UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

SECURITY CHALLENGES WITH VIRTUALIZATION

João Carlos Carvalho dos Santos Ramos

MESTRADO EM SEGURANÇA INFORMÁTICA

Dezembro 2009

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

SECURITY CHALLENGES WITH VIRTUALIZATION

João Carlos Carvalho dos Santos Ramos

Orientador
Hans Peter Reiser

MESTRADO EM SEGURANÇA INFORMÁTICA

Dezembro 2009

Resumo

Virtualização é uma palavra em voga no mundo das tecnologias de informação. Com a promessa de reduzir o constante crescimento das infra-estruturas informáticas dentro de um centro de processamento de dados, aliado a outros aspectos importantes como disponibilidade e escalabilidade, as tecnologias de virtualização têm vindo a ganhar popularidade, não só entre os profissionais de tecnologias de informação mas também administradores e directores. No entanto, o aumento da adopção do uso desta tecnologia expõe o sistema a novas preocupações de segurança que normalmente são negligenciadas.

Esta tese apresenta o estado da arte das soluções actualmente mais usadas de virtualização de servidores e também um estudo literário dos vários problemas de segurança das tecnologias de virtualização. Estes problemas não são específicos em termos de produto, e são abordados no âmbito de tecnologias de virtualização. No entanto, nesta tese é feita uma análise de vulnerabilidades de duas das mais conhecidas soluções de virtualização: VMware EXS e Xen. No final, são descritas algumas soluções para melhorar a segurança de acesso a banco online e de comercio electrónico, usando virtualização.

Palavras-chave: Virtualização, Segurança, Ameaças, Banco Online

Abstract

Virtualization is a hype word in the IT world. With the promise to reduce the ever-growing infrastructure inside data centers allied to other important concerns such as availability and scalability, virtualization technology has been gaining popularity not only with IT professionals but also among administrators and directors as well. The increasingly rising rate of the adoption of this technology has exposed these systems to new security concerns which in recent history have been ignored or simply overlooked.

This thesis presents an in depth state of art look at the currently most used server virtualization solutions, as well as a literature study on various security issues found within this virtualization technology. These issues can be applied to all the current virtualization technologies available without focusing on a specific solution. However, we do a vulnerability analysis of two of the most known virtualization solutions: VMware ESX and Xen. Finally, we describe some solutions on how to improve the security of online banking and e-commerce, using virtualization.

Keywords: Virtualization, Security, Threats, Online Banking

Acknowledgments

This thesis represents a lot of hours of searching, reading and writing. However, my effort would be useless without the priceless support from some people.

First, I would like to thank my advisor, Professor Hans Peter Reiser for his support, his precious insight, his knowledge and endless patience.

I would also like to all my colleges who directly or indirectly help me to accomplish this journey. I am truly grateful.

Many thanks to Keith Adams for his knowledge, valuable time and support. To Martim Carbone, Hezi Moore and Benjamin Vetter who kindly gave me a little of their time, helping me with their knowledge.

I would like to thank John Van Berkum and Jason Koltes for their patience in reviewing this thesis.

To my closest friends for their support and understanding on my absence in important moments of their life during this master.

And last but not least, my special thank to my parents for their support, comprehension, patience and love. I will never forget.

Lisboa, December 2009

Contents

List of Figures

List of Tables

Abbreviations

ADM-V	AMD Virtualization
API	Application Programming Interface
ARP	Address Resolution Protocol
AVI	Attack, Vulnerability, Intrusion
BT	Binary Translation
CD	Compact Disc
CMS	Conversational Monitor System
COS	Console Operating System
CPU	Central Processing Unit
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DHCP	Dynamic Host Configuration Protocol
DLL	Dynamic-link library
DMA	Direct Memory Access
DMZ	Demilitarized Zone
DNS	Domain Name System
DoS	Denial of Service
DR	Disaster Recovery
DVD	Digital Versatile Disc
EPT	Extended Page Tables
HCL	Hardware Compatibility List

HIPS	Host Intrusion Prevention System
I/O	Input Output
IDS	Intrusion Detection System
IF	Interrupt-enable Flag
Intel VT	Intel Virtualization Technology
IOMMU	I/O Memory Management Unit
IPS	Intrusion Prevention System
ISA	Instruction Set Architecture
IT	Information Technology
KVM	Kernel-based Virtual Machine
LUN	Logical Unit
MMU	Memory Management Unit
MS	Management Services
NPT	Nested Page Tables
OS	Operating System
PAE	Physical Address Extensions
PKI	Public Key Infrastructure
POPF	Pop Flags
QEMU	Quick Emulator
RAID	Redundant Array of Independent Disks
SATA	Serial Advanced Technology Attachment
SVM	Secure Virtual Machine
SVME	Secure Virtual Machine Enable
TLB	Translation Lookaside Buffer
TPM	Trusted Platform Module
VIP	Virtual Internet Protocol
VLAN	Virtual Local Area Networks

VM	Virtual Machine
VM86	Virtual 8086 mode
VMM	Virtual Machine Monitor
VPN	Virtual Private Network
VSC	Virtualization Service Clients
VT-d	Virtualization Technology for Directed I/O
x86	Family of instruction set architectures based on the Intel 8086
XSS	Cross-site Scripting
YaST	Yet another Setup Tool
ZKPK	Zero Knowledge Proofs of Knowledge

Chapter 1

Introduction

1.1 Motivation

Virtualization has profoundly changed the information technology (IT) industry in different areas such as network, operating systems, applications or storage. Virtualization is no longer a subject only IT people know about it. It has gained space on the administrators and directors vocabulary. Companies have realized that most of their systems were running at ratios of 10 percent or less of utilization, yet these systems continue to require space, power and cooling system as any other machine. Reducing these requirements would have a direct impact in reducing the IT budget and environment cares as the carbon footprint. Virtualization technology was the solution found by many companies, moving this technology into the mainstream. According to a recent IDC survey [2], companies that have deployed virtualization could see a return of investment of 472 percent in less than a year. The increased utilization and consolidation of x86 architectures had an important role for this as well. Many companies use this architecture because it has lower cost compared with others in the market. However, this architecture had historically hardware support issues for virtualization which significant degrade the performance of the virtual machine (VM) comparing with the same system running on a physical host. In order to solve this problem, Intel and AMD implemented architectural extensions to directly support virtualization in hardware. This overcomes the classical virtualization limitations of the x86 architecture, improving important aspects such as performance and scalability making x86 server virtualization a keystone of most IT consolidation projects.

