State-of-the-Art of Virtualization, its Security Threats and Deployment Models

Fatma Bazargan, Chan Yeob Yeun, Mohamed Jamal Zemerly
Electrical and Computer Engineering Department, Khalifa University of Science, Technology and Research, PO Box 573, Sharjah, United Arab Emirates

Abstract

Virtualization is an emerging technology which provides organizations with a wide range of benefits. But unluckily, from a security standpoint, functionality often takes precedence over a main area like security, leaving security to be retrofitted in later. This paper mainly emphasizes on several security threats that exists today in a virtualization environment. However, the main contribution of the paper lies in highlighting the notable types of virtualization, its security threats and hypervisor deployment models. To start with, we will further our understanding of the state of knowledge for the latest virtualization technology, the role of the hypervisor and the two different types of virtualization architectures. We will then provide an in-depth explanation about the publicly adopted forms of virtualization and discuss the benefits and drawbacks that accompany each of them. The paper will finally highlight several security threats that exist in a virtualized environment today and the state-of-the-art of open source virtual machine monitor deployment models.

1. Introduction

At present, virtualization has become a way of life in many organizations. It has been applied across multiple Information Technology (IT) aspects such as systems, storage, networks, security and applications. Today we can witness the widespread growth of virtualization technology in testing, training, development and even production environments.

So what is the buzz-word virtualization all about? In order to define it in simple words, it is a technology that introduces a software abstraction layer between the underlying hardware i.e. the physical platform/host and the operating system(s) (OS) i.e. the guest virtual machine(s) (VM) including the applications running on top of it. This software abstraction layer is known as the virtual machine monitor (VMM) or a hypervisor [1-5].

Virtualization technology was first developed in the mid-1960s by the IBM Corporation. At that point in time, virtualization was best known as time sharing. The whole concept of time sharing was that it enabled a number of computer programmers to work on their designated consoles on the same large mainframe computer while avoiding the wait time for the availability of the peripheral. This was possible through partitioning large mainframe computers into several logical instances while running on the single physical mainframe computer as its host. The IBM scientists at that time observed that due to the partitioning capability, numerous processes and applications were able to run at the same time, which increased the total efficiency of the computing environment and reduced the maintenance overhead in [1-3].

This overall notion of optimizing hardware utilization, improving resource utilization by providing a unified operating platform, and trimming down the maintenance overhead drove the advent of virtualization technology. Nowadays, virtualization enables enterprise users and IT developers to have a unified physical platform whilst running on it multiple different operating systems and multiple applications. As enterprises continue to embrace this technology in order to take full advantage of the set of benefits it offers; they often overlook an important area like security.

Hence, migrating an enterprise's computing resources to a virtualized environment not only brings with it all the inherent security threats and vulnerabilities of a running service or an operating system running as a guest but also introduces new virtualization-specific security threats and vulnerabilities that need to be addressed.

2. Virtual Machine Monitor

Any virtualized environment consists of the VMM or the hypervisor whose purpose is to allocate the physical resources (such as the CPU, memory, network, and storage) to each virtualized OS or to each application running on a virtualized OS. Once the hypervisor is in place, it emulates a hardware device for each virtual OS and handles each virtual OS's communications with the physical resources [4, 8-9]. Figure 1 depicts a virtualized environment, where the hypervisor provides an interface between each VM and the underlying physical hardware resource.

The hypervisor software can be either installed on its own or as a part of an OS. Hence, the way the hypervisor software is installed introduces two different forms of virtualization architectures as depicted in Figure 2. In the first form of virtualization architecture known as bare-metal virtualization, there exists no host OS because the VMM sits just above the underlying physical hardware and intercepts the communication between the multiple VMs and the physical hardware. In this form there exists a main management VM whose core responsibility is to manage the guest VMs and the communication with the physical hardware [6-7].

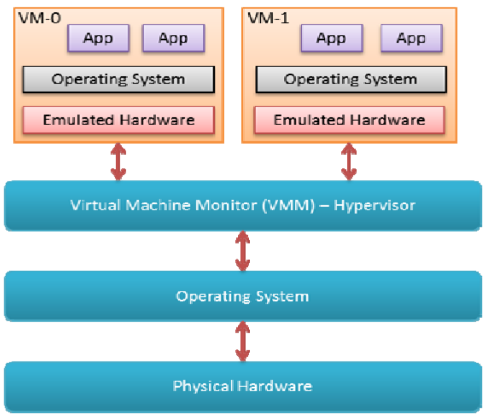


Figure 1. Illustration of a virtualized environment

In the second form of virtualization architecture known as hosted-virtualization, the VMM sits just on top of the host OS and runs as an application. It is the host OS that is responsible to provide Input/Output (I/O) drivers and to manage the guest VMs [6-7].

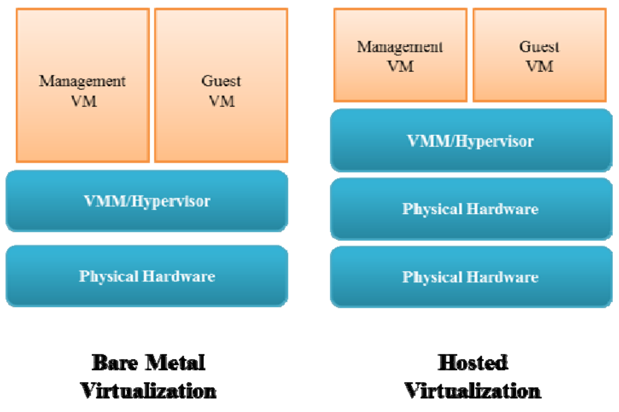


Figure 2. Types of virtualization architectures

The main characteristic of the VMM is that it removes the traditional physical hardware dependency of an OS on its physical hardware resource. In other words, the physical resources are directly controlled by the VMM and not by the host OS or the physical hardware. Due to this fine trait, multiple different OS(s) can run on the same physical hardware at the same time while being isolated from one another. As a result, the physical hardware is partitioned into one or more logical units known as virtual machines (VMs). A VM can run any x86-class physical host system, regardless of the hardware structure as in [1-3] and [8].

