in order to prevent this attacks from happening, these Virtual Machines should be accessed securely. Moreover, instead of

accessing the Virtual machine securely, there are some issues with the virtual machine itself. The issues related to the virtual machines are such as the accessing of the file systems on the Virtual machine which is not running i.e. like the rescuing the virtual machine. The other issue might be when there is the sudden shutdown of the virtual machine which will lose the state of the machine.

Nature and Significance of the Problem

As the virtualization is also used for the storage of the memory, then because of the vulnerabilities, there might be a chance of hacking into the virtual machines and getting access to the data stored in the Virtual machines. This might be some sensitive data related to a company which will be big problem. So, there should be a process in order to prevent the information to get hacked.

Objective of the Study

The Objective of this research is to state the problem of how there is an increase in the virtualization in the cloud and with it which brings some vulnerabilities and provide some ways to access the Virtual machines in the cloud securely and making it hard for the people trying misuse the Virtualization purpose.

Study Questions/Hypotheses

There are couple of study question that might be helpful in doing this research such as like, what are the main vulnerabilities or the loopholes while using the virtualization and are there any methodologies previously that might be helpful in implementing the solutions for this purpose.

Definition of Terms

Virtual machine. The virtual machine id the computer program that will create a computer system that is completely virtual and with the complete virtual hardware devices. This computer machine which is completely virtual and runs as the process in the window on the current operating system. When a new operating system is being installed through a disc inside the virtual machine, then the operating system will be tricked in to believe that it is running form the real computer. Then it would install and run as just it would do on any other physical machine. So, whenever you want to use an operating system, then simply open the virtual machine in a window and use it on the real computer as a virtual machine.

QEMU. Quick emulator (QEMU) is an open source machine emulator and a virtualizer. When QEMU is used as emulator operating systems can run operating systems and the applications made for one machine on a different machine. When QEMU is used as a virtualizer, then is can run the guest code on the CPU of the host and can achieve best performance.

KQemu. In the specific case where both source and target are the same architecture (like the common case of x86 on x86), it still has to parse the code to remove any 'privileged instructions' and replace them with context switches. To make it as efficient as possible on x86 Linux, there's a kernel module called KQemu that handles this.

Being a part module, KQemu can execute most code unaltered, supplanting just the least level ring0-just guidelines. All things considered, client space Qemu still distributes all the RAM for the imitated machine and loads the code. The distinction is that as opposed to

recompiling the code, it calls KQemu to filter/fix/execute it. All the fringe equipment imitating is done in Qemu. This is a lot faster than plain Qemu because most code is unchanged, but still has to transform ring0 code (most of the code in the VM's kernel), so performance still suffers.

Emulation. Emulator is the equipment or the product which empowers one PC framework to act like another PC framework. PC frameworks are the host framework and the visitor frameworks.

An Emulator will typically enable the host system which is used to run the software or can be used as the peripheral devices that are designed for the guest operating system. Emulation refers to the ability of the computer program in an electronic device to emulate another device or the program. If we take an example of the printers, they are typically designed for emulating HP LaserJet printers because of the so much software that is written for the HP printers. So, if any Non-HP printer emulates the HP printer, then the software written for the HP printer works for the Non-HP printer.

Virtualization extensions. To improve the performance of the virtualized extension some of the CPU providers offer another type of hardware assisted virtualization technologies which will help in allowing a virtualized environment to access the available CPU resources directly, instead of using the resources through the and helps in increasing the performance of the virtualized systems.

Summary

In this chapter, we discuss about how the increase in the usage of the system resources and to use the resources efficiently led to the virtualization and the technologies

helping in the implementation of the virtualization and how this may have some issues regarding the securities. Eventually, this paper will then discuss about some of the security issues and then try to implement some of the secure ways to access the virtualized systems.

Chapter II: Background and Review of Literature

Introduction

In the previous chapter, mostly the discussion is about the virtualization in the computer systems and deals with how a virtualization is done. In this chapter, the focus shifts on to how the concept of the virtualization came into existence and the discussion about the different virtualization techniques that are available and the background of the techniques and the review of the other works that is done in the field of virtualization and the security of the virtualization.

Background Related to the Problem

The concept of the virtualization is first developed in the 1960's by the IBM Corporation which was intended to do the partition of the large computer into a several number of the smaller instances and run them on the single physical hardware which acts like the host. Then people realized that, with the help of dividing the main system, the number of processes and applications can be run at the same time which in turn increases the efficiency of the infrastructure and allowing to decrease the maintenance. As the development of the virtualization became more advanced it has increased its popularity in the computing world and it has proven to be one of the fundamental building block for today's computing [6].

Then as the virtualization become more important in the computing world then the new technologies have come for the efficient use of the virtualization such as the hypervisors and container-based virtualization. Container-based virtualization will provide a different level of abstraction in terms of the virtualization when compared to the

hypervisors [7]. Hypervisor is like the virtual Machine Manager (VMM) which is the software that will create and run the virtual machines. When a hypervisor is run computer then it is called as the host machine and the remaining virtual machines are called as the guest machines. Hypervisors are of two types such as the type-1 hypervisor and type-2 hypervisor.

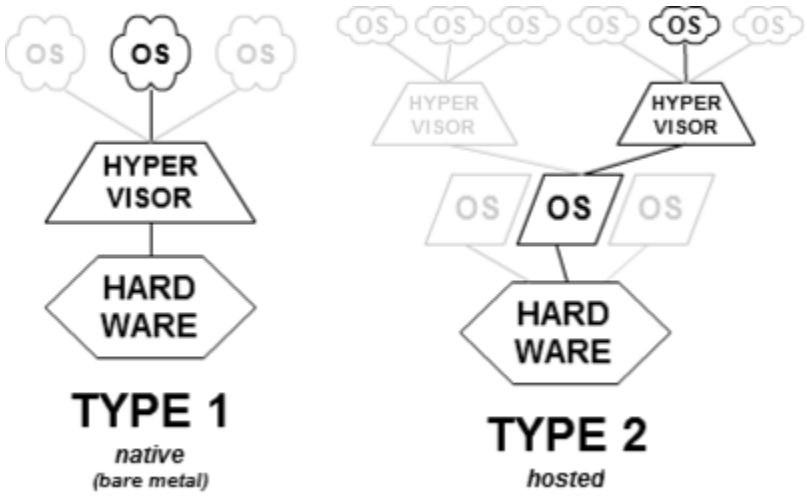


Figure 7. Type 1 and type 2 hypervisor (from www.wikipedia.com).

Hypervisors are divided in to two types such as the type 1 and type 2 hypervisors based on the where they are residing in the system or in the other words, whether the underlying operating systems is present in the system or not. **T**here is no clear or standard definition for the type 1 and type 2 hypervisors. If the Virtual machine manager or the hypervisor runs on top of the hardware directly, then it is called as the type 1 hypervisor. Suppose if there is an operating system and the virtual machine manager or the hypervisor is on top of the operating system and acts separately, then it is called the type 2 hypervisor. Type 1 hypervisor does not need any of the host operating system and interacts with the

system hardware directly. This hypervisor can be directly installed on the bare metal and then make it instantly ready for the hosting of the virtual machine.

Some of the advantages of the type 1 hypervisors are as below:

- It is easy to install and configure.
- It is small, optimized to provide most of the physical resources to the hosted guest such as the virtual machines.
- Generates less overhead, as it comes with only the applications needed to run virtual machines.
- More secure, because problems in one guest system do not affect the other guest systems running on the hypervisor.

But one of the issue with the hypervisor is that it does not support any customization. It is not possible to install any third-party applications or the drivers on top of it.

Type 2 hypervisors will reside on the top of the operating system, which will allow to do a lot of customization as required. This hypervisor is also known as the hosted hypervisors. For performing any operations, the type 2 hypervisor depends on the host operating system. Because of the underlying host operating system is controlling the access to the hardware, type 2 hypervisor used for the wide range of the hardware support.

To decide which hypervisor to be used will mainly depend on the infrastructure of where we are going to deploy the virtualization. But one such assumption among lot of people is that type 1 hypervisor performs better than the type 2 hypervisor as it will operate directly on the top of the hardware.

As from the above figure type 1 hypervisor which is also called the bare metal, runs on top of the hardware and the type 2 will operate on top of the existing operating systems and act as an application. KVM comes under the category of the type 1 hypervisor. As mentioned earlier, there are other virtualization technologies such as XEN hypervisor, Hyper-V, ESXi, and so on. Out of the available virtualization technologies KVM is the most widely used hypervisor.

Literature Related to the Problem

The focus of this study is that managing of the virtual machines and the accessing of these virtual machines securely. As mentioned in the above section for managing the virtual machines there are different virtual machine managers or the hypervisor such as Xen hypervisor, ESX, openVZ, and KML. These all are the hypervisors that are used to make the virtualization easier and out of these virtual machine managers, Kernel based virtual machines is the most used hypervisor. Apart from the hypervisors used for the virtualization, there is other technology for the sake of the virtualization called as the containers such as the Dockers.

Xen/ Citrix XenServer. Xen is the bare metal hypervisor which is the type 1 hypervisor. As the Red Hat Enterprise virtualization used the KVM as the hypervisor for virtualization, Citrix used the Xen as the Hypervisor. Xen will make it possible for the multiple guest OS to run on the single computer by using the software called as the hypervisor for the sake of mediating the access to the real hardware of the system. In the other words the hypervisor will act as the one used for directing the hardware access and coordinating the requests that are coming from the guest operating systems.

XEN virtualization technology is designed to run multiple operating systems on a single server, and provide migration of the running OS instances from one physical server to another. XEN dynamically partitioned the hardware resources of a machine, provides secure partitioning between virtual machines and allows multiple different guest operating system images to be run simultaneously [8]**.** Some of the features that are provided by the Xen hypervisor are as below.

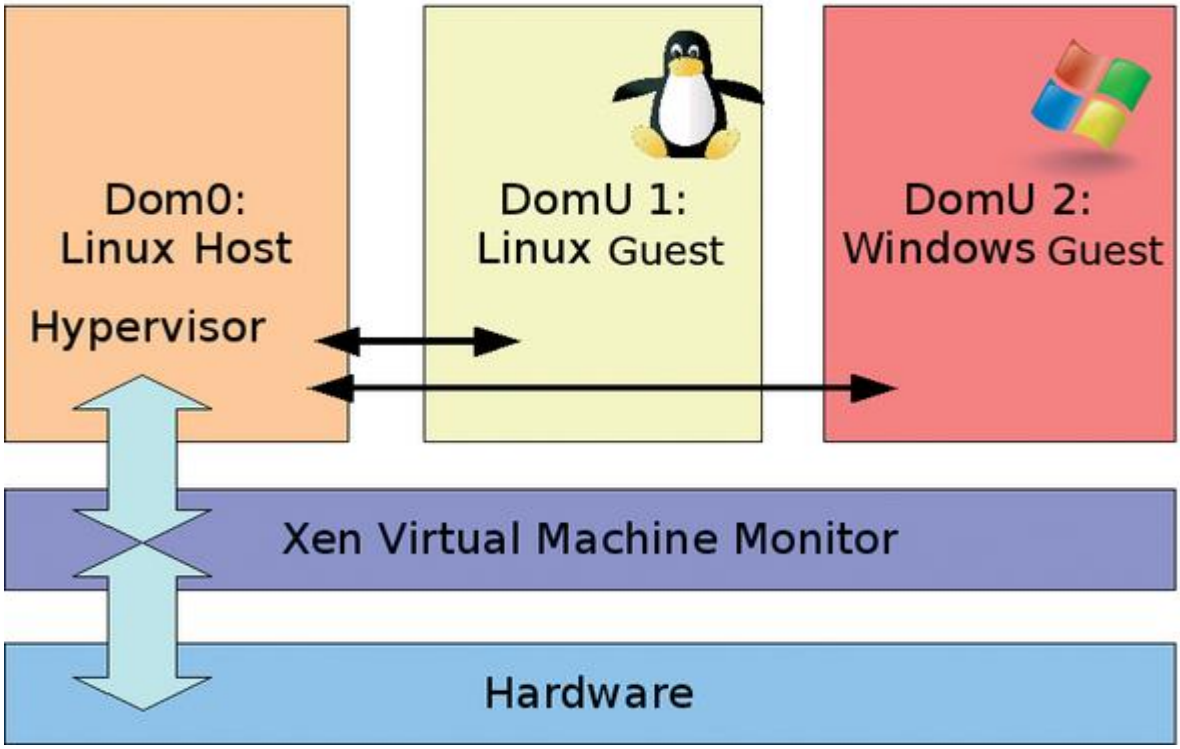


Figure 8. Xen hypervisor.

- Conversion tools
- Integration System Center Virtual Machine Manager
- Snapshot and revert
- Live migration

- Live storage migration
- Distributed virtual switch
- High availability
- Memory optimization

ESXi. The ESX and the ESXi are both the bare metal hypervisors which are developed by the VMware Inc. and these hypervisors can be able to run directly on top of the hardware of the system without the need of any operating system over the hardware. In this virtualization technology, it will only create the kernel and runs it, which will be run after the bootstrapping of the hardware by the Linux kernel. Then the resulting service is a microkernel and has three interfaces.

- Hardware
- Guest system
- Console operating system (service console)

Some of the distinct features of VMware ESXi Server are: The ESX architecture will make use of the service console which is then used for the management tasks which also includes the execution of the scripts and the third-party agent installations. ESXi does not have this service console, which significantly reduces the hypervisor code-base footprint. By removing the service console, ESXi migrates the management functionality from a local command line interface to remote management tools.