The increasing investment and implementation of virtualization is comparable to the implementation of internet in companies at the end of the last decade. However, in the same way,

security was not the top priority, although IT administrators are now more sensible to this subject. The risks of this new technology are something discussed almost only at security events such as Black Hat and it continues to be out of the focus of many references and consultant companies that implement this technology. Virtualization has been presented to companies as an out-of-the-box solution that companies do not have to worry about, as if it was physical machines with the advantages that the hardware virtualized does not "crash" as the physical ones. There are still some myths to break when talking about virtualization security and these myths happen because, as other myths, there is not much information about it. Some people assume that having, for instance, 4 virtual machines running on a physical machine is the same as having 4 physical machines and so the concerns should be the same (e.g. only install patches on the operating system inside the virtual machine). Perhaps this myth exists because there is a common sense that hypervisors are impenetrable, which is false as we are going to see later in this thesis. Some IT directors are not aware of the security level while using virtualization. For instance, with virtualization it is possible to pause or take a snapshot of a virtual machine that has sensible information as cryptographic keys or password in memory and most companies do not look to these snapshots as critical assets as the running virtual machines.

This thesis presents a study about virtualization, focusing in security problems as the ones described on the previous paragraph. It covers both server and desktop environments and virtualization software. Regarding servers, we have conducted a vulnerability study, comparing two virtualization solutions, while for the desktops we have made a study about how virtualization can be used to improve security for online banking and e-commerce.

1.2 Contributions

Our contributions are as follows:

- We create a document with detailed information regarding virtualization concepts that can be used as reference in courses such as Secure Software Systems. Currently there exist much documentation about virtualization, but normally it focuses only on specific product.

- We analyze the different aspects of virtualization security. There are few books and documents dedicated to this subject and our contribution is to add more value on this subject.

- We make a vulnerability study of two of the most used enterprise virtualization products and compare them in the number of vulnerabilities and the impact those vulnerabilities have.

- We study some solutions which demonstrate that virtualization products can help people facing some of the current security threats such as phishing attacks. According to the website PhishTank, in June 2009 there were validated more than 6000 new phishing attacks [3]. With our contribution, the user would not be affected by these types of attacks and would increase the security of online banking and e-commerce.

1.3 Document Structure

The outline of the remainder of this thesis is as follows: in Section 2 we describe the background of virtualization, doing a resume of its history and describing each component, types and benefits of virtualization; in Section 3 we present the state of art, describing the most current used virtualization products; in Section 4 we do an analysis of security vulnerabilities in virtualization, presenting some of the security breaks that can happen in virtualization and doing a vulnerability analysis of two currently used enterprise virtualization products (VMware ESX and Xen); in Section 5 we present two solutions for phishing attacks and future directions for a third solution that could be implemented by the banks; finally, Section 6 concludes.

Chapter 2

Background

2.1 History

"Virtual machines have finally arrived" [4], said Robert P. Goldberg in 1974. Although this is, in fact, our current reality, it seems to have been the reality of the last 35 years with the slow adoption of virtual machines. Back in 1960s, virtualization was better known as time-sharing. Christopher Strachey, Professor of Computation at Oxford University and the first director of the Programming Research Group published a paper titled “Time Sharing in Large Fast Computers” [5]. His paper, as he refers later in a letter, “was mainly about multi-programming (to avoid waiting for peripherals) although it did envisage this going on at the same time as a programmer was debugging his program at a console. I did not envisage the sort of console system which is now so confusingly called time sharing." [6]. The multi-programming technique allows different users to execute jobs simultaneously. This was possible for one job to take advantage of the CPU, while another is not using the CPU, because it is waiting for an I/O device to store or retrieve data. From among some computers that took advantage of this technique, two are considered part of the evolutionary lineage of virtualization: Atlas and IBM’s M44/44X.

The Atlas computer [7] project was run by the Department of Electrical Engineering at Manchester University and funded by Ferranti Limited. It became operational in 1962 and it was considered the fastest computer in the world, until the release of the CDC 6600 in 1964. The Atlas Computer was the first supercomputer to take advantage of time-sharing, multi-programming and shared peripheral control. It introduced a component called supervisor [8], which managed important resources, such as the processing time of the computer and passed special instructions called extracodes, which would help it to manage the com-

puter environment for the user program's instruction. The name supervisor remembers the actual name hypervisor, and in fact, we can consider supervisor as its roots. Trying to compete with Atlas Computer, IBM created the M44/44X at the IBM Thomas J. Watson Research Center in Yorktown, New York. The central principle of its architecture was a set of virtual machines, one for each user. The M44, the real machine, was a modified version of IBM 7044 and the 44X was each virtual machine with an experimental image of the 7044 [9, 10]. However, the incomplete implementation of the underlying hardware simulation by the M44/44X virtual machine made this project to fail, but soon IBM released its System/360 mainframe.