There are three primary attributes to the VMM:

1. **Isolation:** only the VMM in a virtualized environment has the responsibility to control and monitor all guest VMs residing on the physical hardware and allocate the required physical resources to each guest VMs. VMMs provide isolation; that is every VM is isolated from any other VM on the same physical hardware. In other words, applications running on one VM cannot interact or see applications that are running on a different VM. In addition, every VM is isolated from the host OS in the same way [9-11].
2. **Interposition:** VMMs manage all administrative privileged instructions on the physical hardware. The guest OS communicates all its traps and interrupts to the VMM, which will eventually process the events by interacting directly with the physical platform on behalf of the guest OS. The VMM intercepts all I/O requests from the guest OS(s) to the virtual devices and maps them to the exact physical I/O device through an I/O abstraction. Through this abstraction the VMM manages and schedules all VMs concurrently [9-11].
3. **Inspection:** the VMM has access to all the states of the VMs running on the physical hardware. This includes the memory state, CPU state, and I/O device states. This is necessary for VMM in order to encapsulate the state of a VM to enable capabilities such as check pointing (i.e. capture a snapshot of the current state), rollback (i.e. restore to a previously defined state), and replay (i.e. repeat to view an event). This fine trait helps administrators to save, copy, move, compare and instantiate environments with protected states from one physical host to another [9-11].

3. Virtualization and Protection Rings

Protection rings architecture is a formal mechanism to segregate the trusted operating system from the untrusted user programs [2]. These rings allow various levels of isolation and abstraction of privilege within the architecture of a computer system. The rings are arranged in a hierarchical way from most privileged (i.e. most trusted and unrestricted access to resources – Ring-0) to least privileged (i.e. least trusted and restricted access to resources – Ring-3) as presented in Figure 3. Ring-0 is the most privileged ring that interacts directly with the physical hardware resources [1-2, 12].

The least privileged rings cannot access the inner rings without predefined instructions in place. These restrictions are set on the outer rings to protect data and functionality from faults, misuse of resource, and malicious behavior. For example, no user level program running in Ring-3 should be able to turn on the speakers (i.e. a device driver) without the consent of the user, because device drivers are a Ring-1 function which is a higher privileged level than Ring-3 [12].

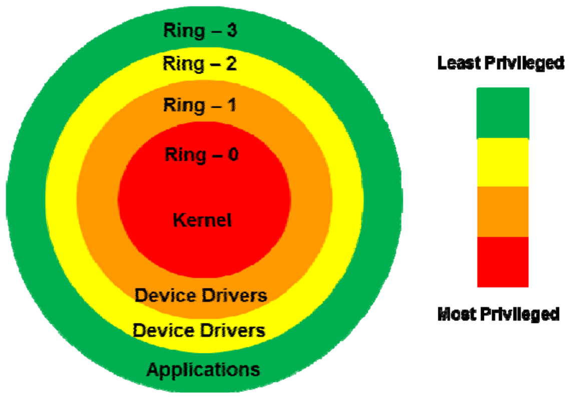


Figure 3. Protection rings in x86 CPU architecture

The most common CPU architecture is the x86-compatible which provides four protection rings: 0, 1, 2, and 3. Only Ring-0 and Ring-3 are usually used. Literarily, Ring-0 is the kernel mode (also known as supervisor mode or system space) and Ring-3 is the user mode. The kernel mode is where the code is executed without any restriction access to the underlying hardware resources. Any CPU instruction can be executed and any memory address can be referenced in this mode. It is reserved for the most trusted functions of the OS. In the user mode on the other hand, the code is executed with no ability to directly access the underlying hardware resources or reference any memory address. Hence, the codes running in this mode should delegate to system application programming interfaces (API) to access the hardware or memory [2].

In a virtualized environment the hypervisor is said to run in the kernel mode (i.e. Ring-0), because, it is the responsibility of the hypervisor to assign the hardware resources and allocate memory address to the guest VMs. Henceforth, the kernel of the guest VMs runs in a less privileged ring than Ring-0. Therefore, the kernel of the guest VMs has less privilege to access resources or reference memory address without the consent of the hypervisor [12].

4. Virtues and Vices of Virtualization

Like every other technology, virtualization also has its own set of benefits and drawbacks. This section will elaborate on the numerous drivers that have motivated the quicker embracement of virtualization technologies and the drawbacks that organizations should be alert about.