ESXi's API-based partner integration model will be able to remove the requirement to install the third-party agents and controlling the agents. This will help in the automation of the routine tasks which are done by writing some automated scripts using the PowerCli.

ESXi Server has a simple security profile configuration when compared to ESX Server.

Hyper-V. Hyper-V is the virtualization technology provided by the Microsoft, which is also known as the windows server virtualization. Hyper-V is a native hypervisor which can create the virtual machines on the systems which are running the windows operating systems. A server computer which is running the Hyper-V can be configured, so that to expose the individual virtual

The features that are offered by the Hyper-V are as follows:

- Live migration
- Storage migration
- VM Replication (Replica)
- Dynamic memory
- Extensible virtual switch
- High availability
- Scale up to 320 logical processors, 4TB of memory, 2,048 virtual CPUs per host, 64 vCPUs per VM, 1TB of memory per VM, and 64 nodes / 8000 VMs per cluster.

In the paper [9] there was a study conducted on the performance of the virtual machines which are running on the Xen hypervisor. This study provides about the CPU performance of the VM's on the different hosts such as the CentOS and Ubuntu. In another paper [8], there is a study on the performance analysis of the three major virtualization technologies used such as Xen, ESXi and Hyper-V. [10] has provided the security of the

hypervisors in the cloud computing and provides the side-channel attacks and defenses, performance-based attacks and about the hypervisor attacks and defenses.

Literature Related to the Methodology

There are many different research papers on the Kernel-based virtual machines (KVM) which are focused on the understanding of the security issues related to the virtualization and about the development of the architectures for the securing of the virtual machine. The virtual machines can be migrated live [11], where they discussed about the securities related to Virtual Machine while they are migrated to the other environment. They discussed about the issues in this process such as the attacks done while it is still migrating and the other issue with this is that when there is a problem and suddenly the connection between the two machines might be lost and the state of the machine changes. [12] discussed about the security management architectures for the protection of the Kernel virtual machines. This study provided an architecture to analyze the data related to the guest and monitor the behavior of the guest system.

There is a paper [13] which discussed about the managing of the virtual machines using the virtual machine manager and provided models and the architecture of the Virtual machine manager in maintaining the VM's. Later on in the paper they provided the factors related to the virtual machines such as the security, memory management, hardware support and guest support.

Summary

This chapter mainly focused on the background related to the study and explains about how the virtualization has been evolved since its usage since the 1960s and provided on the

information about types of hypervisors. Then provided about some of the previous works that are done in the problem related to the study.

Chapter III: Methodology

Introduction

In this chapter, the implementation of the virtualization using the KVM is explained further. This section still provides the details of the tools such as, what kind of the platform is used to implement and how is that going to be implemented by analyzing the studies which were done previously related to this study.

Design of the Study

The underlying reason for this study is the security of the accessing Virtual machines remotely. While considering the security of the virtual machine, there will be many reasons and causes that are contributing to the comprising of the virtual machine. In order to have a better securing access to these machines, there should be an understanding of the underlying reasons for this.

In this study, we follow the qualitative approach. With the help of this approach, we can gain the understanding of the underlying reasons for using the Kernel based virtual machine (KVM) over that of the other hypervisors such as the XEN hypervisor or the ESXi hypervisor. Below is the figure that show the components that will maintain the full hardware virtualization.

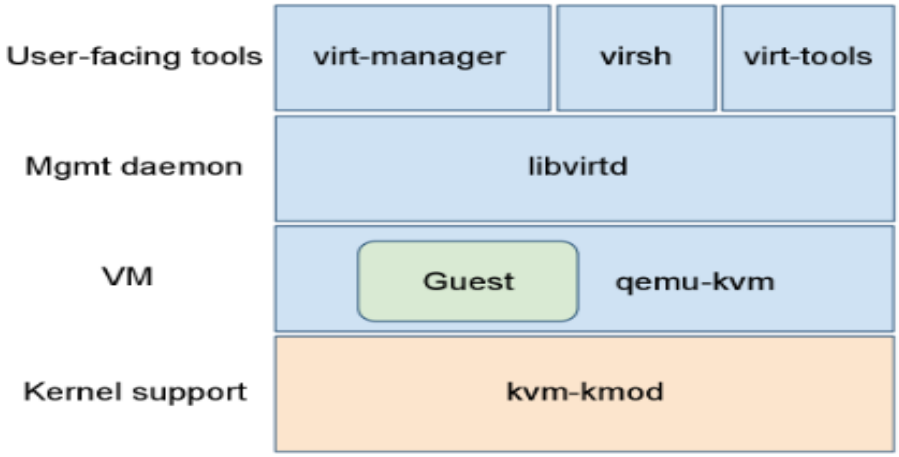


Figure 9. Full hardware virtualization (from IBM developer works).

Kvm-kmod. Kernel-based virtual machine kernel module (Kvm-kmod) is the one which provides the kernel support and consists of a kernel model ‘kvm.ko’ which will provide the virtualization of the core infrastructure and also a processor specific module called as the ‘kvm-intel.ko’ or the kvm-amd.ko’ based on the type of the virtualization extensions.

Qemu-Kvm. Qemu is called as the quick emulator which is used for the hardware virtualization. Qemu is like the hosted virtual machine monitor which will emulate the CPUs through the dynamic binary translation and then provide a set of the device models, which will allow to run different variety of the guest operating systems. Here the QEMU deals with the setting up and migration of the KVM images. Here the execution of the guest is done by the KVM as requested by the QEMU.

QEMU has different operating modes which are discussed below:

User-mode emulation: In this mode QEMU runs single Linux or Darwin/macOS programs that were ordered for an alternate guideline set. Framework calls are clunked for endianness and for 32/64 bit befuddles. Quick cross-arrangement and cross-troubleshooting are the primary focuses for client mode copying.

System emulation: In this mode QEMU imitates a full PC framework, including peripherals. It can be utilized to give virtual facilitating of a few virtual PCs on a solitary PC. QEMU can boot numerous visitor working frameworks, including Linux, Solaris, Microsoft Windows, DOS, and BSD; [4] it bolsters copying a few direction sets, including x86, MIPS, 32-bit ARMv7, ARMv8.

KVM hosting: Here QEMU deals with the setting up and migration of KVM images. It is still involved in the emulation of hardware, but the execution of the guest is done by KVM as requested by QEMU.

Xen hosting: QEMU is included just in the imitating of equipment; the execution of the visitor is done inside Xen and is completely escaped QEMU.

Libvirt: Libvirt is the collection of the software which will provide the convenient way to manage the virtual machines and virtualization functionality of the machines, such as like the storage and the network interface management. The software that are combined to form the libvirt are the application program interface (API), a daemon called as the libvirtd and a command line utility called as Virsh. The important goal of the libvirt is that to provide single way to manage the multiple different virtualization providers or the hypervisors such as the KVM/QEMU, Xen and others. The major features of the libvirt are as shown below:

- VM Management
- Remote machine support
- Storage management
- Network interface management
- Virtual NAT and route based networking

There is a thinking among lot of people that the libvirt is only restricted to the single node or the local node where it is running and this is not true. There is a remote support built into the libvirt library. KVM is deployed as a kernel-based bare OS module that can be loaded to extend the Linux OS by this proficiency. Thus, Linux OS is turned into a VMM. In a regular Linux OS virtual environment every process executes in one of the two modes, either user-mode or kernel-mode [14].

As discussed in the earlier part, the main goal of the libvirt is to produce an environment that is stable and common for managing of the virtual machines which are running on the hypervisor. In the other words, this can be thought as the management layer which is responsible for providing the Application program interface (API) which will be able to do the tasks such as the virtual machine creation, provision, Monitoring, modification, migration, control and so on. There will be some of the processes that are demonized in the Linux and the libvirt is also demonized and this is called as the libvirtd. As like any other daemon processes, libvirtd will provide the services to the clients up on their requests. Let us try to understand what exactly happens when a libvirt client such as virsh or virt-manager requests a service from libvirtd. Based on the connection URI passed by the client, libvirtd opens a connection to the hypervisor. This is how the clients' virsh or virt-manager ask the

libvirtd to start talking to the hypervisor. In the scope of this book, we are aiming at KVM virtualization technology. So, it would be better to think about it in terms of a QEMU/KVM hypervisor instead of discussing some other hypervisor communication from libvirtd. You may be a bit confused when you see QEMU/KVM as the underlying hypervisor name instead of either QEMU or KVM. The sVirt and SELinux [15] present an infrastructure model which presents a degree of isolation and security unrivalled in the industry [16].

Virt-manager. Virt-manager is the graphical user interface for the view of the different hypervisors and all the guests on the host system and also on the remote host system. The Virt-manager will provide the Virtualization management task such as the defining the guests, assigning the virtual CPUs, saving and restoring such as taking the screenshot of the state of the machine at particular time so that if there is any problem with the virtual machine and it shutsdown, then this will help in maintain the status of the machine, check the operation performance of the virtual machines, starting and shutting down of the virtual machine and other functions. Virt-manager can be started form remotely using the SSH using the following command.

ssh -x host_address

Virsh. Virsh is the command line interface (CLI) tool which is used for managing the guests and the hypervisor. The Virsh tool is built on the libvirt management API and will act as an alternative tool for the xm tool and the Virt-manager which is the graphical interface. The Virsh tool can be used for loading the scripts for the guest machines. Below are the some of the commands that are used for connection to the hypervisor and creating

the virtual machine. For initiating the hypervisor session with the Virsh tool the following command will be used. Virsh connect <name>

Where the name is the machine name of the hypervisor. We can append some commands to the above command to be secure such as if we want to make it be read-only then we have to append the -readonly to the command.

Data Collection

This paper deals with the managing of the virtual machines and the secure access of the virtual machines. So mainly the focus will be on how to implement the accessing of the VM's securely. So, there is no data that I am using in this study but I have referred the articles that are published whose focus is on the KVM security for the understanding.

Tools and Techniques

For the implementing the secure access through the remote management, here the techniques such as the Secure Shell (SSH) tunnels, Simple Authentication and Security Layer (SASL) authentication and the Transport layer security (TLS) are used. Moreover, for the management of the Virtual machines, here we use the Virt-manager.

Hardware Requirements

- PROCESSOR–Intel/AMD
- RAM–2GB
- DISK SPACE–40GB
- OPERATING SYSTEM–Red Hat / CentOS

Software Requirements

- KVM hypervisor
- Virt-manager
- Virsh

Chapter IV: Implementation

Introduction

When we talk about the virtualization, there will be two types of the systems such as the Host system and the Guest system. When the host system is compromised with security, then the guest system will also be affected. So, here we will be briefly discussing about the host security of the system and recommended practices for securing the host machine of the RHEL Linux and CentOS Linux. Then we also briefly talk about the Guest System security and the recommended practices for securing the Guest system in the RHEL Linux and the CentOS Linux.

Security of host system. If we take a RedHat Enterprise Linux, then the RHEL host system is the one which take main responsibility for making sure that the controlling and providing access to the physical devices, guest OS systems, network and the storage of the system. So, if we don't secure the RHEL, then it means entire storage, physical devices, and the guest OS systems are also compromised. So, securing the Host system is the initial step that can be taken in keeping the virtualized environment safe.

As the Host Machine plays a pivotal role in securing the virtualized environments, we discuss some of the best practices that are to be recommended for securing the RedHat Enterprise Linux Host System.

Accessing the Host Machine is limited to only some users who are in real need of using the resources of the system, not all people are provided with access to the machine. Users are not given the root access instead they are given the access the administrators using the "sudo" command and provide the privileges to the users with certain administrative roles.

Run just the administrations important to help the utilization and administration of your visitor frameworks. If you must give extra administrations, for example, document or print administrations, you ought to consider running those administrations on a Red Hat Enterprise Linux visitor.

SELinux should be configured well and it should be operating in the enforcing mode. The sVirt which provides the advanced virtualization security relies on the SELinux.

We must make sure that the remote management of the system or the machine should take place only through the SSH tunnel and also using the network protocols such as the SSL and TLS, both of which provide the data encryption and the authentication.

Also making sure that the firewall is configured properly during the installation of the Host system and making sure that is up and activated during the boot. Only certain ports which are used in the management of the system are allowed.

Security of the guest machine. As the Host machine is secure which plays a key role in protecting the virtualized environments, but this does not completely assure that the guest machine is also secured. Whatever the security risks that are bothering the non-virtualized systems, the same security risks can bother the security of the virtualized systems. So, we must make sure that the virtualized environments are also secure from the attacks.

With all administration of the visitor likely occurring remotely, guarantee that the administration of the framework happens just finished secured arrange channels. Devices, for example, SSH and system conventions, for example, TLS or SSL give both verification and information encryption to guarantee that exclusive affirmed overseers can deal with the framework remotely.

Some virtualization advances utilize unique visitor operators or drivers to empower some virtualization highlights. Guarantee that these specialists and applications are secured utilizing the standard Red Hat Enterprise Linux security highlights, for example, SELinux.

In virtualized situations there is a more serious danger of touchy information being gotten to outside the assurance limits of the visitor framework. Secure put away touchy information utilizing encryption devices, for example, dm-tomb and GnuPG; albeit uncommon care should be taken to guarantee the secrecy of the encryption keys.

Kernel based virtual machine (KVM). Kernel based virtual machine(KVM) is a technique for achieving full virtualization solution for the Linux machines which are on the x86 hardware which is containing the virtualization extensions such as the intel VT or the AMD-V. KVM consists a loadable kernel module kvm.ko which will provide the core virtualization infrastructure along with the processor specific module kvm-intel.ko for the machines that support intel-VT virtualization Extension and kvm-amd.ko for the machines that support AMD-V virtualization extension.