Virtualization is best known to have been started with the development of the System/360 mainframe, by IBM Corporation. One of the problems presented at the time was the high cost of the machines, which were inefficiently used by people. The main operations were made by using key punches and submitting batch jobs. Engineers were trying to let multiple users to come into the system, by making these batches more interactive. Implementing a time-sharing system at the time for multiple users was not an easy thing to do. For that very reason, IBM's engineering team in Cambridge, Massachusetts presented an idea that would provide each user a virtual machine (VM), with a simple operating system, which only has to support one user. Robert Creasy and Les Comeau from IBM started to developed CP-40 in 1964. This operating system was designed for the System/360 mainframe and was the first step to create virtual machines on these systems. It could support up to fourteen simultaneous virtual machines.

Each virtual machine ran in a mode called "problem state", where privileged instructions (e.g. I/O operations) would cause exceptions, which were intercepted by the control program and simulated. It was replaced in 1965 by CP-67 with the System/360 model 67. This new hypervisor was the first fully virtualized virtual machine operating system and because of this, it is referred in many documentation as the beginning of virtualization. It provided CMS to each mainframe user. CMS stands initially for Cambridge Monitor System, then it was designed as Console Monitor System, but at the end it was renamed to Conversational Monitor System. This CSM was a lightweight single-user operating system supporting time-sharing capabilities. The S/360-67 had a new component called the "Blaauw Box" (designed by Gerrit Blaauw) which implemented virtual memory. CP-67 had the functionality of memory sharing across VMs while providing each user with his own virtual memory space. The advantages were impressive. It was possible to implement test platforms for software development and testing in a much more efficient way. In addition, it increased the debugging efficiency, since it was possible to analyze the virtual memory when the application failed. VM technology stayed as an internal project inside IBM until

1972, when it became a commercial product. One year after, Madnick and Donovan [11] released the first security analysis about virtual machines.

Even so, during these years, VM technology was an important technology in the mainframe world. IBM continued its VM technology on the System/360 and System/370, which appeared in 1970. Nowadays, it continues on the IBM's 64-bit z/Architecture with their z/VM. Until late 90's, some companies released their Virtual Machine, but none with continuous success. In 1998, in California, it was founded VMWare. Their first product was released one year after and has remained one of the most used products on the market, VMWare Workstation [12]. In 2001 their first server edition of VMware ESX 1.0 (Elastic Sky X) was released. Beside VMware, some other companies, such as Sun, Microsoft, Parallels and Citrix, have released their virtualization products that have wide acceptance, some of them as a commercial product, other as an open source solution.

2.2 What is Virtualization

Defining virtualization is not an easy task because as we will see later, there are different types of virtualization and a definition that would be adequate for all is not easy to achieve. Singh [13] describes virtualization as "framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others". However, this definition leaves out cases as network virtualization, application virtualization or storage virtualization. Kiyanclar [14] describes virtualization as "the faithful reproduction of an entire architecture in software, which provides the illusion of a real machine to all software running above it". Most of the definitions are correct if we only consider server virtualization, nevertheless, I adapted Singh's definition saying that:

Definition. *Virtualization is as a framework dividing the resources of the device from the execution environment, allowing environment plurality by using one or more techniques such as time-sharing, emulation, partitioning.*

2.3 Popek and Goldberg

The classical requirements for virtualization provided by Popek and Goldberg, in 1974, on their article "Formal requirements for Virtualizable Third Generation Architectures" [15] is considered the most common qualitative benchmark and can be used as a criterium to judge virtualization variants.The original analysis by Popek and Goldberg was for third-generation computer systems, such as the IBM System/370, the Honeywell 6000, and the Digital PDP-10, but it still holds for present-day machines. On the article, the authors generalized the requirements that the software that provides the abstraction of a virtual machine must guarantee. These requirements can be divided in three:

- Efficiency: The majority of operations must be performed on actual resources rather than being intercepted by the virtualization layer. Sometimes this is referred to as Performance.
- Resource Control: The virtualization layer should be in complete control of the virtualized resources. It should be impossible to bypass the virtualization without control. Sometimes this is referred to as Safety.
- Equivalence: A program running on a virtual resource must exhibit identical behavior as if it was running on the actual resource. Sometimes this is referred to as Fidelity.

According to Golberg et al., the problem at the time that virtual machine monitor (VMM) developers must deal with is to conceive a VMM that would satisfy the three previous conditions when operating within the characteristics of the Instruction Set Architecture (ISA) on the target hardware platform. The ISA can be classified into three groups of instructions:

- Privileged: Instructions that trap only if the processor is in user mode and has no trap if it is in supervisor mode.
- Control Sensitive: Instructions that try to change the configuration of actual resources in the hardware platform
- Behavior: Instructions where the configurations of resources will have an effect on the behavior or results.

VMMs must work with each group of instructions while guaranteeing the three initial requirements of efficiency, resource control and equivalence. In Popek and Goldberg terminology, a VMM should satisfy all three properties, however some of the current VMMs (as

type 2 hypervisor, described in Chapter 2.4 on page 11) only satisfy the Equivalence and Resource Control properties. Moreover, they do it by taking advantage of the emulation, isolation, allocation and encapsulation functions of the VMMs in order to manage the guest operating system and hardware platform.

Emulation represents an important component for a guest operating system that will run on the VM. The VMM has to supply a complete hardware environment (e.g. CPU, Memory, disk) designated as virtual machine. The goal would be that any VM should be available for any application or operating system running inside and should be the most transparent to them, make it unaware that they are sharing hardware resources. Emulation is an important component to satisfy the equivalence requirement.

Another important function of a VMM is to allocate platform resources in an unfailingly way to all the VMs that it manages. In order to have an optimize performance and correct service levels as it is required, platform resources (e.g. network I/O, processing, memory) must be balanced correctly. Allocation is an important component to satisfy resource control and since it is also related with performance then it can be considered as a component to satisfy the efficiency requirement. Isolation, by the other hand, ensures security and reliability of the environment, by making each virtual machine separated and isolated using hardware abstraction. This allows total independence and isolation to each VM from operations on other virtual machines. A fault on one of the VMs should not affect the others on the same VMM, allowing a high level of security and availability.