4.1 The Benefits of Virtualization

1. **Consolidation:** the primary goal behind it is combining and unifying. The workloads are combined on fewer physical platforms which are capable of sustaining their demand for computing resources, such as CPU, memory, I/O. The result is increased resource utilization and hardware optimization. In the sense, you have a single hardware platform that can support a multitude of virtual environments. In addition, virtualization helps ease and simplify legacy system migrations by providing a common and widely compatible platform upon which legacy system instances can run. This improves the overall chances that applications can be migrated for older, unsupported, and riskier platforms to newer hardware and supported hardware with minimal impact. Consolidation also helps to streamline development and test environments.
2. **Reliability:** virtualization maintains functionality and operation availability by providing high isolation between its virtual machines. In the sense, a system fault on one VM or its partition will have no effect on other partitions that are running on the same physical platform. In addition, virtualization can dynamically allocate resources to a single logical or physical partition at a given runtime. This is known as just-in-time or on-demand provisioning of additional partitions as and when required with no concern about hardware procurement, configuration, or installation.
3. **Security:** virtualization introduces the added value of security through the encapsulation and isolation of virtual machines operating systems. In other words, if a certain partition or service on a VM is compromised this stops the compromise from being extended to other partitions that are existent on the same virtualization platform. For example if a virtualization platform hosts a VM for a web server and a VM for a mail server, if the web server service is compromised this does not mean that the mail server service is also compromised because each VM is isolated from any other VM on the same physical host. In addition, the security configurations for each partition remain specific to its requirements rather than the server as whole [13-15].

4.2 The Drawbacks of Virtualization

1. **Performance** is an important trait for any organization to accomplish its operations efficiently. IT professionals should identify what are their performance requirements. For example some hardware platforms do consider virtualization; whereas others just operate efficiently as a standalone. In addition, some types of applications may need substantial CPU processing power whereas other may need significant I/O requirement [5].
2. **Redundancy:** the virtualization hardware platform is still considered a major issue. In other words, organizations rely immensely on data accessing and data processing. They need to ensure that this sort of access is highly available and in any case of hardware failure the workload is transferred to another system. But in case of a virtualized platform; if it is hosting five different virtual machines that are running different services if a hardware failure occurs in the virtualization platform that will bring down the entire VMs, unless a redundant system is put in place with the same configuration and system specifications [2], [4-5].
3. **Operations:** IT professionals responsible for virtualized environments should consider the maintenance cost that accompanies the deployment of a virtualized environment such as licensing constraints, software upgrades, change control, VM lifetime, hardware maintenance, etc. In addition, IT professionals should have a unique set of expertise to maintain and manage a virtualized environment considering the complexity characteristics it comes with [1-2] [4-5].

5. Approaches of Virtualization

There exist several approaches of virtualization that are deployed in IT environments. This section will handle the various approaches of virtualization and will highlight some of their advantages and disadvantages.

5.1 Full Virtualization

This approach of virtualization is a type of Server Virtualization. Server virtualization abstracts both the physical resources on the physical host as well as the guest OS that runs on the physical hardware. In other words, full virtualization virtualizes all features of physical hardware resources [6]. This approach provides the virtualization environment with the ability to entirely simulate the underlying physical hardware. The resultant is a system where any software that is capable of direct execution on the physical host is able to run in the VM, and any OS that is supported by the underlying physical host can run in each individual VM.

Different operating systems can run concurrently in a full virtualization setup. In addition, the Input/Output (I/O) devices are allocated to the guest OSs by emulating the physical devices in the VMM. Interacting with these I/O devices in the virtual environment are then directed to the real physical devices either by the host OS driver or by the VM driver [6]. Figure 4 illustrates the full virtualization concepts.

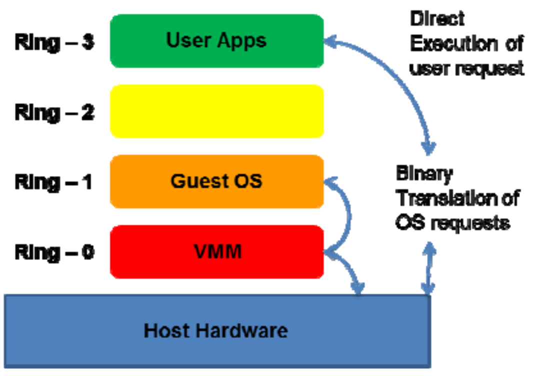

Ring – 3
User Apps
Direct Execution of user request
Ring – 2
Ring – 1
Guest OS
Binary Translation of OS requests
Ring – 0
VMM
Host Hardware

Figure 4. Full virtualization concepts

Full virtualization is fulfilled by the use of both binary translation and direct execution. In other words, the hypervisor resides in Ring-0; hence, the physical CPU executes non-critical instructions at native speed. On the other hand, guest OS instructions are translated on the fly and cached for future use. In addition, user level instructions do run unmodified at native speed. With these settings full virtualization offers best isolation and security for guest virtual machines and can be easily migrated [2].

The advantages of this approach are that it provides complete isolation between each guest VM and any other VM residing on the same physical host and between the guest VMs and the VMM. Moreover, it is easy to use, in the sense that any user can install a software product such as VMware Workstation on the preferred choice of OS and once they switch the VMware workstation on; a guest OS can be installed and used. In addition, users can run multiple different guest OS(s) simultaneously and this approach provides near-native CPU and memory performance.

The disadvantages of this approach are that it requires the exact appropriate blend of hardware and software components and poor performance of the emulated VM due to the impact by trap- and emulate techniques of x86 privileged instructions [2-3, 6, 12]. Both VMware and Microsoft Virtual Server virtualization solutions use the full virtualization approach.