KVM has turned out to be a standout amongst the most broadly utilized virtualization advances today, and it has gone up against a wide range of structures by organizations or associations that have adjusted the code, including IBM and Red Hat. It is an open-source alternative to restrictive virtualization innovations, for example, ESXi offered by VMware and Microsoft. Other open-source virtualization arrangements incorporate Xen, which is bolstered by Citrix.

Here for doing the implementation, I am using the VMWare Workstation to install another software on my machine. I have installed CentOS 7 operating system on the VMWare workstation instead of RedHat Enterprise Linux (RHEL), because the RHEL is licensed version for which I have to pay certain amount of money, instead went for CentOS because it is an open source software of the RHEL version.

Preparing the Environment.

Hardware. We must make sure that the processor with which the host machine is running on, should support the Virtualization extensions of either the Intel or the AMD. For checking whether the processor is supporting the virtualization the following command is used.

```
[root@localhost /]# grep -E 'svm|vmx' /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse
36 clflush dts mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc arch_perf
mon pebs bts nopl xtopology tsc_reliable nonstop_tsc aperfmperf eagerfpu pni pclmulqdq
ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx
f16c rdrand hypervisor lahf_lm abm 3dnowprefetch epb fsgsbase tsc_adjust bmi1 hle avx2
smep bmi2 invpcid rtm rdseed adx smap xsaveopt dtherm ida arat pln pts hwp hwp_notify h
wp_act_window hwp_epp
```

Figure 10. Checking whether processor supports VT.

If for the execution of the above command nothing shows up, then the processor doesn’t support the virtualization.

Then we must install the required packages which are not installed with the CentOS 7 Operating system. Here, we install the packages which also includes the installation of the KVM also.

```
[root@localhost /]#  yum install qemu-kvm libvirt libvirt-python libguestfs-tool
s virt-install
```

Figure 11. Installing required packages.

The installed packages include the KVM, virt-install, libvirt-python, libguestfs-tool and libvirt.

Virt-install. Virt-install is used for the creation or the installation of the guests from the command line by specifying the required memory, RAM, and the installation source of the guest virtual machine.

Qemu-kvm. Qemu is called as the quick emulator which is used for the hardware virtualization. Qemu is like the hosted virtual machine monitor which will emulate the CPUs through the dynamic binary translation and then provide a set of the device models, which will allow to run different variety of the guest operating systems. Here the QEMU deals with the setting up and migration of the KVM images. Here the execution of the guest is done by the KVM as requested by the QEMU.

Libvirt. Libvirt is the collection of the software which will provide the convenient way to manage the virtual machines and virtualization functionality of the machines, such as like the storage and the network interface management. The software that are combined to form the libvirt are the application program interface (API), a daemon called as the libvirtd and a command line utility called as Virsh. The important goal of the libvirt is that to provide single way to manage the multiple different virtualization providers or the hypervisors such as the KVM/QEMU, Xen and others. The major features of the libvirt are as shown below:

- VM Management
- Remote machine support
- Storage management
- Network interface management
- Virtual NAT and route based networking

Libguestfs-tool. Libguestfs is a set of tools for accessing and modifying virtual machine (VM) disk images. You can use this for viewing and editing files inside guests, scripting changes to VMs, creating guests, performing backups, cloning VMs, building VMs, formatting disks, resizing disks, and much more.

At the last, we must make sure that the KVM is loaded properly and its modules. If the KVM is not loaded properly then, we msut load it manually using the “modprobe” command.

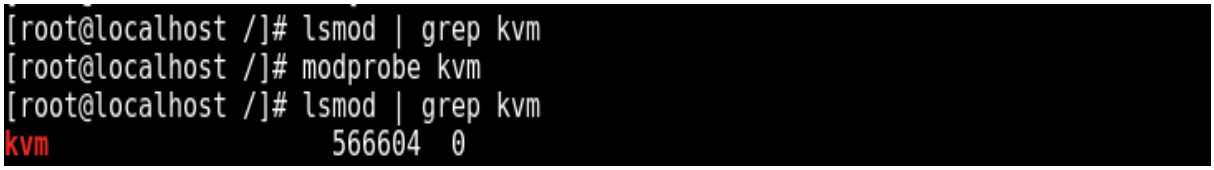


Figure 12. Checking KVM loaded properly.

In the above screenshot, when we tried to see whether the KVM is loaded properly we did not get any output. So, the LVM is loaded manually using the command called “modprobe” and then again tried to load the KVM and this time it is loaded successfully and is shown in the output. By this we have done the installation of the KVM on the host machine which will support the creation of the virtual machines on the host.

The OS that is needed for the creation of the virtual machines can either be downloaded to the host machine or it can be accessed using the https and grab it form the internet directly. Here I have preferred to download it from the internet and then use for the creation of the virtual machines.

Creation of the bridge. By default, the virtual machines will only have the network access to the other virtual machines which are present on the server if you are creating the

virtual machines from the server or the host machine via the private network. If we want our Virtual Machine to use your LAN, then we must create a bridge for the network access. For creating a bridge, the following steps are needed to be followed.

```
[root@localhost network-scripts]# vi ifcfg-ens33
[root@localhost network-scripts]# cat ifcfg-ens33
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=347373f4-fc8c-4efc-b472-0cb37a8124b8
DEVICE=ens33
ONBOOT=yes
BRIDGE=br0
[root@localhost network-scripts]#
```

Figure 13. Creating a bridge.

In the above screenshot we have edited the host network ifcfg-ens33 and added a bridge called br0 using BRIDGE=br0. Then we have opened the ifcfg-ens33 for seeing the contents of the file. After this step we must edit the ifcfg-br0 file which is the bridge file and add the certain lines as shown in the screenshot below.

```
[root@localhost network-scripts]# vi ifcfg-br0
[root@localhost network-scripts]# cat ifcfg-br0
DEVICE="br0"
# BOOTPROTO is up to you. If you prefer “static”, you will need to
# specify the IP address, netmask, gateway and DNS information.
BOOTPROTO="dhcp"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
ONBOOT="yes"
TYPE="Bridge"
DELAY="0"
[root@localhost network-scripts]#
```

Figure 14. Editing the bridge file.

Then we have to edit the “sysctl.conf” file in the /etc folder and make sure the IP forwarding is enabled in the configuration file which as shown below in the screenshot.

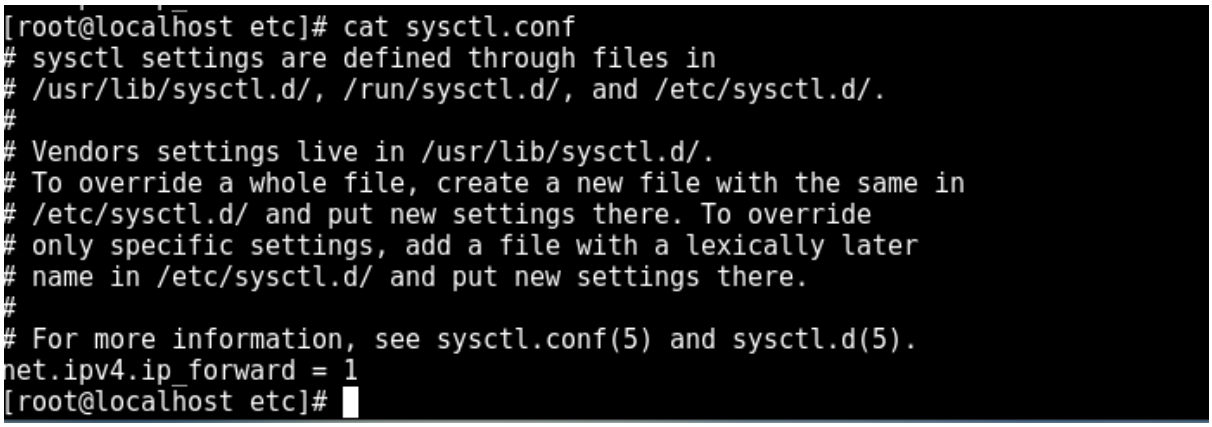


Figure 15. Enabling the IP forwarding.

Then restart the Network Manager to apply these changes.

Figure 16. Restarting the Network Manager

We have installed the KVM and the related packages that are required for the installation of the virtual machine and for managing it. Then created the network bridge “bro” for accessing the LAN. Now we must create the virtual machine which will be discussed in the next section.

Creation of Virtual Machines (VMs)

For the creation of the virtual machines we have a command called as the “virt-install” and along with the command we specify, how the Virtual Machine contacts another VM by providing the virtual bridge, we give the name of the virtual machine, size of the RAM that the Virtual machine needed, number of virtual CPUs, the path that the virtual

machine should be installed, and providing whether the graphics are needed or not, then from where the virtual machine must be installed by providing its location. Below is the screenshot for creating the virtual machine.

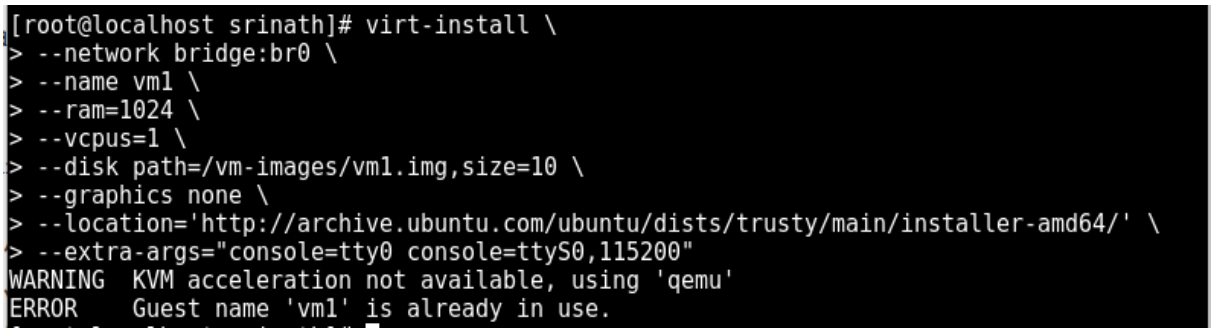


Figure 17. Creating the virtual machine.

We specified the name of the virtual machine as the “vm1” with the RAM of 1024 MB, the number of the virtual CPUs as 1 and the path to be installed at and graphics as none which means that there will be nothing except the Command line.

The additional args parameter is utilized to pass bit boot parameters to the OS installer. For this situation, since we are associating with the VM‟s serial port, we should utilize the best possible part parameters to set it up, much the same as we would on any server, virtual or not.

Cloning the Virtual Machines (VMs)

If you need a few VMs with a similar OS and same setup, I prescribe cloning existing VMs as opposed to introducing the OS on everyone, which can rapidly turn into a tedious and dreary undertaking. In this illustration, we clone vm1 to make another VM clone called vm1-clone.

For cloning a Virtual machine, we must make sure that the virtual machine is in the suspended mode. This can be achieved as shown in the screenshot below:

```
[root@localhost srinath]# virsh suspend vm1
Domain vm1 suspended
```

Figure 18. Suspending the virtual machine.

Then we must run the “virt-clone” command for cloning the virtual machine, and while cloning the Virtual machine we can specify which VM should be cloned and what will be the name of the cloning VM and where the Clone image is stored.

```
[root@localhost srinath]# virt-clone \
> --connect qemu:///system \
> --original vm1 \
> --name vm1-clone \
> --file /vm-images/vm1-clone.img
Allocating 'vm1-clone.img'                                |  10 GB  00:00:00

Clone 'vm1-clone' created successfully.
[root@localhost srinath]# virsh resume vm1
Domain vm1 resumed

[root@localhost srinath]# virsh start vm1-clone
Domain vm1-clone started

[root@localhost srinath]#
```

Figure 19. Cloning the VM and resuming.

As shown in the above screenshot, the cloning of the VM is created with the name of vm1-clone. Then we can check the vm1-clone by starting the vm1-clone using the command Virsh.

Managing the Virtual Machines (VMs)

With some commands in our hand we can manage the Virtual Machines such as starting the virtual machine, stopping or shut downing the virtual machine and accessing

the virtual machine information and deleting the Virtual machines. Below are some of the commands that are used for managing the virtual machines.

```
[root@localhost srinath]# virsh list --all
 Id    Name                           State
----------------------------------------------------
 1     vm1                            running
 2     vm1-clone                      running

[root@localhost srinath]#
```

Figure 20. Listing all the VM's.

Virsh list–all command will give all the Virtual machines that are installed in the system. Here we have created a VM called VM1 and cloned it and named it as VM1-clone. Both the Virtual machines are up and running.

```
[root@localhost srinath]# virsh dominfo vm1
Id:             1
Name:           vm1
UUID:           53e7a8d6-9e7c-439d-9801-f518b21eeebe
OS Type:        hvm
State:          running
CPU(s):         1
CPU time:       1.4s
Max memory:     1048576 KiB
Used memory:    1048576 KiB
Persistent:     yes
Autostart:      disable
Managed save:   no
Security model: selinux
Security DOI:   0
Security label: system_u:system_r:svirt_tcg_t:s0:c457,c962 (enforcing)

[root@localhost srinath]#
```

Figure 21. Checking the domain info.

For checking the information of the virtual machine, we run the command "Virsh dominfo vm1" and the information regarding the VM will pop up.

There is also a tool called as the “Virtual Machine Manage” which is used for the managing the VMs using the GUI console, instead of creating it through the command line using the commands.

On the CentOS desktop, go to the applications folder and click on the system tools and there we can find the virtual machine manager which is built in into the software and does not require any software installation. Right click on the VMM and click on the new, then we get the VMM console as below.