The Popek and Goldberg requirements did not address encapsulation, which is a component of the VMM process. Encapsulation enables the portability of each software stack (operating system and application), allowing it to be copied or moved from one hardware platform (running the VMM) to another. With this function, it is possible, for instance, to do live migration of running VMs [16].

2.4 CPU Virtualization

The x86 architecture is the most used CPU architecture in enterprise datacenters today, and virtualization can take benefits of that. The Intel 80286 chipset, introduced on February 1982, was the first of the x86 family to provide two main methods of addressing memory: real mode and protected mode. Later, in 1985, with the 80386 chipset, a third mode was introduced called virtual 8086 mode (also called virtual real mode, V86-mode or VM86). The VM86 allowed multiple real mode processes to be run simultaneously while taking full

advantage of the 80386 protection mechanism. Real mode soon became obsolete because it had some disadvantages, such as it was limited to a one megabyte of memory and only one program can be run at a time. The same way, virtual mode was locked in at 16-bit and became obsolete with the high use of 32-bit operating system. Protected mode, by the other hand, is the natural 32-bit environment of the 80386 processor providing many features in order to support multitasking, such as hardware support for virtual memory and segmenting processor.

Protected mode in the x86 family uses 4 privilege levels, numbered from 0 to 3. Sometimes these levels are designated as rings, and the term comes from the MULTICS system [17], in which privilege levels were illustrated as a set of concentric rings. We are going to use the term "ring" as level, because it is a terminology more used. System memory is divided into segments and each segment is assigned and dedicated to a particular ring. The processor uses the privilege ring to decide what actions can be done with the code or data within a segment. As it shows in the Figure 2.1, Ring 0 is considered the innermost ring, which has total control of the hardware while Ring 3 is the outermost ring and has restricted access.

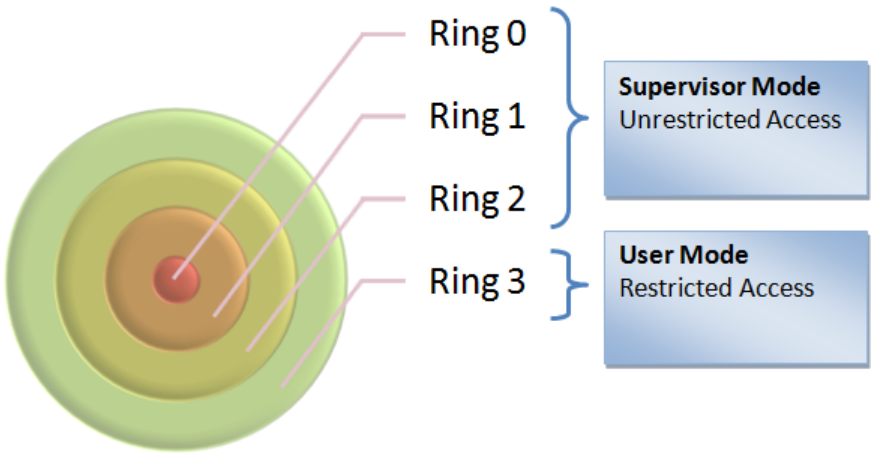


Figure 2.1: *Privilege rings of the x86 architecture. High privilege:0; Low privilege: 3*

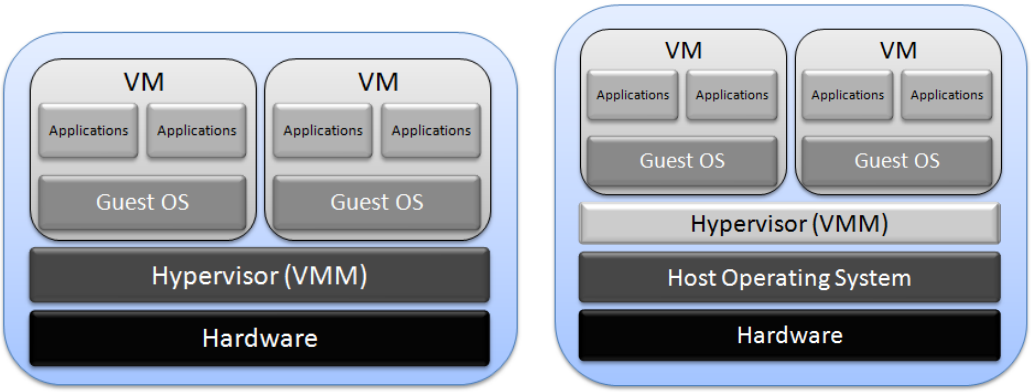
The supervisor mode is the execution mode on an x86 processor with unrestricted access, which enables the executions of all instruction, including I/O and memory management operations, which are privileged instructions. Operating system runs on this supervisor mode, normally on the ring 0. However if this ring is compromised, it will have direct impacts on the ring 3 (user mode). The idea of having isolated ring 0 for each virtualized guest is that if one of the ring 0 of a virtualized guest is affected by, for instance, a failure it will not have impact the ring 0 of others virtualized guest. In order to do this, it is necessary to make this ring 0 closer to the guest, residing in either ring 1 or ring 2 for x86

architectures. However, the further it goes from the real ring 0, the more distant is from executing direct hardware operations, resulting in a loss of performance and independence.

Virtualization moves ring 0 up one level in the privilege rings model and places the virtual machine monitor in the next higher privilege ring. This will be the ring 0 and it is upon this the guest operating systems runs, while the VMM handles the interaction with the underlying hardware platform. VMMs can be classified in two types:

- Type 1: This type is also called as native or bare-metal because the hypervisor software runs on top of the host's hardware on the real ring 0 (Figure 2.2a). A guest operating system thus runs on another level above the hypervisor, allowing for true isolation of each virtual machine. This is the classic VM architecture. An example of this implementation is the VMware ESX Server and Xen.