5.2 Para virtualization

This approach is considered a subset of server virtualization. However, it differs from full virtualization in the sense that the running guest OS should be modified in order to be operated in the virtualized environment [2]. Para virtualization provides partial simulation of the underlying hardware. Wherein, most but not necessarily all hardware components are simulated. There exists a thin software interface between the physical hardware and the modified guest OS. A fascinating point in this technique is that the guest VMs are aware of the fact that they are running in a virtualized environment [6].

In other words, in this approach the guest OS kernel acts as a bridge between the applications and the execution done on host hardware level. Para virtualization substitutes non-virtualized instructions with something known as Hypercalls which communicate directly with the VMM. The concept of a Hypercall is similar to that of a system call. Wherein, system calls are used by any application at the user level to ask for services from the OS. Hence, it provides the interface between the application and the OS. But with Hypercalls the communication is between the application and the hypervisor itself [6].

One of the main characteristics of Para virtualization is that it is easier to implement than the full virtualization approach. The reason behind it is because once the host OS boots the VM emulator is launched which leads to the host going to a suspension mode and the guest OS running in an active state. In addition, the guest OS shows high performance for network and I/O disk when no hardware assistance is available [3-4]. Talking about its disadvantages, the guest VMs running in a Para virtualized environment require a substantial amount of OS kernel modification, VMs that suffer from lack of backward compatibility and are not very portable or easy to migrate to other hosts [6]. Figure 5 illustrates the Para virtualization concept. Xen virtualization system uses the Para virtualization approach.

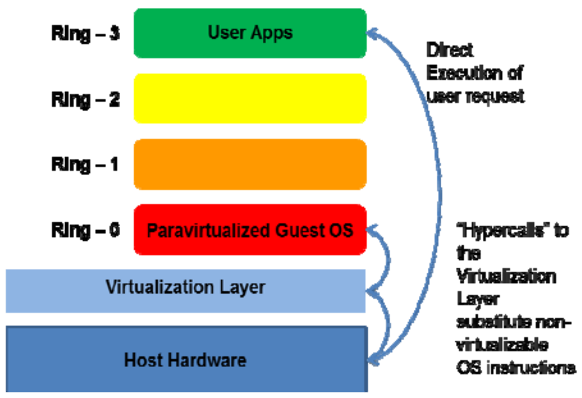

Ring – 3
User Apps
Direct Execution of user request
Ring – 2
Ring – 1
Ring – 0
Paravirtualized Guest OS
"Hypercalls" to the Virtualization Layer substitute non-virtualizable OS instructions
Virtualization Layer
Host Hardware

Figure 5. Para virtualization concepts

5.3 Hardware-Assisted Virtualization

This technique is also known as native virtualization, accelerated virtualization, and hardware VM depending on the manufacturer. This approach of virtualization neither uses the binary translation as used in full virtualization nor uses the Hypercalls as used in Para virtualization [2]. But it is an approach that allows for a CPU instruction set communication in which the hypervisor runs in a new level known as the root level mode which resides below the OS kernel level.

In this virtualization form the critical and privileged instructions are set to automatically trap the hypervisor directly. This approach was introduced recently hence the disadvantage introduced by the first generation included lagging behind in performance when compared to full virtualization. But, this can be solved with the introduction of the second generation of hardware-assisted technologies. Figure 6 illustrates the hardware-assisted virtualization concepts [6].

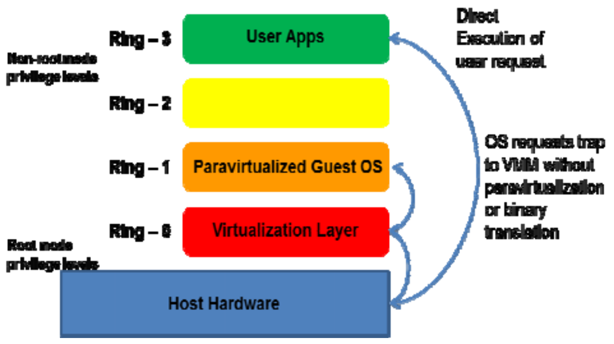

Non-root mode privilege levels
Ring – 3
User Apps
Ring – 2
Ring – 1
Paravirtualized Guest OS
Ring – 0
Virtualization Layer
Root mode privilege levels
Host Hardware
Direct Execution of user request
OS requests trap to VMM without paravirtualization or binary translation

Figure 6. Hardware-assisted virtualization concepts

5.4 Operating System Virtualization

This technique is form of server virtualization and is also known as Single Kernel Image (SKI) or container-based virtualization. This concept is based on single OS instance which means running more instances of the same OS in parallel. In other words, not the actual hardware but the host operating system is the one being virtualized and the resulting VMs all use the same virtualized OS image. The virtualized operating system image is known as the virtualization layer [1-2, 5-6].

This architecture eases the administration of the system by allowing system administrators to allocate resources both when creating a VM as well as dynamically at runtime. When comparing this technique to other server virtualization setups, OS-layer virtualization tends to be more efficient but slightly fails to provide the same isolation level as the rest.

However, this approach also has its shortcomings since the VMs must use the same kernel as the host OS it becomes inconvenient to run for example Windows on top of Linux.

5.5 Application Virtualization

In this form of virtualization, the user has the ability to run a server application locally by using the server application resources without the need and the complexity of installing the required application on his/her personal computer [1-2] and [5,6].

These virtualized applications on the fly are designed to run in a small virtual environment containing only the resources required for the application to execute. Thus in this approach each user has a virtual isolated application environment virtually.