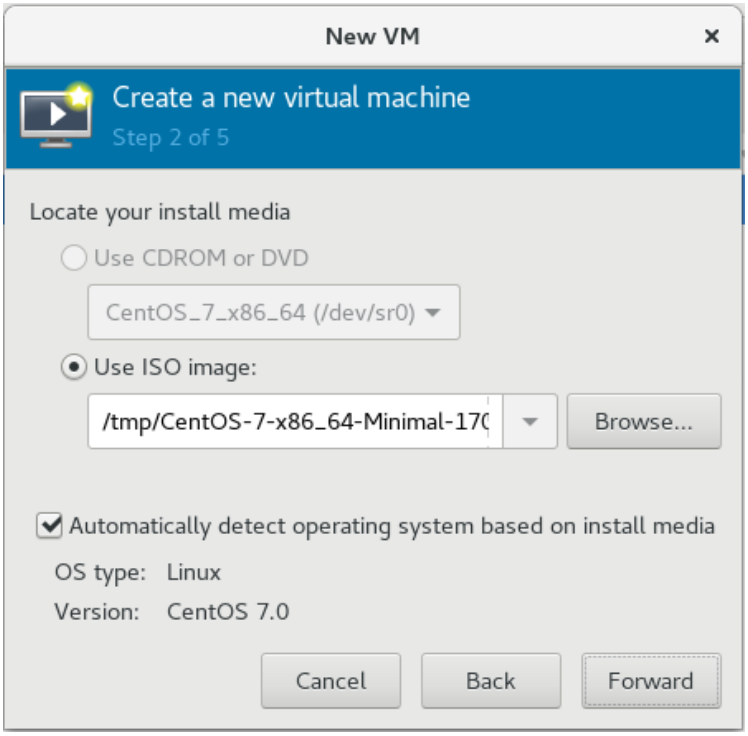


Figure 22. Creating a VM using VMM.

There will be two options for installing the virtual machine software, using the CDROM or DVD where the installation file is present in the system. And the other way us to download an ISO image of the installation software and keep it in any folder in the

system. In our case, I have downloaded the ISO mirror image file and then stored it in the /tmp folder. Give the address and click forward.

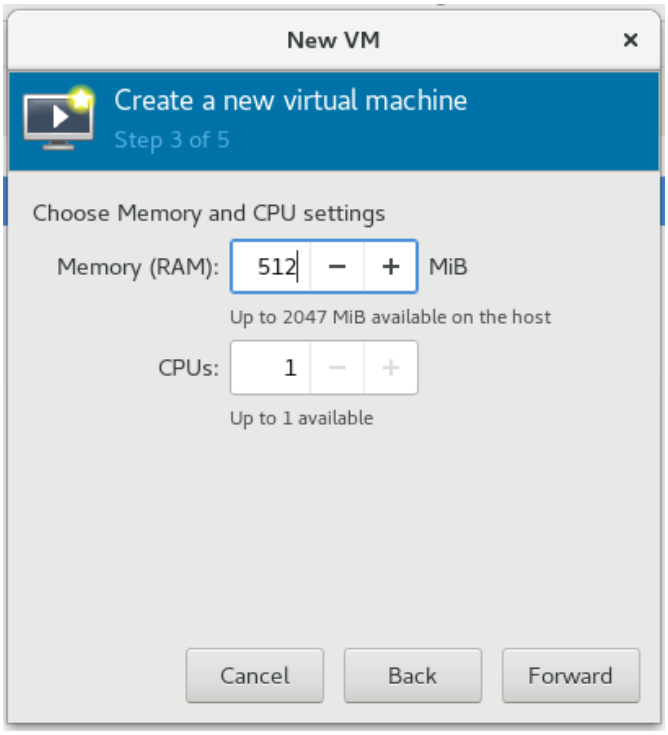


Figure 23. Adding memory and CPUs to VM.

Then we must allocate the Memory for the virtual machine and the CPUs. Here I have given 512 MB of the Memory (RAM) and gave just 1 CPU. Then proceed to the next step.

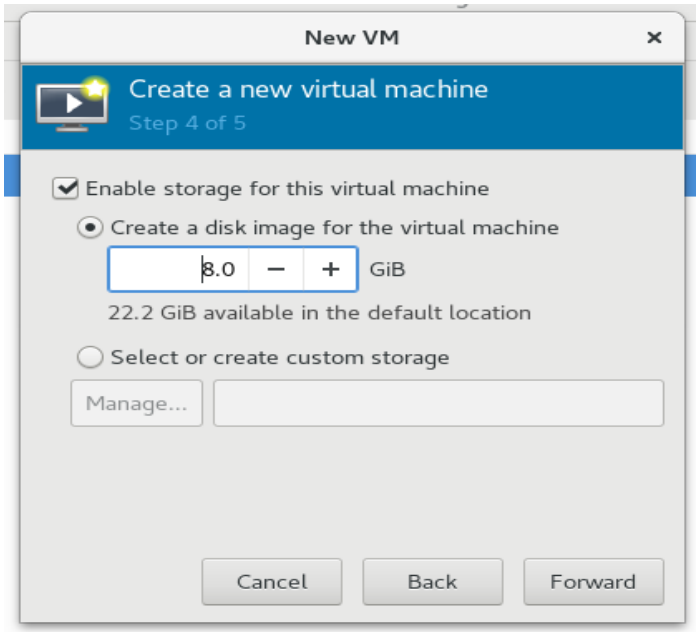


Figure 24. Allocating the storage to the VM.

Then in the next step, we must provide the storage memory for the virtual machine and I have given 8GB of the memory.

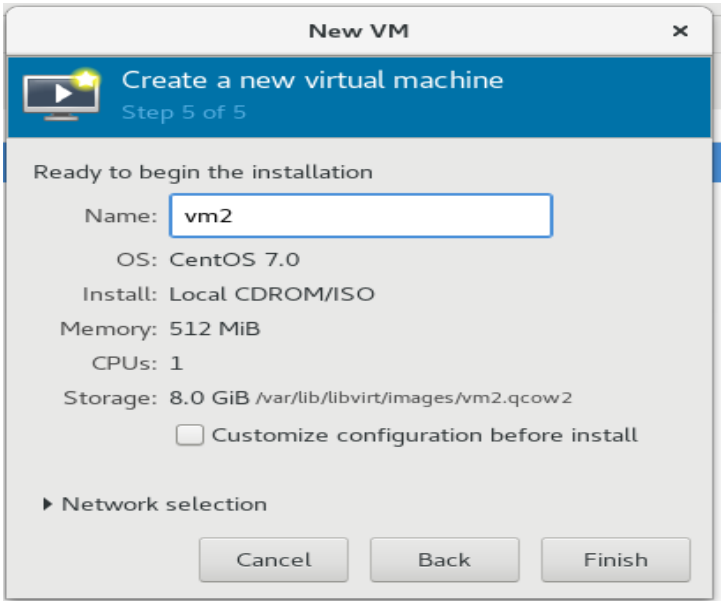


Figure 25. Naming the VM and installing.

In the next step we must provide the name for the virtual machine and review the inputs given and then click finish. Then it will install a new virtual machine with the name vm2 on the host machine.

Deleting the Virtual Machines(VMs)

When you no longer need a VM, it is best practice to remove it to free up its resources. A VM that is shutdown is not taking up a VCPU or memory, but its image file is still taking up disk space. There are three commands for deleting the virtual machines which are as shown in the screenshots below.

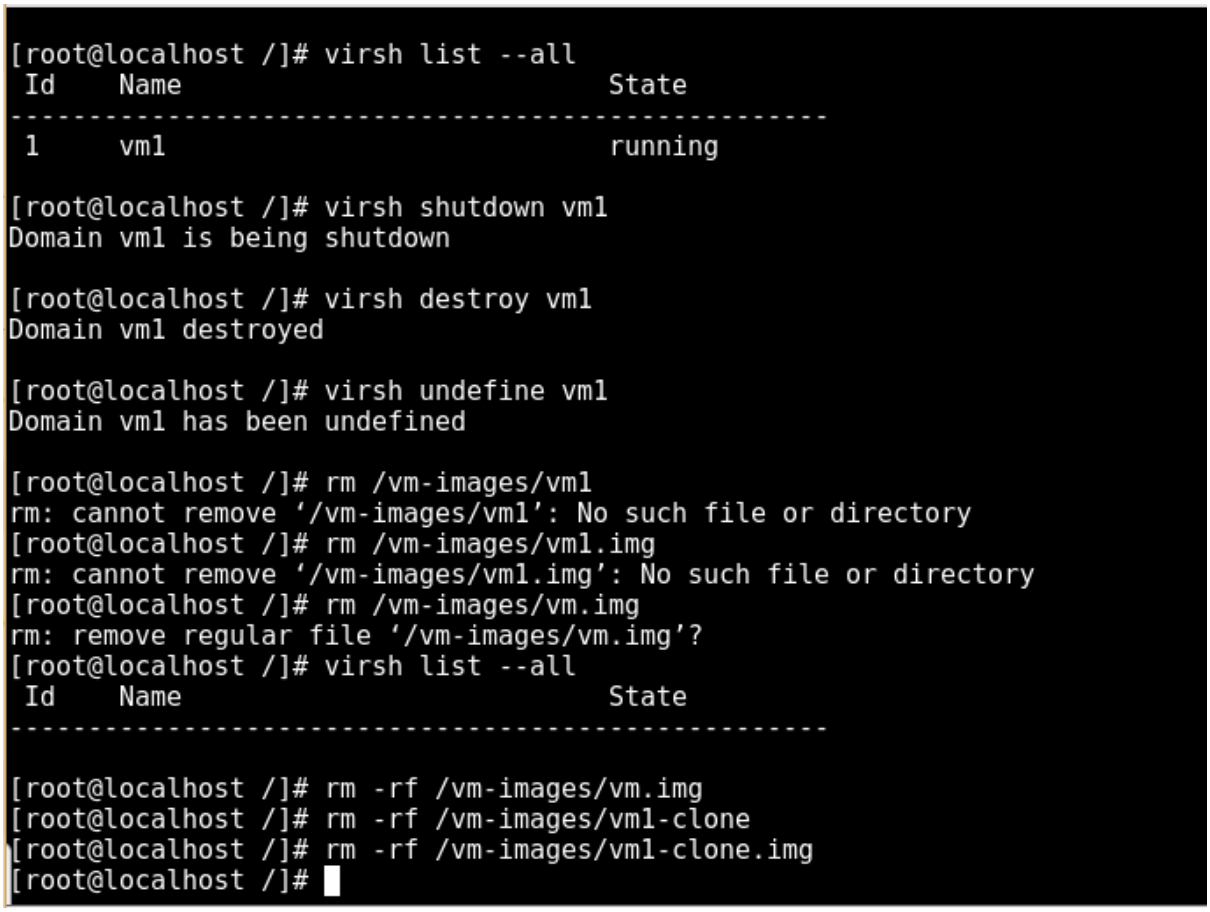


Figure 26. Deleting the VM.

Virtual Machine Snapshots

Snapshots will take the state of the virtual machine that will include the memory of the VM, Disk at any specific moment of the time. Suppose, if we had done a vulnerable step or any command which is not supposed to be executed, then we can go back to the snapshots and can acquire the state of the VM which is not affected. Snapshots can be taken while the virtual machine is running or not, but it is always recommended to take the snapshots while the virtual machine is in the shutdown mode. The snapshots can be identified with a unique number.

Before we go through how to take a snapshot of the virtual machine, we need to know about the disk formats, because the snapshots using the libvirt can only be performed on some of the disk formats. There are many types of the disk formats, but the main disk formats are the raw and the qcow1 which means copy -on-write. While the qcow1 disk format supports variant range of the features that include the encryption, snapshots and the compression.

```
[root@localhost vm-images]# file /vm-images/vm.img
/vm-images/vm.img: QEMU QCOW Image (v3), 10737418240 bytes
[root@localhost vm-images]# qemu-img info /vm-images/vm.img
image: /vm-images/vm.img
file format: qcow2
virtual size: 10G (10737418240 bytes)
disk size: 2.2M
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: true
[root@localhost vm-images]#
```

Figure 27. Checking the format of the disk.

By using the command file and the passing in the location of the virtual machine image path, we can get the format of the disk and its version. Form the above screenshot we can confirm that the disk format is qcow2 and it can support the Screenshots.

Creating the snapshots. There is a command called “snapshot-list” which will provide the list of all the available snapshots of the virtual machine.

```
[root@localhost vm-images]# virsh snapshot-list vm1
 Name                 Creation Time             State
------------------------------------------------------------

[root@localhost vm-images]#
```

Now when we pass the above command we can see that there are no available snapshots for the virtual machine vm1.

```
[root@localhost /]# virsh snapshot-list vm1
 Name                 Creation Time             State
------------------------------------------------------------

[root@localhost /]# virsh shutdown vm1
Domain vm1 is being shutdown

[root@localhost /]# virsh snapshot-create vm1
Domain snapshot 1508968630 created
[root@localhost /]#
```

Figure 28. Shutting down the VM and creating a snapshot.

As discussed in the previous section creating a snapshot will be clean when the Virtual machine is in the shutdown mode, so we first shutdown the VM and then try to create

the snapshot using he command snapshot-create. It will show the output as Domain snapshot created with an unique identification number.

Now when we give the command to provide the available snapshots, we get the list of all the snapshots. As now we have created only one snapshot we get the one snapshot in the list.