- Type 2: This type is also called hosted VMM because the hypervisor software runs within a normal host operating system already installed, usually in ring 3 (Figure 2.2b). This type of VMM has a lower performance than the other type because factors as calls to the hardware must traverse many diverse layers before the operations are returned to the guest operating system. Examples of this implementation include VMware Workstation, Sun VirtualBox and Parallels Workstations.



(a) *Type 1 Hypervisor/VMM* **(b)** *Type 2 Hypervisor/VMM*

Figure 2.2: *The two types of Hypervisors/VMMs*

It is important to clarify that the term Hypervisor has the same meaning as VMM.

Privileged instructions trap when they are called from user mode. When called from supervisor mode (kernel mode), they do not trap. A trap passes control to a trap-handler, located in the kernel, so that processor mode changes. A sensitive instruction is an instruction

that reads from or writes to memory locations or sensitive registers. All sensitive instructions have to trap on a virtualizable architecture and therefore, in this context, sensitive instructions can be seen as subset of privileged instructions. A VM cannot access hardware directly without passing the hypervisor, which is responsible for maintaining control over sensitive instruction and the hardware. When a VM tries to access sensitive data, the instruction is trapped and control is passed to the hypervisor. This happens because VMs runs in user mode but if the guest OS attempts to access sensitive data, a trap will occur. Then, the hypervisor, which is running in supervisor mode, it will catch this trap, inspect the state of the guest OS that cause it and emulate the behavior that would occur if the guest OS was running on a real machine. The hypervisor will then resume the VM, allowing the executing to continue. This method is called "trap and emulate".

The Popek & Goldberg requirements are satisfied by this approach as long as the processor is guaranteed to trap whenever any privileged operation is attempted in user mode. However x86 architectures does not guarantee this, since there are 17 sensitive non privileged instructions that disables "trap and emulate". The control is not passed to the hypervisor when these sensitive instructions are called and so, the hypervisor cannot emulate the expected behavior.

According to Smith and Nair [18], the 17 sensitive non privileged instructions fall into two categories:

- Protection system references: These instructions reference address relocation system, memory system or storage protection system. The problem is the possibility of a virtual machine to access locations outside its virtual memory. An example presented by the authors is the MOVE instruction, which moves a value from general-purpose register to the CS register, the control register that specified the current privilege ring number in bits. An instruction such as `move ax,cs`, when executed in the user mode disallows the CS register to be loaded. This happens to offer some protection, but which makes it not well virtualizable is that instead of generate a trap, the instruction generates a no-op. The instructions in this category are:
  - CALL: Call procedure
  - JMP: Jump
  - INT n: Software Interrupt
  - LAR: Load access rights
  - LSL: Load segment limit

    - MOV: Move data between general-purpose registers or between memory and general-purpose/segment registers. It can also move immediates to general-purpose registers
    - POP: Pop off of stack
    - PUSH: Push onto stack
    - RET: Return
    - STR: Store task register
    - VERR: Verify segment for reading
    - VERW: Verify segment for writing

- Sensitive register instructions: These instructions read or change resource-related registers and/or memory locations, such as a clock register or interrupt registers. The authors detail the example of the POPF instruction. POPF pops the flag registers from a stack held in memory. One of the flag registers is the interrupt-enable flag (IF), which can only be modified in privilege mode. The problem happens when the guest OS requires that the IF bit be changed and since it is running in the user mode under the VM, then the IF bit cannot be changed. This can lead the guest OS to take erroneous action because the flag bit was not set as expected. These instructions are:
    - PUSHF: Push EFLAGS onto stack
    - POPF: Pop EFLAGS from stack
    - SGDT: Store global descriptor table register
    - SIDT: Store interrupt descriptor table register
    - SLDT: Store local descriptor table register

On the paper, the authors describe 18 instructions, which are the ones specified above and SMSW (store machine status word). However, in literature about virtualization it is only mentioned 17 sensitive, unprivileged instructions are considered. We believe the SMSW instruction was deprecated since it is only provided for backwards compatibility with the Intel 286 processor.

On the x86 architecture, the operating system normally runs in ring 0, because it is the highest privilege level, which provides total access to platform resources, like CPU and memory. Individual applications usually run in ring 3 in User Mode, with restricted access. When we add virtualization on this scheme, what normally happens is that VMM runs in

ring 0, since it must have privileged control of platform resources and the guest operating system goes to ring 1 or ring 3.

In order to overcome the limitation of implement CPU virtualization on x86 architectures, some techniques such as direct execution combined with fast binary translation and paravirtualization [19].

Binary translation (BT) is not a new technique and can be use for various purposes such as migrations between different architectures [20, 21]. The combination of direct execution with BT was an idea developed by VMware to be used for CPU virtualization. This technique allows running supervisor mode code controlled by the binary translator. The translator replaces the privileged code into a similar block, patching the sensitive, unprivileged instructions. This translated block can then run directly on the CPU and they are cached by the BT system in a trace cache so they can be used on subsequent executions. Using BT, only the sensitive instructions like POPF are replaced while the normal instructions are executed unchanged. This binary translation is only applied when the code first executes [19].

Paravirtualization uses a different approach to overcome the x86 virtualization issue. With paravirtualization, the nonvirtualizable instructions are replaces with virtualizable equivalent ones. This requires the guest OS to be changed although most of the normal applications remain unchanged. One difference with the BT approach is that in paravirtualization, the guest OS knows that it is running in a virtual environment, while using BT the guest OS have the illusion that is running on a real machine. The paravirtual hypervisor is smaller and easier to implement containing only a small interface for the 17 sensitive, unprivileged instructions and then is more trustworthy than one using BT and similar to a VMM using "trap and emulate".