5.6 Network Virtualization

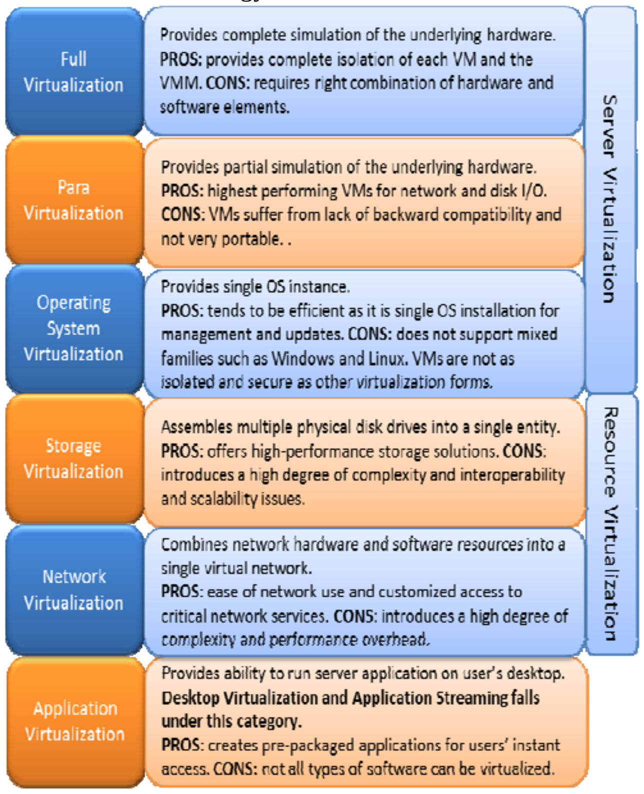
This form is a type of resource virtualization. Resource virtualization virtualizes system specific resources such as storage drives, name spaces and the network resources. Network virtualization is to combine the network hardware and software resources into a single software-based manageable entity known as a virtual network. This form can be categorized either by combining various networks (external) or parts of networks (internal) [1-2].

5.7 Storage Virtualization

Storage virtualization is also a form of resource virtualization, where a multiple physical disk drives are assembled into a single logical entity that is then provided to the host server and operating system. The anatomy of this assembly is as follows: the physical storage resources are aggregated to form a storage pool which then forms a logical storage. This logical storage appears to be a single uniform storage device to the end user [1-2].

5.8 Summary of Approaches to Virtualization

Figure 7 summarizes all the various types of virtualization technology.



Full Virtualization — Provides complete simulation of the underlying hardware. **PROS:** provides complete isolation of each VM and the VMM. **CONS:** requires right combination of hardware and software elements.

Para Virtualization — Provides partial simulation of the underlying hardware. **PROS:** highest performing VMs for network and disk I/O. **CONS:** VMs suffer from lack of backward compatibility and not very portable. .

Operating System Virtualization — Provides single OS instance. **PROS:** tends to be efficient as it is single OS installation for management and updates. **CONS:** does not support mixed families such as Windows and Linux. VMs are not as isolated and secure as other virtualization forms.

Storage Virtualization — Assembles multiple physical disk drives into a single entity. **PROS:** offers high-performance storage solutions. **CONS:** introduces a high degree of complexity and interoperability and scalability issues.

Network Virtualization — Combines network hardware and software resources into a single virtual network. **PROS:** ease of network use and customized access to critical network services. **CONS:** introduces a high degree of complexity and performance overhead.

Application Virtualization — Provides ability to run server application on user's desktop. **Desktop Virtualization and Application Streaming falls under this category. PROS:** creates pre-packaged applications for users' instant access. **CONS:** not all types of software can be virtualized.

Figure 7. Summary of the types of virtualization

6. Security Threats in a VE

Many of the unique characteristics of virtualization offer both benefits and drawbacks to security. This section will elaborate on virtualization-specific threats and vulnerabilities that exist today and need to be addressed.

6.1 Virtualization Based Malware

When talking about virtualization based malware we categorize it to either software-based malware or hardware-based malware.

The first controversial rootkit that comes to mind when talking about virtualization based malware is that of the codename “Blue Pill”. It is a special type of malware that utilizes the virtualization methods of certain CPUs to execute as a VMM. It induces a virtual platform on which the entire operating system runs on. It explores the whole status of the machine and allows the intruder full privilege whilst the OS thinks it is running directly on the physical hardware. The main concept behind its functionality is that the piece of malware can take over the host OS and can be undetected by residing within the VMM [1-2].

The “Red Pill” on the other hand operates just the opposite manner of the “Blue Pill”. In the sense, the Red Pill technique aids the OS to detect the presence of a hypervisor. When Red Pill runs on VM, it focuses on identifying VM usage without looking for file system artifacts based on relocation of sensitive data structures. In other words, it is able to conclude if it is running in a virtual platform or on a real physical platform. Other two rootkits are the Vitriol and SubVirt. The SubVirt installs a VMM beneath an existing OS and moves the original OS into a VM. It is capable of modifying the system boot sequence and emulates a set of different virtual devices. Vitriol on the other hand is a hardware-based rootkit with capacity to swap the entire OS-visible state of the processor in and out of memory and the VMs have direct memory and device access [16-17].

The main reason why these virtualization based malware succeed in being a security challenge in a virtualized environment is because VM tools leave a large footprint including running processes, services, and registry keys. These VM based malware often look for these items and then conclude if they are running in an environment that is real or virtual.