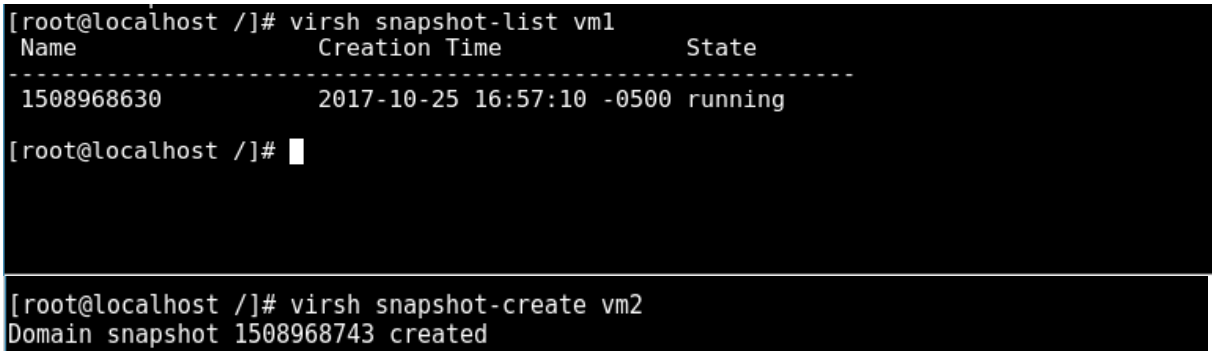


Figure 29. Listing all the snapshots.

Now I have created another snapshot which is of the machine VM2.

Restoring snapshots. While restoring the Virtual machine state using the snapshots, we have to shut down the VM and then revert back to the original state. The command used for getting the state from the snapshot is as below:

virsh snapshot-revert vm2 1508968743

Deleting a snapshot.

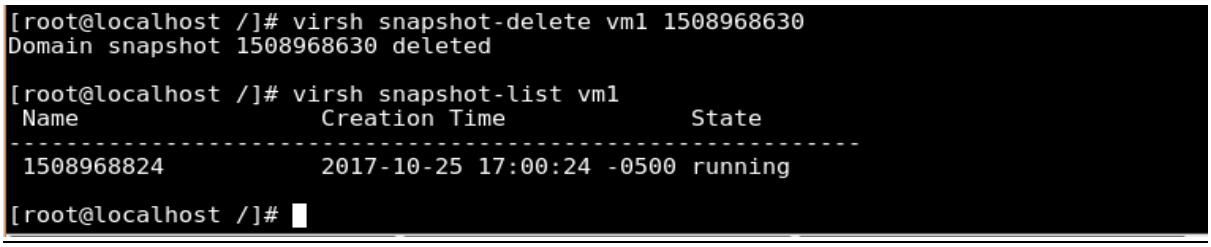


Figure 30. Deleting the snapshot.

The command used for deleting a snapshot is “snapshot-delete” and providing it with the name of the virtual machine and the unique number with which the state of the snapshot is identified.

Remote Management

There are multiple software bundles available which are open source, similarly KVM-based virtualization also supports the standard interfaces, flexibility and modularity where a single functionality of the system is separated and is maintained is different packages. But this modularity provides the security concerns for the system and the virtual machines that are present on the host system. In this section, we discuss about securing kernel based virtual machines (KVM) and at the same time having the advantage of using the small and isolated interconnected packages. The management of the kernel based virtual machine gusts cannot be accessed or managed remotely. This is the default function of the KVM. This happens because the libvirtd daemon will not create any of the listening sockets to have a connection from externally, except for the connection to the local host on which the guests are running. But there are different solutions for accessing the KVM guests remotely using different techniques which are discussed in this section.

The different solutions for accessing the kernel based virtual machines remotely are as below:

- using the Simple Authentication and Security Layer (SASL) authentication and encryption for the Remote Management
- using the Transport Layer Security (TLS) for Remote Management
- Remote Management using the Secure shell (SSH)

SASL remote management. SASL is the one which provides with the data encryption and the secure authentication and at the same time allowing the integration of the authorizing and authentication services which are either internally or externally. If we describe the SASL in the simplest way, it can be used as the database for authorization of the credentials of the user or the server or client. It can also be used in the complex situations where it can also work with the external authentication services such as the LDAP which is used for the authorization of the users and Kerberos. Whatever be the case, the libvirtd daemon will guarantee the confidentiality by adding the GSSAPI for the SASL authentication if the methods for the remote management are not running on top of the secured layer of TLS. Here both the SASL and GSSAPI are the frameworks that are used for the authentication of the users or servers and clients. GSSAPI is an application interface definition for plugins that support the different authentication mechanisms. SASL can use the GSSAPI for extending the authentication mechanisms of itself. There are different methods that are used for the authentication mechanisms such as the GSSAPI and the DIGEST-MD5method. But the thing is that, using the MD5 is considered as unsafe, but for the simplicity here we use the MD5 encryption as the SASL method.

As discussed above SASL is a framework but not a protocol and it can be used in the number of the protocols such as the simple mail transfer protocol (SMTP) or the LDAP. If a protocol is using the SASL, then there will be a specification provided by the internet standard on how to use the SASL for that protocol. So the SASL can be used on wide range of protocols.

The basic functioning of the SASL is bit straight. The server will provide the supported authentication mechanisms and the client will respond back with which authentication mechanism will be used based on its capability.

For configuring the system for the remote management by using the SASL, we take a simple scenario where the external authentication is not required or the TLS security. We must follow the following steps to configure the SASL for the remote management.

First, we have to login to our KVM host and open the terminal and run the commands as the root as we must change the configuration files which can only be done as the root. Then we must edit the libvirtd.conf file. Here I am using the “vi” editor for editing the files. So, type in the command to edit the libvirtd.conf file.

Vi /etc/libvirt/libvirtd.conf

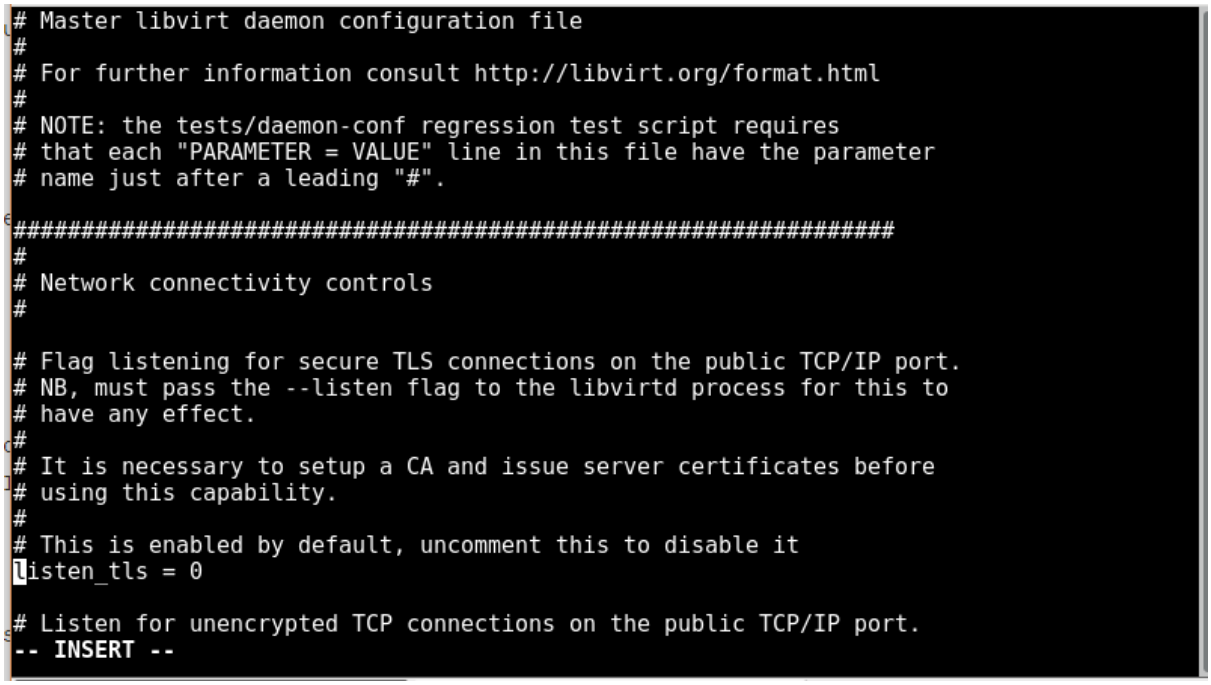


Figure 31. Editing the libvirtd.conf.

In the above screen shot I have disabled the “listen_tls=0”, this is initially set to enable by default. We must disable this because there are no Certificates of the TLS are configured. So, if we keep it enabled it will look for the tls certificate and if it is not there then the libvirtd daemon will not start.

Figure 32. Enabling the listen_tcp in libvirtd.conf.

Here, we must make sure that the “listen_tcp=1” is enabled as it is disabled by default.

```
# Change the authentication scheme for TCP sockets.
#
# If you don't enable SASL, then all TCP traffic is cleartext.
# Don't do this outside of a dev/test scenario. For real world
# use, always enable SASL and use the GSSAPI or DIGEST-MD5
# mechanism in /etc/sasl2/libvirt.conf
auth_tcp = "sasl"

# Change the authentication scheme for TLS sockets.
#
# TLS sockets already have encryption provided by the TLS
# layer, and limited authentication is done by certificates
#
# It is possible to make use of any SASL authentication
# mechanism as well, by using 'sasl' for this option
#auth_tls = "none"
```

Figure 33. Enabling the auth_tcp to sasl

In the above screenshot, we have enabled the auth_tcp=”sasl” for the SASL for enabling the SASL authentication over TCP.

We also must make sure that the libvirtd daemon is also listening to the TCP/IP connection. For this we must edit the libvirtd file and then enable the “–listen” parameter in the libvirtd file as it is disabled by default.

```
# Override the default config file
# NOTE: This setting is no longer honoured if using
# systemd. Set '--config /etc/libvirt/libvirtd.conf'
# in LIBVIRTD_ARGS instead.
#LIBVIRTD_CONFIG=/etc/libvirt/libvirtd.conf

# Listen for TCP/IP connections
# NB. must setup TLS/SSL keys prior to using this
LIBVIRTD_ARGS="--listen"
```

Figure 34. Enabling the libvirtd_args to listen.

Now we must restart the libvirtd daemon because of the changes done to the libvirtd file and its configuration file so the changes can take place

```
[root@localhost /]# /etc/init.d/libvirtd restart
Stopping libvirtd daemon:                                  [  OK  ]
Starting libvirtd daemon:                                  [  OK  ]
```

Figure 35. Restarting the libvirt daemon.

As now the libvirtd daemon is accepting the connection through TCP, we need to add users to the SASL database and the following command will add an user named “admin” to the SASL database using the saslpasswd2 command.

```
[root@localhost /]# saslpasswd2 -a libvirt admin
Password:
Again (for verification):
```

Figure 36. Adding the user “admin” to libvirt.

Now that we must verify that the setup done is successful by using the SASL authentication and then instruct the libvirt enabled application should be able to connect using the TCP transport. Now by using the command called Virsh we can connect to the guest vm1 and start it from the remote management station and start the guest in the localhost.

```
[root@localhost /]# virsh -c qemu+tcp://root@localhost/system start vm1
Please enter your authentication name:admin
Please enter your password:
Domain vm1 started
```

Figure 37. Connecting to guest VM using the user.

This is how the remote management is done using the SASL authentication and encryption.

TLS remote management. Transport layer security (TLS) and is formerly called as the Secure sockets layer(SSL) are the cryptographic protocols that will provide the security for the communications over the computer network [17].

Transport Layer Security (TLS) makes sure that the connections are made secure by using the digital signature verification when the server and the client exchanges certificates which are previously signed by the certificate authority(CA). if we look at the most common example, when a web browser is connecting to the web servers, the client application running on the web browser is configured to trust a certain list if the certificate authorities (CAs). At the same time each of the server must prove its identity by providing the certificate which is signed by the list of the CAs trusted by the client application in the web browser. They must provide the same subject name which is usually a fully qualified domain name (FQDN) of the server.

There will be other situations where there is a need of additional security, the server will also require the client to also provide the digitally signed certificate to prove its identity. In this section we will discuss about the how the CAs are created and how the keys and the certificates are created. For this purpose, we use the command called “openssl” which is used to create the private keys which are then directly used by the libvirt. There are several steps for remote management of the guests using TLS which are discussed below:

Creating the CA key in the local host. As the first step we have to login to the local host, i.e., the KVM host and run as root to run the commands.

Then create a temporary directory for storing the files and move into that directory as shown below.

Figure 38. Creating a directory cert_files.

Now by using the “openssl” command we create a RSA key which is of 2048 bits which is as shown in the screenshot below.

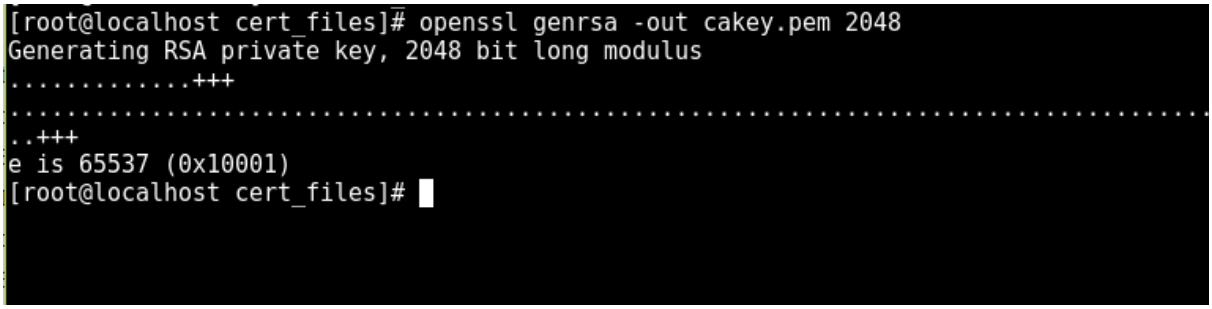


Figure 39. Generating a 2048 bits key.