The second generation of hardware virtualization (Intel VT [22] and AMD-V [23]) was designed with the goal of eliminate the need for BT and paravirtualization on the x86 architecture. The CPU creates containers and introduces new modes of operations that can distinguish if the CPU is real or virtual. VMM runs the highest privilege container which is commonly designed as ring -1. Guests run within a lower privileged container, although conceptually is considered a ring 0. This allows guest OS to run at their normal ring and only leaving when the guest tries to execute a privilege or sensitive instructions which will trap to the VMM. This can be seen as "trap and emulate" and therefore, all security issues related with it can be also applied to hardware virtualization.

2.5 Memory Virtualization

Normal operating system use page tables to translate virtual addresses into physical addresses. Virtual machines brought new challenges regarding memory virtualization since memory is going to be shared although isolation as to be guaranteed. We can consider three classes of addresses on a virtualized system:

- Virtual addresses, which are the same as the ones used by a conventional OS
- Guest physical address
- Machine memory

Guest operating systems maintain page tables that translate from virtual to pseudo-physical addresses, and hypervisor maintains separate shadow page tables that translate from virtual addresses to machine addresses [24].

The recent x86 CPUs support memory in hardware. Translation from virtual to physical addresses is performed by the memory management unit (MMU) and the most used parts of the page tables are cached in the translation lookaside buffer (TLB). Guest OS sees page tables, which run on an emulated MMU. These tables provide the guest OS with the illusion that it can translate the virtual guest OS addresses into real physical addresses, but it is the hypervisor that deals with it. The real page table is the shadow page table used to translate the virtual addresses of the guest OS into the real physical pages.

The classic implementation of hypervisors maintains a shadow page table, which allows to control what page of the machine's memory is available to a virtual machine. Just like in a traditional operating system's virtual memory subsystem, when the memory allocated to VMs exceed the host physical memory size, the hypervisor can page the VM to the disk. This way, the hypervisor can dynamically control how much memory each VM receives.

However, the hypervisor's virtual memory system does not have the perception of which pages are good for paging out but the guest OS should have because it can identify, for instance, that the process that created a page has exited and so nothing will access to that page. This page would not be a good candidate for paging out, but since the hypervisor's virtual memory system has no clue about that, it might page out that page.

To deal with this problem, VMware's ESX Server created a mechanism which allows the hypervisor to ask to the guest OS for the pages it can swap out, using a balloon process [24]. A balloon process runs inside a guest OS and communicates with the hypervisor.

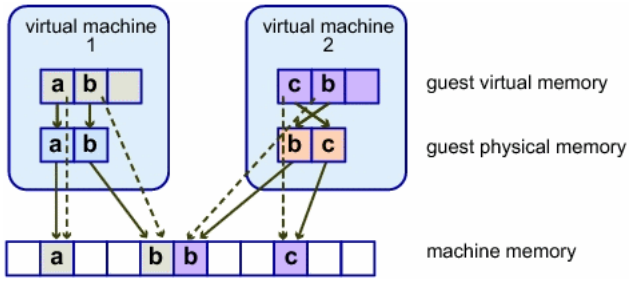


Figure 2.3: *ESX server memory mapping*

When the hypervisor needs to take memory away from the VM, it communicates with the balloon process to "inflate" the process, allocating more memory. This will force the guest OS to select the pages to give to the balloon process, which will be passed to the hypervisor for reallocation, but also it will force the guest OS to page memory to the virtual disk.

VMware engineers also develop a mechanism that would decrease the memory used by several VMs running the same version of an operating system. They identify that different virtual machines running the same operating system will produce redundant copies of code and data stored in memory that could be shared among them. To address this, they have designed a content-based page sharing system, analyzing if the contents of physical pages are identical. When such content is identified, the hypervisor modifies the VM's shadow page table to point to only a single copy and freeing the redundant copy. If the content is changed, then the hypervisor provides the VM with its own copy of the page, in a copy-on-write page-sharing scheme [19]. However, this can have some drawbacks. In order to detect duplicate pages, it is necessary to periodically scan the memory and build a list of page fingerprints, which will be used to compare page contents. VMware ESX scans frequency is set by default to once an hour (with the maximum of six times per hour [25]), but this means that short-lived sharing opportunities will be missed [26].

Without shadow pages, it would be necessary to translate guest virtual memory into guest physical memory and then translate this one into the real machine memory. The shadow page table avoids the double bookkeeping by making the MMU work with the guest virtual memory to real physical memory page table.

Figure 2.3 illustrates the VMware ESX server implementation of memory virtualization. In this figure, the boxes represent the pages and the arrows show the different memory mappings. As we can see, there are arrows from the guest virtual memory to the guest physical memory which represents the mapping maintained by the page tables in the guest OS. The arrows from guest physical to machine memory represent the mapping maintained by the hypervisor, while the dashed arrows show the mapping from guest virtual memory

to machine memory in the shadow page tables which is also maintained by the hypervisor.

Nevertheless, there is a performance issue because each update of the guest OS page tables forces a shadow page table bookkeeping. The second generation of hardware virtualization (Intel VT and AMD-V) partly solves this problem with their AMD's Nested Page Tables (NPT) and Intel's Extended Page Tables (EPT).

When using nested paging, the CPU caches both guest virtual and physical memory as the guest physical memory to real physical memory transition in the TLB. The TLB has a new tag called Address Space Identifier (ASID) which allows to keep track of which TLB entry belongs to which VM. This way, entries of different virtual machines can coexist in the TLB at the same time. Using nested paging has an increment importance if there is being used multiple virtual CPU per VM, because they have to sync the page tables many times with direct impact on the shadow page table update. With NPT, the CPU only has to synchronize TLBs as it would happen in a non-virtualized environment.