6.2 Denial of Service Attack

The isolation attribute of virtualization is considered both an opportunity and a threat. In a virtualized environment the isolation happens among guest OS VMs and between the guest OS VM and the host OS. Each VM has its own dedicated set of physical resources such as memory disk, CPU, network resource, etc. that is shared with the underlying host. None of the guest VMs can allocate,share or take over the physical resources of any other guest VM running on the physical host due to the fine isolation attribute.

However, Denial of Service (DoS) attack can occur in a virtualized environment when an attacker takes over a guest VM and is then able to gain control over the physical resources of other guest VMs on the same physical host. A simple DoS scenario can occur when a compromised guest VM tries to overcome the isolation barriers by taking over all the physical resources of the physical host causing corruption and/or unavailability of other guest VM or the host OS. Hence, the physical host is unable to process service requests made by other individual guest VMs leading to denial of access due to the unavailability of requested resources as in [8] and [17].

6.3 Communication Attack among guest VMs and the host

As mentioned, isolation is one of the primary benefits of virtualization. This attribute enables each guest VM to execute all its actions confined to its own address space. Hence, allowing each guest VM to be self-protected and encapsulated from one another (i.e. in resource allocation or execution of its own applications). Isolation should be carefully configured and closely monitored in a virtualized environment to avoid the interference or unwanted accessibility among the various guest VMs themselves or between the guest VMs and the physical host.

However, the introduction of features such as shared clipboard that allows data to be transferred back and forth between guest OS VMs and the host OS has introduced a security challenge. Wherein, this may be treated as a gateway for transferring malicious codes between the guest OS VMs and between the guest OS VM and the host OS. In addition, some virtualization avoids isolation altogether to support applications designed for one OS to be operated on another OS, this may introduce flexibility but it also raises a security challenge. Because where there is no isolation between the host OS and the guest OS VMs grants the guest OS VMs an unlimited access to the host OS’s resources, such as file system and networking devices and in this case the host OS’s file system becomes vulnerable [8] and [17].

6.4 VM Escape

The anatomy of a virtualized environment is that of having multiple guest VMs running in isolation. The VMM is responsible to provide isolation between the various VMs residing on the physical host. This isolation and encapsulation feature is either made among the guest VMs themselves or between the guest VMs and the actual physical host. In other words, any guest VM on a physical host is in an isolated environment and cannot interact with, monitor, or control any other VM on the same host. In addition, the guest VM OS is incapable of knowing if it is actually running in a virtual or real environment. More importantly, there is no way to jailbreak out of a guest VM and interact directly with the VMM.

However, the process wherein a guest VM can jailbreak and directly interact with the VMM is known as “VM escape”. The VMM is the core of any virtualized environment. It resides between the physical host and the guest VM OS and it has the capability to control the execution of all different guest VMs that are running on the physical host. Therefore, any attacker having the ability to escape the guest VM environment and directly interact with the VMM will gain access over every other VM on the physical host. This compromise of the VMM by VM escape is known as “hyperjacking” [18].

6.5 Inter-VM Attacks and Network Blind Spots

Network blind spots is when the traditional network security solutions are blinded and cannot detect the malicious communication behavior that might occur between the guest VMs and between the guest VMs and the host (i.e. inter-VM) residing on the same physical platform. This introduces a significant security threat in a virtualized environment.

There are two scenarios to inter-VM attacks. In one scenario, the compromise of a guest VM residing on a physical host can enable the attacker to compromise all other guest VMs on the same host. The increase in the number of guest VMs that resides on a physical host eventually increases the risk of greater compromise from a guest VM to another guest VM [18].

The other scenario is, understanding the vital role of the hypervisor in a virtualized environment makes it the perfect target for an attacker. Because, once the hypervisor is compromised it makes it fairly easy to compromise all other guest VMs that reside on the physical host [18].

6.6 Summary of Security Threats in a VE

The crucial aspect in order to ensure information security in an organization is to safeguard the confidentiality, integrity, availability, authentication, authorization and accountability of network traffic and business data. In a VE context, these crucial security aspects can be defined as follows:

1. Confidentiality: to ensure that the network traffic and business user data in a VE remains protected while in transit and/or at rest from unauthorized access.
2. Integrity: to ensure that the network traffic and business user data in a VE cannot be modified, damaged, or deleted by unauthorized access.
3. Availability: to ensure that network traffic and business user data, and services are available when in need by authorized users.
4. Authentication: a process to ensure the identity of the authorized user.
5. Authorization: to ensure that the authorized user has a set of privileges and rights to execute certain activities.
6. Accountability: to ensure that proper audit trails and checks are in place to monitor the access rights of the authorized user.

Table 1 summarizes the impact of security threats on the various crucial security aspects and the required safeguard measures towards a safer and secure VE.

Table 1. Impact of security threats on crucial security aspects and the required safeguard

Security Threats	Security Components	Safeguards
Virtualization Based Malware	Integrity	1. Hypervisor security & integrity checks. 2. Guest OS security & hardening. 3. Virtualized network security & isolation. 4. Zero-day real-time detection of malicious activities. 5. Security policies and controls in place 6. Automatic restoration of guest VMs to a clean state.
Denial of Service	Availability	
Communications Attack	Confidentiality Authentication Authorization	
VM Escape	Authentication Authorization Accountability	
Inter-VM Attacks and Network Blind Spots	Authentication Authorization	

7. VMM Deployment Solutions

In this section, we will elaborate on the most publicly deployed virtual machine monitors. These include VMware, Microsoft’s (Hyper-V), Citrix (Xen system), Kernel-Based Virtual Machine (KVM), and OpenVZ. Because, KVM, Xen and OpenVZ are currently the open source VMM versions available for x86 platform, hence, this section will focus on these three.