As shown in the above screenshot, we execute the command and generate the key of 2048 bits and put it into the cakey.pem file.

By using the cakey.pem we must create a certificate which is self-signed using the command shown below.

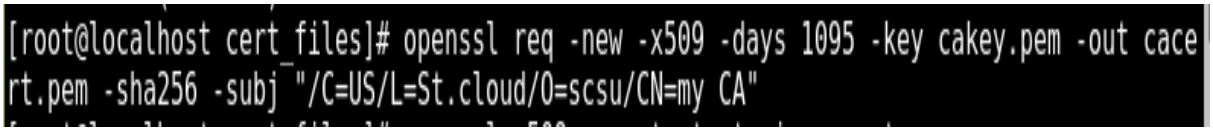


Figure 40. Creating a key for the local host.

With the above command we create the cacert.pem for the KVM local host. Then for checking the certificate we must execute the command shown below.

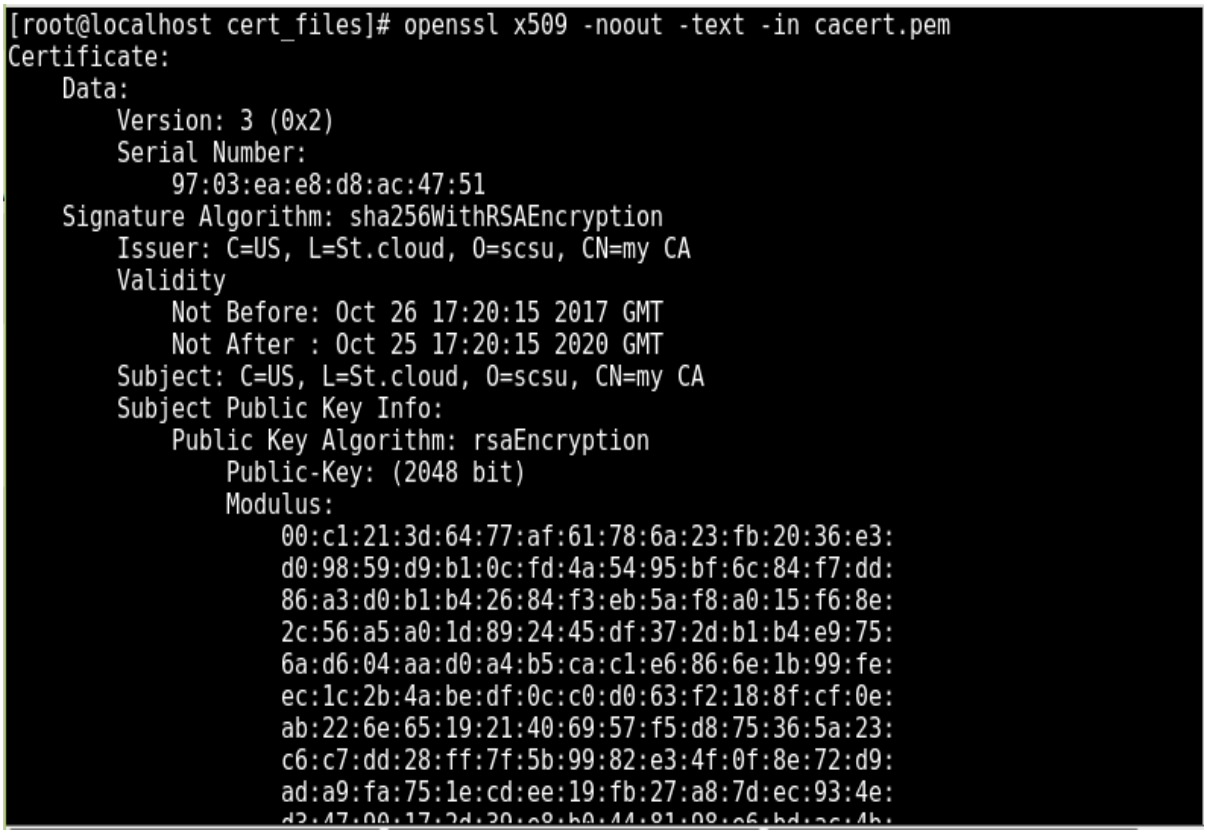
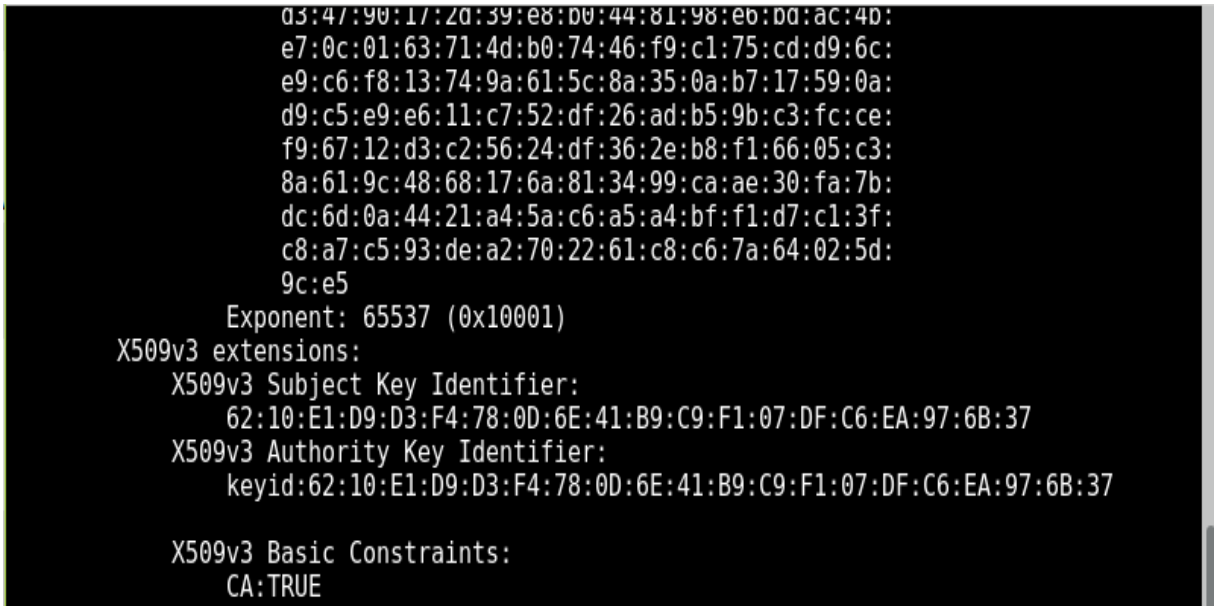
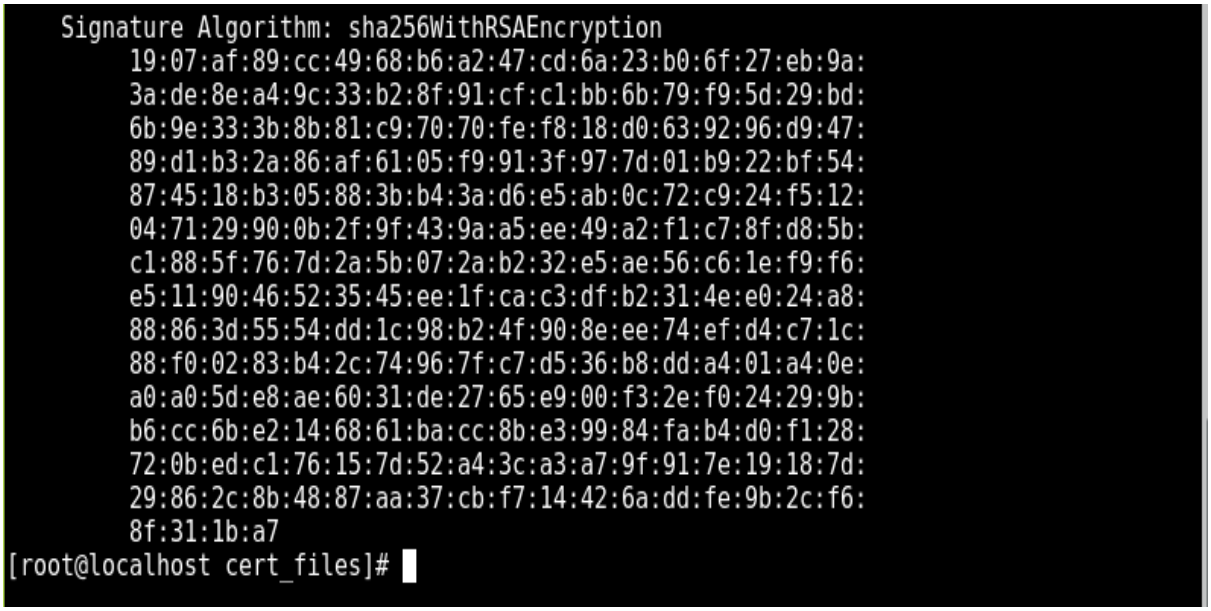


Figure 41. Checking the certificate generated.





Server and client keys and certificates creation in KVM host. The message that is applied for the certificate to a CA is called as the certificate signing request. The certificate signing request contains the identification information for the certificate owner which contains the details of the country, organization, location and so on. This certificate is signed by using the private key.

In this first we create the keys using the command “openssl” which is as shown in the screenshots below.

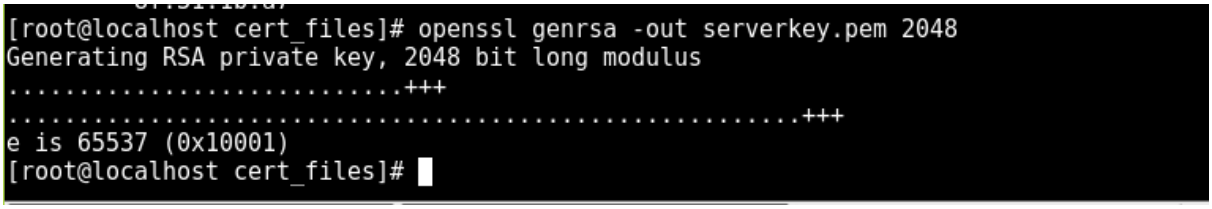


Figure 42. Generating the private RSA key.

The above screenshot shows the creation of the key for the server and put the key in the serverkey.pem.

```
[root@localhost cert_files]# openssl genrsa -out clientkey.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................+++
.....................................................+++
e is 65537 (0x10001)
[root@localhost cert_files]#
```

Figure 43. Generating the private RSA key of the client.

The above screenshot shows the creation of the key for the client and put the key in the clientkey.pem.

Now we must create a certificate signing request for the server which is done by the command which is as shown in the below screenshot.

```
[root@localhost cert_files]# openssl req -new -key serverkey.pem -out serverkey.csr \
> -subj "/C=US/O=SCSU/CN=vm1.img"
[root@localhost cert_files]#
```

The above command creates the certificate signing and put that into the serverkey.csr file. Similarly, we must create a certificate signing request for the client and the server using the command shown below.

```
[root@localhost cert_files]# openssl req -new -key clientkey.pem -out clientkey.csr -su
bj "/C=US/O=SCSU/OU=virtualization/CN=root"
[root@localhost cert_files]#
```

Figure 44. Certificate signing for the local host.

Now we must create the client and the server certificates which are done in the equivalent way we have done in the previous step for the local host.

```
[root@localhost cert_files]# openssl x509 -req -days 365 -in clientkey.csr -CA cacert.p
em -CAkey cakey.pem -set_serial 1 -out clientcert.pem
Signature ok
subject=/C=US/O=SCSU/OU=virtualization/CN=root
Getting CA Private Key
[root@localhost cert_files]#
```

Figure 45. Signing the certificate of the client.

```
[root@localhost cert_files]# openssl x509 -req -days 365 -in serverkey.csr -CA cacert.p
em -CAkey cakey.pem -set_serial 94345 -out servercert.pem
Signature ok
subject=/C=US/O=SCSU/CN=vm1.img
Getting CA Private Key
[root@localhost cert_files]#
```

Figure 46. Signing the certificate of the server.