2.6 Device and I/O Virtualization

The VMM virtualizes the physical hardware and allows each virtual machine a set of customizable virtual devices. Most of this virtualized I/O requires software drivers that run on the host operating system to access the real hardware. If it is a type 2 hypervisor, then it will use the device drivers already in the host OS, otherwise it may be necessary to develop its own device drivers for the hardware on the machine, like in the case of VMware ESX. Emulation is normally used for a VMM to handle I/O devices, and it is the VMM the responsible to implement a software model of the I/O device, making believe the guest OS that it is communicating to a hardware device, when is communicating with a software model. The I/O virtualization may provide to the guest, virtual hardware that does not exist in the real hardware, for instance, emulating an IDE hard disk when the real hardware is SATA. The direct memory access (DMA) has problems when used with virtual machines. The DMA controller can write to the entire physical memory instead of only the memory assigned to the guest OS. In order to deal with this problem, Intel and AMD added I/O Memory Management Unit (IOMMU). With IOMMU it is possible to restrict which physical address a device may access.

2.7 Types of Virtualization

When people talk about virtualization, normally they are talking about server virtualization. However, information technology has other forms of virtualization commonly known and used by other groups of people. For some, virtualization means storage virtualization, or network virtualization or even application virtualization. Although my thesis will only concern about server virtualization, I will do a brief explanation of each one.

2.7.1 Server Virtualization

There are many different implementations of server virtualization on, and for a big range of CPU platforms and architectures. Informally, server virtualization can be seen as creating many virtual systems within a single physical system. To accomplish this, we can take three approaches: physical layer, virtualization layer and OS layer. Hardware partitioning divides a single physical server into partitions where each partition is able to run an operating system while hypervisor places a layer of software between the physical hardware and the multiple operating systems that will share the same physical hardware.

Physical layer:

- Hardware partitioning: The server is physically segmented into distinct smaller systems that will act as a physically independent and self-contained server. Normally each of these smaller systems has their own CPUs, OS, boot area, memory and network resources. The implementation of this technique includes Static Hard Partitioning, vPar, nPar among others [27].

Virtualization layer: Hypervisor technology can be organized in some distinct categories:

- Full virtualization: Allows virtual infrastructures to run unmodified operating systems in isolation. The operating system running inside the virtual machine is called guest operating system. This approach was pioneered in 1967 with IBM CP-40 and CP-67, predecessors of VM family. In order to implement full virtualization, it is necessary a full combination of hardware and software, however not all architectures have hardware to support virtualization. It was not possible on IBM System/370 until 1972 and it was not natively possible in the x86 architecture [28] until 2005 when Intel and AMD added the hardware virtualization extensions (Intel VT and AMD-V respectively). Nevertheless, many companies tried to accomplish full virtualization

on x86 architecture even before Intel VT and AMD-V additions. VMware uses a combination of direct execution with binary translation techniques [29] to accomplish full virtualization of an x86 system. This provides full disassociation of the guest OS from the underlying hardware by the virtualization layer. As depicted in Figure 2.4, the OS requests that needs to interact with the hardware are needs to be translated by the VMM, replacing nonvirtualizable instructions with new sequences of instructions, which have the same result on the virtual hardware , while the user's applications are directly executed on the processor for high performance virtualization.

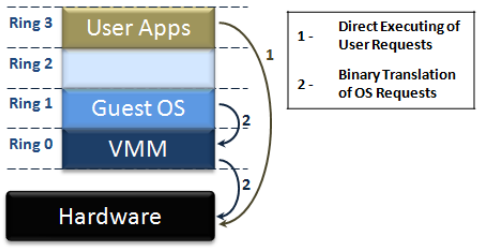


Figure 2.4: *The binary translation approach to x86 virtualization used by VMware*

- Paravirtualization: modifies the guest kernel system in order to purge the necessity of binary translation. It has the advantage of higher performance but has the drawback of needing a modified operating system kernel. The fact the virtual platform is not identical to the real hardware, it makes necessary for the operating system to be ported to the abstracted machine interface. This could be seen as a violation of the Goldberg's equivalence requirements, because the architecture-dependent part of the operating system kernel needs to be changed [30]. The non virtualizable instructions are replaced with hypercalls that communicate directly with the virtualization layer hypervisor. The architecture independent part and the entire user mode software stack stay unmodified.

- Emulation: Sometimes people confuse emulation with full virtualization. Although both run unmodified guest operating systems, they are both very different. In emulation, the virtual machine simulates the entire hardware set needed to run the unmodified guest OS normally for a completely different hardware architecture. There are some utilities for this technique. For instance, it allows to develop programs and operating systems for new hardware design before the hardware is physical available. It also allows, as we are going to see later, to run an unmodified version of

Microsoft Windows in Power PC architecture. Emulation does not satisfy Popek and Goldberg's efficiency requirement.

Operating System layer:

- Operating System-Level Virtualization: This is a technology that virtualizes servers at the OS (kernel) layer. The physical server and instance of the OS is virtualized into multiple isolated partitions. Each of them will look like a real server, from the point of view of its owner. The OS kernel will run a single OS and provide its functionality to each of the partitions. On Unix systems, this technology can be seen as an advanced extension of the standard chroot mechanism. Operating system-level virtualization has the disadvantage that strong isolation is difficult to implement. The implementation of this technique include Solaris Container/Zone, FreeBSD Jails, Aix Workload Partitions, Parallels Virtuozzo Containers, Linux VServers and OpenVZ.