7.1 Xen

Xen, is a widely adopted open source industry standard for virtualization. It can work both in para-virtualization and the hardware-assisted virtualization modes. It supports a wide range of guest operating systems including Linux and Windows. It allows several guest operating systems to be executed concurrently on the same physical machine. XenServer is based upon Xen which is currently owned by Citrix. The Xen system structure consists of the Xen hypervisor which is the lowest and most privileged software layer; this layer supports one or more guest operating systems [19-20].

The first created guest OS on Xen is a privileged VM known as domain 0 (Dom0) which executes automatically once the hypervisor boots, receives special management privileges, and gets direct access to all physical hardware by default. Any other additional guest OS is known as domain user (DomU). It is in the DomU where the guest VM operating systems are executed [19-20]. Figure 8 illustrates the Xen execution model.

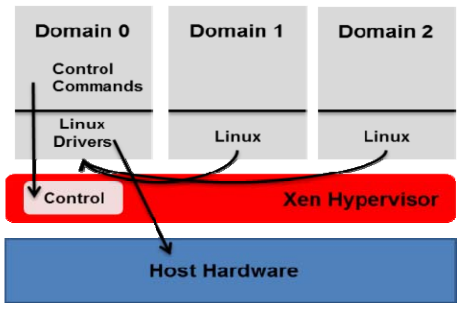


Figure 8. Xen execution model

The Xen hypervisor runs in Ring-0 and all the other guest operating systems run in Ring-1, by following this structure Xen hypervisor decouples the guest OS from underlying physical machine while holding full control on system resource. The privileged Dom0 hosts most unmodified Linux device drivers, hence, playing the role of a driver domain and takes control of other guest domains.

7.2 KVM

KVM on the other hand is an open source software fairly recent Linux based virtual machine monitor. It supports full virtualization on processors with hardware-assisted virtualization extensions (i.e. Intel or AMD) for Linux on x86 hardware. It also supports a wide selection of guest operating systems including Linux and Windows. KVM consists of a hypervisor (i.e. Loadable Linux Kernel Module, kvm.ko) this module provides the core virtualization infrastructure and it has a processor specific module, kvm-intel.ko or kvm-amd.ko [20]. Figure 9 illustrates the KVM execution model [20].

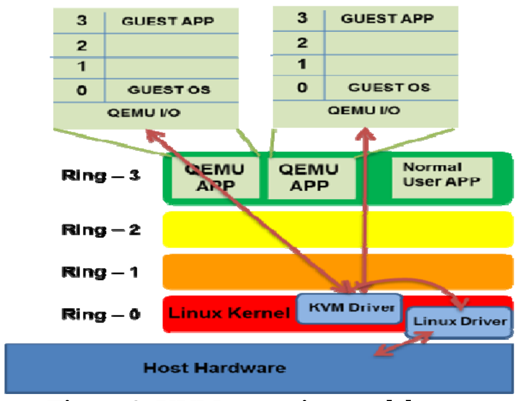


Figure 9. KVM execution model

KVM also entails a modified version of QEMU which is emulation software. KVM consists of two main components: a kernel module and a user space. Kernel module is responsible for the virtualization of hardware resources. However, the user space program uses the /dev/kvm interface to create the guest VM's address space and is responsible for I/O's virtualization with the assistance of the QEMU to simulate the behavior of I/O. For example, any I/O request made by the guest operating system is trapped within the user space and simulated by QEMU [20].

7.3 OpenVZ

OpenVZ is a Linux-based operating system-level server virtualization developed by SWsoft. On a single physical server, OpenVZ creates multiple isolated and secured operating system instances known as containers or virtual environments (VEs). Each of these VEs is a stand-alone server that can be rebooted independently and has its own root directory. OpenVZ uses a single kernel shared by all its various VEs. Hence, it is faster and more efficient as it does not have the overhead of a true hypervisor. Figure 10 depicts the OpenVZ architecture [20].

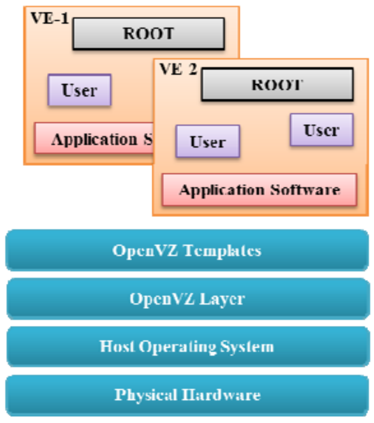


Figure 10. OpenVZ architecture

The OS that OpenVZ supports for the host is limited to all Linux distributions. As for the guest OS; it should be the same OS as the host OS. Hence, in OpenVZ the OS is virtualized instead of hardware virtualization [20].

7.4 Summary of VMM Deployment Solutions

Table 2 summarizes all the various types of security threats in a virtualized environment.

Table 2. Summary of VMM deployment solutions

	Xen	KVM	OpenVZ
Developed by	Uni of Cambridge	Qumranet	Swsoft
Virtualization Type	Hardware-assisted and Para-Virt	Full-Virt	OS-Virt
Compatible Host OS	Linux (few)	Linux	All Linux distros
Compatible Guest OS	Windows, Linux & Solaris	All OS	same as the host OS

8. Conclusion and Future Work

Virtualization is an emerging technology that every organization is keen to embrace due to the various set of benefits it provides and to join the hype cycle with others. Although virtualization brings some value added security features but those are trivial to overcome the many security threats that exist in a virtualized environment.