Now we check the server and client keys using the below commands.

```
[root@localhost cert_files]# openssl rsa -noout -text -in clientkey.pem
Private-Key: (2048 bit)
modulus:
    00:ab:9a:4a:71:58:da:73:e5:a7:6f:e3:af:5b:0d:
    a9:03:1d:87:a9:2a:aa:c8:d6:17:d2:b2:a4:7c:18:
    98:dc:4b:53:aa:9e:5a:6d:7f:40:29:c3:7c:68:3c:
    d9:ba:b8:f7:59:08:b2:5d:d0:d8:41:e7:7a:4c:68:
    55:d3:4f:cf:bb:2a:57:d0:6a:5f:35:05:98:ad:76:
    3f:86:1f:08:d0:5b:c3:38:3b:6c:52:4c:1d:e4:1d:
    0d:e6:7d:7d:34:f5:77:b8:c7:48:f7:6a:a9:aa:58:
    bf:b2:71:45:a1:13:58:bc:27:8d:9d:61:5e:2a:c4:
    73:76:9c:67:01:20:61:67:10:a7:63:ed:37:e4:43:
    42:18:2a:e6:1c:78:cc:95:c5:b9:44:12:22:9a:10:
    f1:f9:9a:b6:57:c4:e4:3a:42:d1:4a:80:8b:6a:9e:
    fb:48:d5:0b:16:a0:19:54:f5:82:1b:8c:97:c3:4d:
    73:e5:27:1e:10:f4:c5:86:5c:56:34:42:2d:42:2b:
    42:f6:c0:aa:c8:65:b4:43:df:45:e2:d5:b4:1f:68:
    4f:32:5a:bc:28:dc:24:81:68:fc:07:d5:e2:e5:7e:
    bf:8c:17:49:b6:a5:92:04:da:9a:c5:05:34:a7:a0:
    5d:20:ad:d9:b3:6e:68:e0:8e:8c:3c:bc:3b:da:e8:
    0a:03
publicExponent: 65537 (0x10001)
```

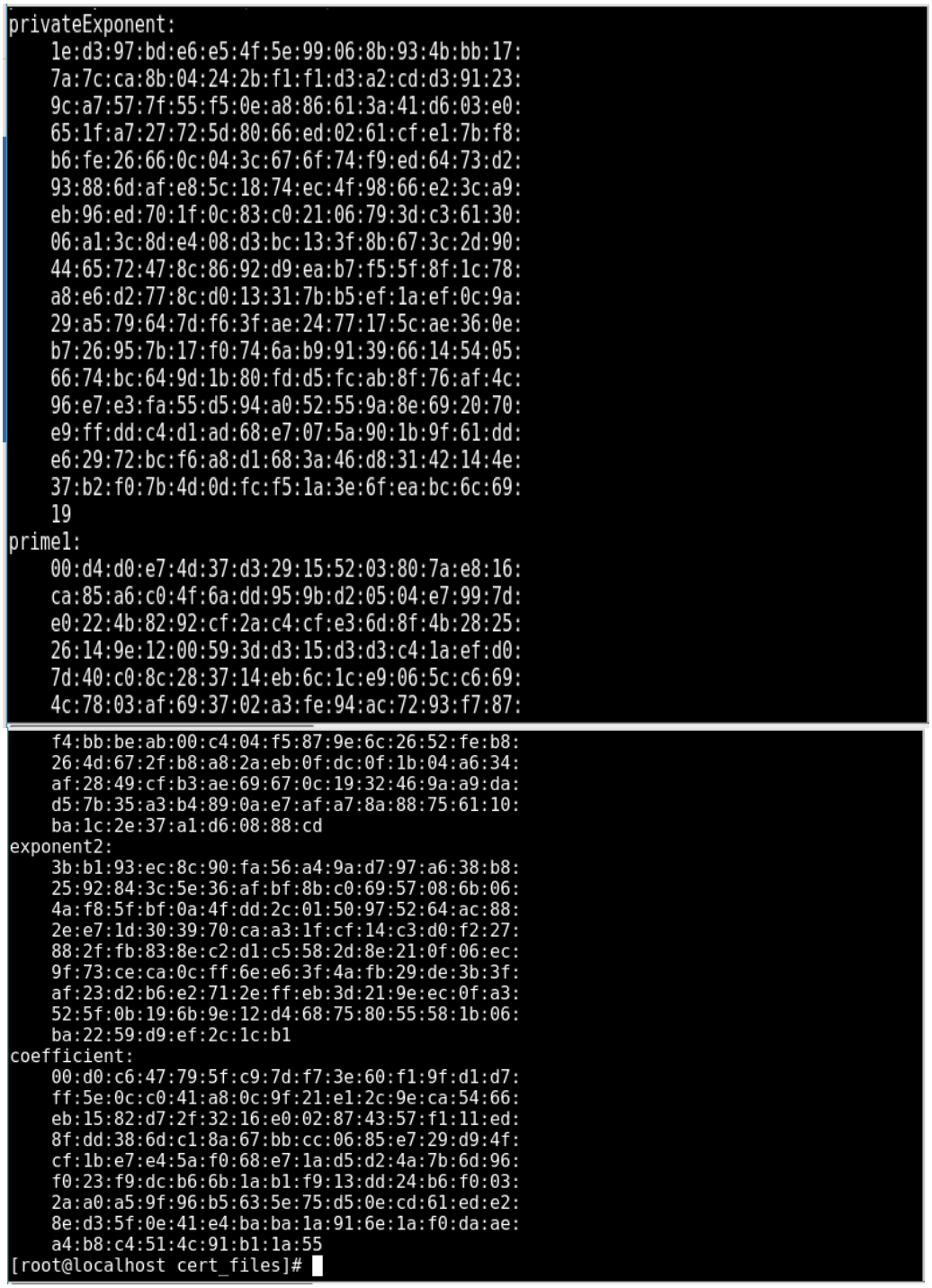


Figure 47. Checking the client key.

The above one is the client key. Similarly, we check the server key.

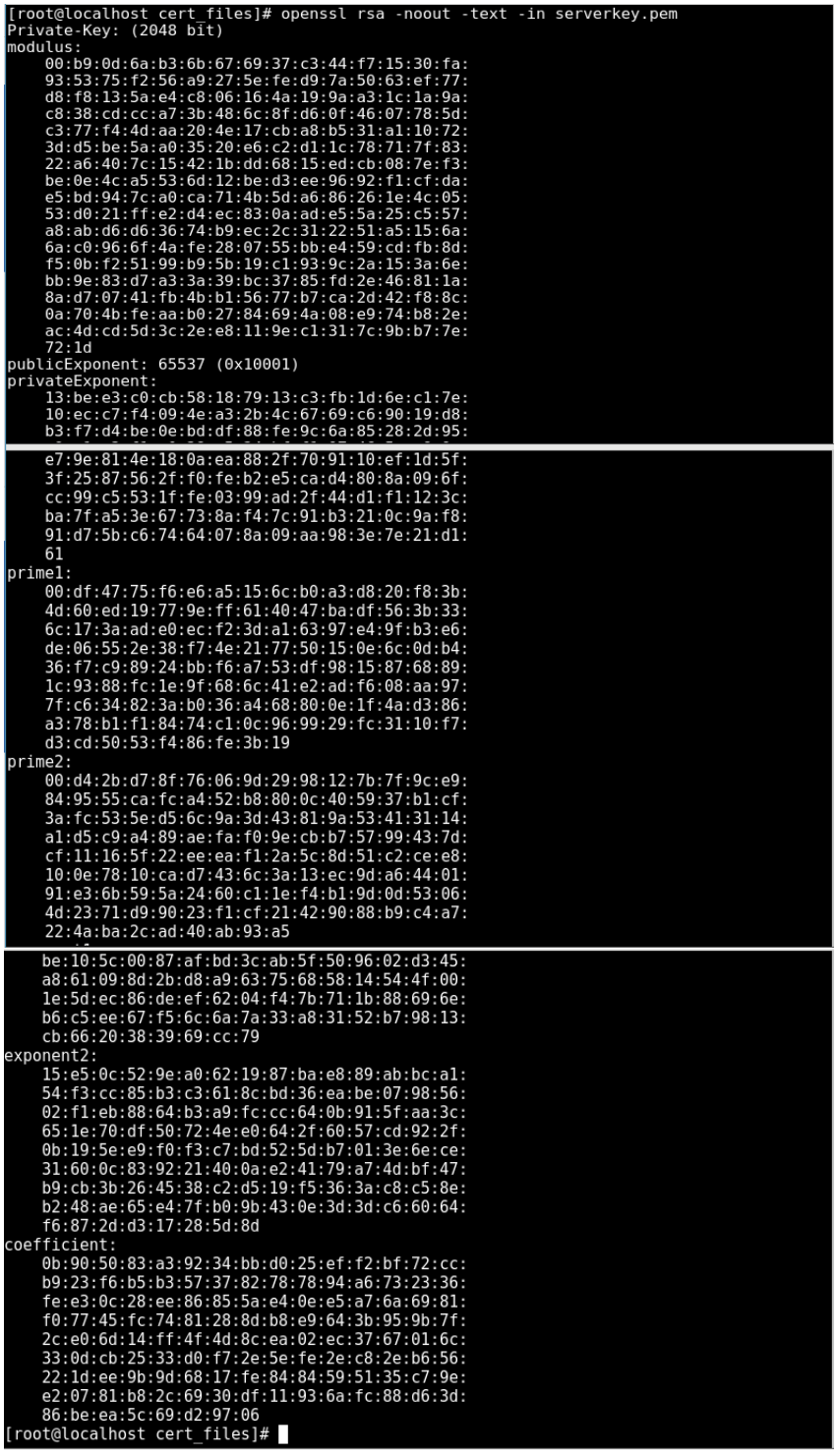


Figure 48. Checking the server keys.

Now we check the certificates of the server using the commands below.

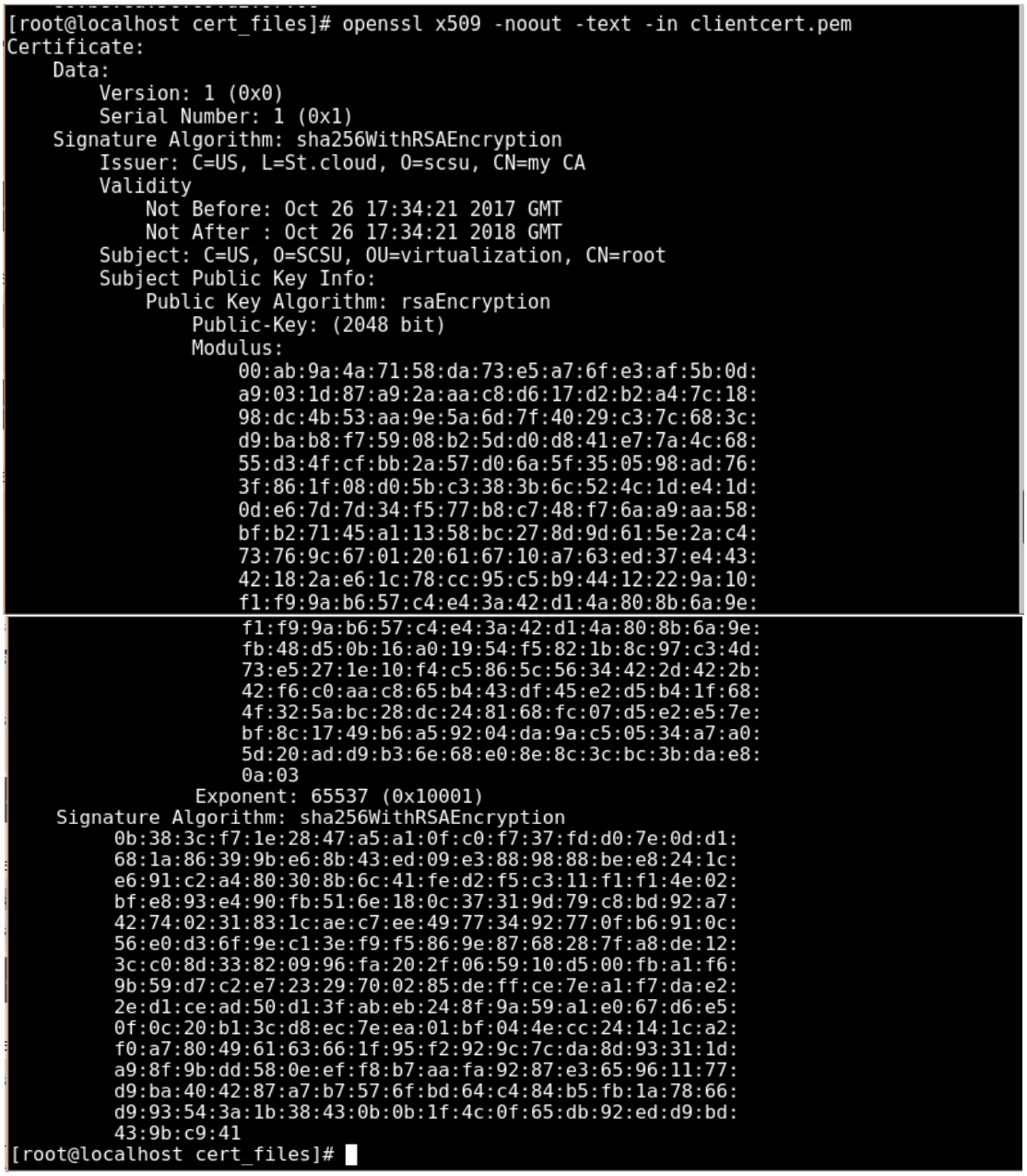


Figure 49. Checking the certificates of the client.

Now check the server certificate using the same command as above.