The core element of the overall virtualization technology is the hypervisor or VMM that is responsible to provide the isolation among the guest VMs and between the guest VMs and the host. The hypervisor is capable of monitoring and controlling the guest VMs, allocating the required physical resources to the guest VMs and ensuring that the entire virtualized environment is behaving well.

Therefore, the overall security of a virtualized environment depends mainly on the level of protection of the hypervisor. With the ease of guest VM migration among different virtualized platforms; there should be appropriate segmentation, change control policies, and security requirement policies in place.

Besides the mentioned importance of the hypervisor the protection of the physical host is also important. Some of the security threats can be solved if the physical host is configured and maintained in a secure manner.

Hence, the future work towards ensuring a secure virtualization environment is through the security of the hypervisor, guest OS and the virtualization infrastructure. Besides having in place a secure monitoring and management solution in order to protect various crucial security aspects of the overall virtualization environment.

This paper presented a range of security threats that exists today in a virtualized environment. It commenced with having an understanding of the anatomy of virtualization, the role of the hypervisor, its different architecture and then highlighted the various forms of virtualization and VMM deployment models. It is crucial to the marketplace to consider the security threats that accompany virtualization technology to have an efficient and effective infrastructure in place.

9. References

[1] D. Barrett, and G. Kipper, *Virtualization and Forensics: A Digital Forensic Investigator's Guide to Virtual Environments,* Syngress - Elsevier Inc., 2010.

[2] J. Hoopes, *Virtualization for Security,* Syngress - Elsevier Inc., 2009.

[3] A. Newman, A. Patrizio, L. Barrett and A. Goldman, *Understanding the Security Implications of Virtualization,* Internet.com Security eBook, a division of QuinStreet Inc., 2010.

[4] M. Rosenblum, and T. Garfinkel, *Virtual Machine Monitors: Current Technology and Future Trends,* IEEE Computer Society, Vol.38, No.5, 2005, pp. 39-47.

[5] J. Sahoo, S. Mohapatra, and R. Lath, *Virtualization: A Survey on Concepts, Taxanomy and Associated Security Issues,* IEEE Computer Society, 2010, pp. 222-226.

[6] K. Scarfone, M. Souppaya, and P. Hoffman, *Guide to Security for Full Virtualization Technologies,* NIST Special Publication, 800-125, 2011.

[7] C. Li, A. Raghunathan, N. Jha, *Secure Virtual Machine Execution under an Untrusted Management OS, roc.* IEEE 3rd Int'l Conf. Cloud Computing - CLOUD'10, Miami, Florida, 5-10 July 2010, pp. 172-179.

[8] S.J. Vaughan-Nichols, *Virtualization Sparks Security Concerns* in *Technology News*, IEEE Computer Society, Vol.41, No.8, 2008, pp. 13-15

[9] L.M. Kaufman, *Can a Trusted Environment Provide Security?* IEEE Computer Society, Vol.8, No.1, 2010, pp. 50-52.

[10] P.A. Karger, and D.R. Safford, *I/O for Virtual Machine Monitors: Security and Performance Issues,* IEEE Computer Society, Vol.6, No.5, 2008, pp. 16-23.

[11] K. Nance, B. Hay, and M. Bishop, *Virtual Machine Introspection: Observation or Interference?,* IEEE Computer Society, Vol.6, No.5, 2008, pp. 32-37.

[12] C. Gebhardt, C. Dallon, and A. Tomlinson, *Seperating Hypervisor Trusted Computing Base Supported by Hardware, Proc. 5th ACM Workshop on Scalable Trusted Computing,* STC'10, Octboer 4, 2010, Chicage, Illinois, USA, pp. 79-84.

[13] M. Price, *The Paradox of Security in Virtual Environments.* IEEE Computer Society, Vol.41, No.11 ,2008 ,pp. 22-28.

[14] E. Ray and E. Schultz, *Virtualization Security*, Proc. 5th Annual Cyber Security and Information Intelligence Research Workshop, CSIIRW'09, April 13-15, 2009, Oak Ridge, Tennessee, USA.

[15] G. Collier, D. Plassman, and M. Pegah, *Virtualization's Next Frontier: Security,* SIGUCCS'07, October 7–10, 2007, Orlando, Florida, USA, pp. 34-36.

[16] M. Christodorescu, R. Sailer, D.L. Schales, D. Sgandurra and D. Zamboni, *Cloud Security is not (just) Virtualization Security*, CCSW'09, November 13, 2009, Chicago, Illinois, USA, pp. 97-102.

[17] A.v. Cleeff, W. Pieters, R. Wieringa and F. van Tiel, *Integrated Assessment and Mitigation of Physical and Digital Security Threats: Case Studies on Virtualization.* Information Security Technical Report, Elsevier, Vol. 16, No. 3-4, 2011, pp. 142-149.

[18] *Security Guidance for Critical Areas of Focus in Cloud Computing,* ver.3.0, CSA, Cloud Security Alliance, 2011.

[19] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, *Xen and the Art of Virtualization,* SOSP'03, October 19-22, 2003, Bolton Landing, New York, USA, pp. 164-177.

[20] J. Che, Y. Yu, *A Synthetical Performance Evalluation of OpenVZ, Xen and KVM,* APSCC, December 6-10, 2010, pp. 587-594.