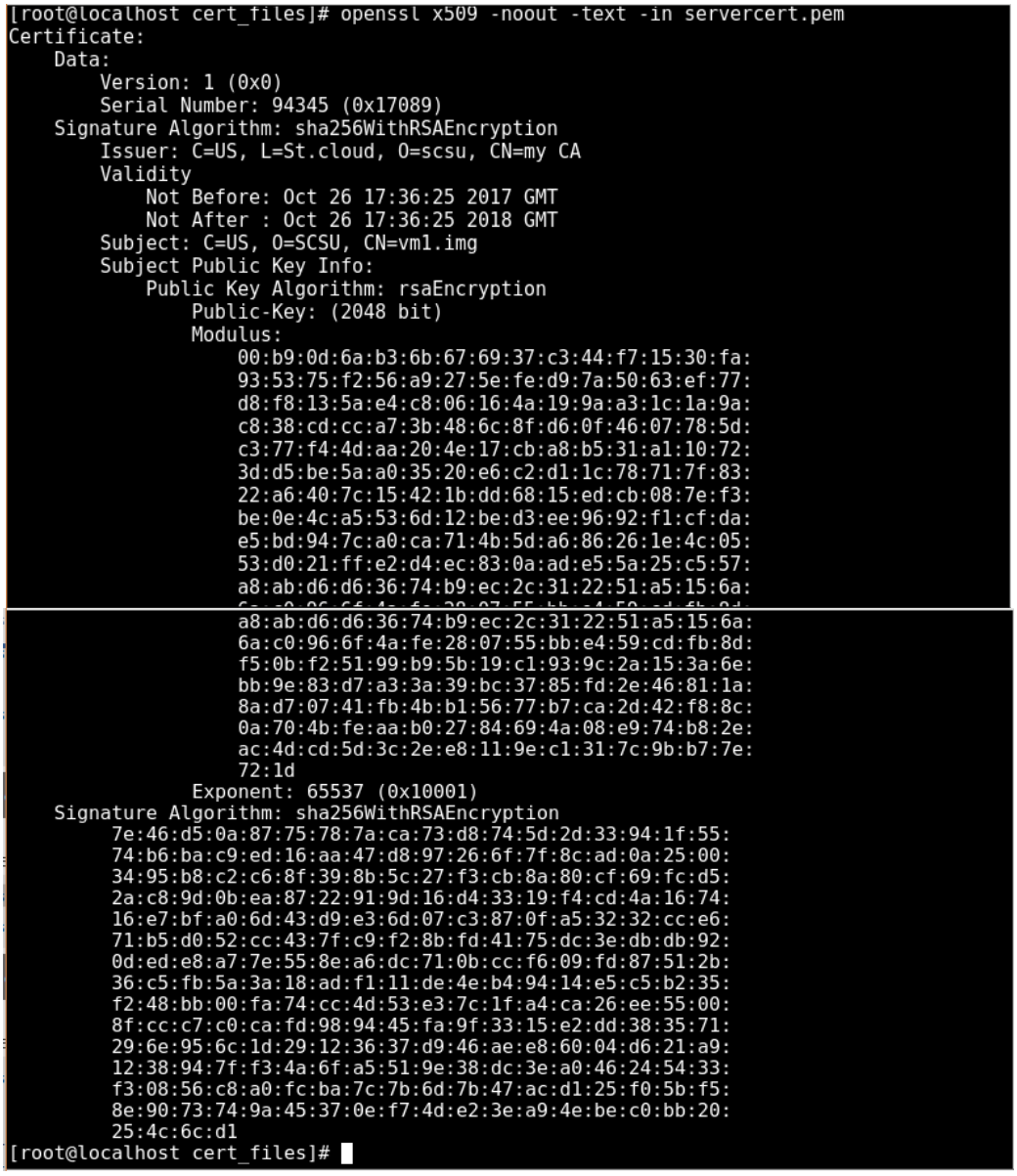


Figure 50. Checking the certificate of the server.

Distributing the certificates and keys to the host (KVM). When the keys and the certificates that are generated by the client and the server are in the format that can be readable by the libvirt, we have to distribute them to the host and configure them so that they are usable by the TLS.

First, to distribute the keys, we have copy the CA certificate file cacert.pem to the /etc/pki/CA/cacert.pem.

```
[root@localhost cert_files]# cp cacert.pem /etc/pki/CA/cacert.pem
[root@localhost cert_files]#
```

Figure 51. Copying the CA certificate.

Then we must copy the servercert.pem and serverkey.pem to the libvirt directory created which is as shown in the below. And make sure that only the root can only access these files.

```
[root@localhost cert_files]# mkdir /etc/pki/libvirt
[root@localhost cert_files]# cp servercert.pem /etc/pki/libvirt/.
[root@localhost cert_files]# mkdir /etc/pki/libvirt/private
[root@localhost cert_files]# cp serverkey.pem /etc/pki/libvirt/private/.
[root@localhost cert_files]# chmod -R o-rwx /etc/pki/libvirt/private
[root@localhost cert_files]#
```

Figure 52. Copying the server certificate and key to the libvirt directory.

Now we must verify whether the files are placed correctly or not.

Distribute the certificates and the keys to the client. First, we have to login to the system do then copy the cacert.pem, clientcert.pem and clientkey.pem to the /etc/pki/libvirt directory in the remote management machine.

```
[root@localhost /]# scp srinath@10.101.80.173:/tmp/cacert.pem /etc/pki/CA/
```

Figure 53. Logging into the remote host and copying the keys and certificates.

The “scp” command will help in copying the files from one host to another host through the command line.

Then we copy the clientcert.pem to the /etc/pki/libvirt/ directory and clientkey.pem to the /etc/pki/libvirt/private/ directory. These steps are shown in the below screenshot.

```
[root@localhost cert_files]# cp clientcert.pem /etc/pki/libvirt/.
[root@localhost cert_files]# mkdir /etc/pki/libvirt/private
mkdir: cannot create directory ‘/etc/pki/libvirt/private’: File exists
[root@localhost cert_files]# cp clientkey.pem /etc/pki/libvirt/private/.
[root@localhost cert_files]# chmod -R o-rwx /etc/pki/libvirt/private
[root@localhost cert_files]#
```

Then make sure that the certificate and the key are placed in the correct folder by executing the following command.

```
[root@localhost /]# ls -lR /etc/pki/libvirt/
/etc/pki/libvirt/:
total 8
-rw-r--r-- 1 root root  767 2010-04-09 13:54 clientcert.pem
drwxr-xr-- 2 root root 4096 2010-04-09 14:00 private

/etc/pki/libvirt/private:
total 4
-rw-r--r-- 1 root root 1044 2010-04-09 13:55 clientkey.pem
```

Figure 54. Checking keys and certificates are placed in correct folder.

Editing the libvirtd daemon configuration. We must make sure that the libvirtd daemon is accepting the network connections and making sure that the libvirtd.conf file specifies which client certificates and the subjects are allowed.

First, we must login into the KVM host. Then we must edit the libvirtd file and enable the “—listen” command in the file which will help the libvirtd daemon is listening to the network connections.

```
# Override the default config file
# NOTE: This setting is no longer honoured if using
# systemd. Set '--config /etc/libvirt/libvirtd.conf'
# in LIBVIRTD_ARGS instead.
#LIBVIRTD_CONFIG=/etc/libvirt/libvirtd.conf

# Listen for TCP/IP connections
# NB. must setup TLS/SSL keys prior to using this
LIBVIRTD_ARGS="--listen"

# Override Kerberos service keytab for SASL/GSSAPI
#KRB5_KTNAME=/etc/libvirt/krb5.tab

# Override the QEMU/SDL default audio driver probing when
# starting virtual machines using SDL graphics
#
```

Figure 55. Enabling the libvirtd args to listen.

Now we must edit the libvirtd.conf file and add what subjects are allowed using the tls_allowed_dn_list as shown in the screenshot below.

```
# A whitelist of allowed x509 Distinguished Names
# This list may contain wildcards such as
#
#    "C=GB,ST=London,L=London,O=Red Hat,CN=*"
#
# See the POSIX fnmatch function for the format of the wildcards.
#
# NB If this is an empty list, no client can connect, so comment out
# entirely rather than using empty list to disable these checks
#
# By default, no DN's are checked
#tls_allowed_dn_list = ["DN1", "DN2"]

tls_allowed_dn_list = ["C=*, O=SCSU,OU=virtualization,CN=*"]
```

Then again restart the libvirtd daemon for the changes to take place.

```
[root@localhost /]# /etc/init.d/libvirtd restart
Stopping libvirtd daemon:                                  [  OK  ]
Starting libvirtd daemon:                                  [  OK  ]
```

Figure 56. Restarting the libvirtd.

Checking whether the remote management is working. Now that we have configured the keys and certificates for the clients and the servers, we have to make sure whether the

client is able to connect to the libvirtd daemon of the KVM host and be able to perform the administration tasks of the virtual machine guest.

To make sure that the client can connect we have to run the following commands form the client which is the remote management system.

```
[srinath@localhost ~]$ virsh -c qemu+tls://root@localhost/system list --all
 Id Name                 State
----------------------------------
  - vm1                  shut off
  - vm2                  running
```

Figure 57. Client is connected to the remote machine.

From the above screenshot, we can confirm that the access of the remote machine which is server in this case is accessed using the remote machine which is a client. We can start the shutdown VM from the remote machine.

```
[srinath@localhost ~]$ virsh -c qemu+tls://root@localhost/system start guest01
Domain vm1 started
```

Figure 58. Accessing the remote machine form the local machine.

SSH Remote Management

SSH, otherwise called Secure Socket Shell, is a system convention that gives executives a safe method to get to a remote PC. SSH additionally alludes to the suite of utilities that actualize the convention. Secure Shell gives solid validation and secure encoded information interchanges between two PCs interfacing over a shaky system, for example, the Internet. SSH is generally utilized by arrange heads for overseeing frameworks and applications remotely, enabling them to sign in then onto the next PC over starting with system, execute orders and move documents starting with one PC then onto the next.

SSH can allude both to the cryptographic system convention and to the suite of utilities that execute that convention. SSH utilizes the customer server demonstrate, associating a safe shell customer application, the end at which the session is shown, with a SSH server, the end at which the session runs.

Aside from Microsoft Windows, SSH programming is incorporated as a matter of course on most working frameworks. SSH likewise bolsters burrowing, sending self-assertive TCP ports and X11 associations while record exchange can be refined utilizing the related secure document exchange or secure duplicate (SCP) conventions. A SSH server, as a matter of course, tunes in on the standard TCP port 22.

The SSH suite involves three utilities–slogin, ssh and scp–that are secure variants of the prior uncertain UNIX utilities, rlogin, rsh, and scp. SSH utilizes open key cryptography to verify the remote PC and enable the remote PC to validate the client, if fundamental.

SSH will provide the authentication and encrypted data while there is a communication between two different computers. This protocol is widely used for the maintenance of the applications remotely which also allows them to login to another computer through the network.

This basically requires only one command “Virsh” through which we give the ssh command and the address of the remote host. Then we can connect directly. The command is as shown below.

Virsh -c qemu+ssh://root@localhost/system list

Chapter V: Conclusion

In this paper, we have presented a problem of how there are security vulnerabilities to the virtual machines running on the host machine and the security of the virtual machines while accessing them through the remote management. Then implemented the creation of the virtual machines using the Kernel Based Virtual Machine as the hypervisor and managing the virtual machines using the virtual machine manager. Then again demonstrated on how one can securely connect with the remote machines using some of the protocols and authentications with the help of the Simple Authentication and Security Layer (SASL), Transport Layer Security and Secure shell.

Future Work

In this paper I have mainly talked about the virtualization while running the virtual machines on the host machine. In the same way we can also connect to the virtual machines running remotely in the cloud instead on our local machine. Then there is another type tool called as “DOCKERS” through which we can implement the virtualization. And most of the companies now use the Dockers for their development.

References

[1] A. Corporation, “Server virtualization: A step toward cost efficiency and business agility,” Avanade Corporate Headquarters, Tech Rep., 2009.

[2] G. Corporation, “Energy saving via virtualization: Green it on a budget,” Gartner Corporation, Tech. Rep., 2008.

[3] S.-W. Lee and F. Yu, “Securing KVM-based Cloud systems via virtualization introspection,” presented at 47th Hawaii International Conference on System Science, 2014.

[4] H. Devassy, C. P. Mukhedkar, and A. Vettathu, *Mastering KVM Virtualization*. 2016.

[5] S. Anish Babu, M. J. Hareesh, and J. P. Martin, “System Performance evaluation of para virtualization, container virtualization, and full virtualization using Xen, OpenVZ, and XenServer,” presented at 4th International Conference on Advances in Computing and Communications (ICACC), 2014.

[6] G. Chen and A. Gillen, “KVM for server virtualization: An open source solution comes of age,” October 2011.

[7] VMware security center, Palo Alto, CA, http://www. vmware.com/support/ security.html.

[8] A. Pokharana and R. Hada, “Performance analysis of guest VM’s on Xen hypervisor,” presented at International Conference on Green Computing and Internet of Things (ICGCIoT), 2015.

[9] R. Morabito, J. Kjällman, and M. Komu, “Hypervisors vs. Lightweight virtualization: A performance comparison,” presented at IEEE International Conference on Cloud Engineering, 2105.

[10] A. Elsayed and N. Abdelbaki, “Performance evaluation and comparison of the top market virtualization hypervisors,” presented at 8th International Conference on Computer Engineering & Systems (ICCES), 2013.

[11] A. R. Riddle and S. M. Chung, “A survey on the security of hypervisors in Cloud computing,” presented at IEEE 35th International Conference on Distributed Computing Systems Workshops, 2015.

[12] L. Yamuna Devi, P. Aruna, D. Sudha Devi, and N. Priya, “Security in virtual machine live migration for KVM”, presented at International Conference on Process Automation, Control and Computing (PACC), 2011.

[13] F. Lombardi and R. Di Pietro, “A security management architecture for the protection of Kernel virtual machines,” presented at 10th IEEE International Conference on Computer and Information Technology (CIT 2010), 2010.

[14] A. Sharma, A. Riaz Ahmad, and D. Singh, “CloudBox—A virtual machine manager for KVM based virtual machines,” presented at 2nd International Conference on Next Generation Computing Technologies (NGCT), 2016.

[15] R. Zhang, G. Liu, X. Yuan, S. Ji, and G. Zhang, “a new intrusion detection mechanism in SELinux,” presented at 2016 International Symposium on System and Software Reliability (ISSSR), 2016.

[16] T. Dierks and E. Rescorl, “The Transport Layer Security (TLS) Protocol, Version 1.2,” August 2008.

[17] A. Sharma, A. Riaz Ahmad, and D. Singh, “Next generation computing technologies (NGCT), 2016 2nd International Conference”, 2016.

[18] A. Sharma, A. Riaz Ahmad, D. Singh, and J. Chandra Patni, “CloudBox–virtual machine manager for KVM based virtual machines,” presented at 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), 2016.