2.7.2 Storage Virtualization

Storage virtualization has been around for a number of years. It has beginning with the use of redundant array of independent disks (RAID). Using RAID it is possible to logically group physical disks and present those groupings as a virtual disk to the OS. Using storage virtualization it is possible to merge physical storage from many devices which will appear as a single storage pool. This storage can be classified as direct attached storage (DAS), network attached storage (NAS) and storage area network (SAN). They can be linked using Fibre Channel, Fibre Channel and Internet Small Computer Systems Interface (iSCSI), Fibre Channel on Ethernet or Network File System (NFS). Storage virtualization it is not a requirement for server virtualization but its use provides benefits since it can rely on the assignation of a logical unit (LUN) of storage, but provisioning it only when needed. For instance, if we have a LUN of 500 GB but we are only using 20GB, then only 20GB of actual storage is provisioned. This reduces the cost of storage, since we only use what is needed. Storage virtualization brings also help to the storage administrator, since it is easier to manage tasks as backup, archiving or recovery.

2.7.3 Network Virtualization

When people talk about network virtualization, probably the first thing that comes to their minds is Virtual Private Network (VPN) or perhaps Virtual Local Area Networks (VLAN).

However there is more when we talk about network virtualization. The most used network virtualizations are:

- Virtual LAN (VLAN): Defined in the IEEE 802.1Q standard, is a method of creating independent networks using a shared physical network. They are used to logically segment broadcast domains and control the interaction between different network segments. VLANS is a common feature in all modern Ethernet switches, allowing to create multiple virtual networks, which isolates each segment from the others. All the available resources are segments and allocated to each of these segments. Therefore, VLAN is a safe method of creating independent or isolate logical networks within a shared physical network.

- Virtual IP (VIP): A VIP is an IP address that is not associated to a specific computer or network interface, but is normally assigned to a network device that is in-path of the network traffic. Incoming packets are sent to the VIP but are redirected to the actual network interface of the receiving host or hosts. It is used in solutions like High-Available and Load-Balancing, where multiple systems have a common application, and they are able to receiving the traffic as redirected by the network device.

- Virtual Private Network (VPN): It is a private communication network that uses public network, such as Internet. Its purpose is to guarantee confidentiality on an unsecured network channel, from one site to another. It is normally used as a means of extending remote employee home networks to the company network. This is normally done by using special software (as Cisco VPN Client), but after the connection being established, all the interaction with the other resources on the network is handled as if the computer was physically connected to the same network, although this depends of the way security policies are applied.

2.7.4 Application Virtualization

Desktop applications have always been a headache for administrators. There were always problems with missing or wrong versions of DLLs, or wrong registry keys or other programs (like antivirus software) that would interfere with their behavior. The web applications and dynamically updated applications have been very popular, because it can be a workaround to most of desktop application problems. However, not everything can be converted into a web application, and sometimes it is necessary to run application on the

user side. Application virtualization tries to solve the desktop application problem by encapsulating a virtualization layer and all resources needed for the application to be run on a user's desktop. The virtualization layer is the one responsible to make the channel between the application and the operating system, and so, it is possible to have many isolated applications running, and even different versions of the same application without interfering between them[31]. Java Virtual Machine is an example of Application Virtualization.

2.8 Benefits of Virtualization

Nowadays, virtualization is in the vanguard, helping companies to take advantage of two important properties of virtualization: scalability and management. It can bring many benefits and there are many reasons for its application. We are going to see some of the keys benefits of virtualization.

One of the benefits of virtualization has historical reasons and is related with the misused of servers, mainly because of Microsoft Windows NT Server. This operating system started to be used in datacenters back in 1990 although it was a hard battle in the beginning to be accepted as a good operating system for enterprise datacenters. Today we know it was a battle won and Microsoft has it share on most datacenters. However, Windows NT was a monolithic OS and when an application freeze it would often freeze the OS causing the well known Blue Screen of Death. Administrator started to apply the philosophy of single-purpose servers, running a single application per server. This would prevent that if one application fails, causing the operating system to fail, it would not disrupt any other application running on another server. For these reason, each time it was necessary a new business application, it was also necessary another server to be used. Over time, Microsoft solved the monolithic problem but administrator's habits were already created. However, it is not only Microsoft's fault. Many software vendors required their application to be isolated so they can support them. In addition, security taught us that the less application we have installed in a machine, the smaller will be the attack surface.

For this reason, companies have realized that most of their systems were running at ratios of 10 percent or less of utilization, yet these systems continue to require space, power and cooling system as any other machine. Instead of having some servers for their principal services (e.g. email, stock programs), they can invest in a better server and consolidate all those services in separated virtual machines. With that, they gain scalability, since they can upgrade their virtual machine without necessary upgrade the physical machine. They gain security, since they have a more control environment and easier to backup and restore,

which also is related with management. However there is a drawback which has to be balanced. Since all the main services of the company are installed on one only server (or few), this is considered a single point of failure.

High availability is essential for corporate environment where availability of their services is crucial to the success of their business. One of the advantages of using virtual machines is that it can be restored very easily (one or very few files) and so, if updated backups exists, then the IT manager can simply restore that file on another machine. Normally, it is recommended hot standby virtual machines with at least two servers. Taking the example of email server and stock programs, on the server A would be installed the email server running on a VM (VM.A.1.ori) and an updated offline copy of the VM (VM.B.1.bck) with the stock program running on the server B. And the opposite on server B, which would be an update offline backup of the email server that is running on server A (VM.A.1.bck) and the stock program running on a VM (VM.B.1.ori). If any problem happens to one of the servers, there would be another server that could temporary support both VMs. This example shows also another benefit of virtual machines, which is disaster recovery (DR). Some companies have their disaster recovery center in another geographic location and applying a hot standby allows them to easily replicate the VM to their DR center and when needed, be able to quickly make the services available. Another advantage of virtualization on disaster recovery is when facing an exploit that can compromise a server, be able to use a VM trustworthy (i.e. not infected) baseline installation of the affected system, patch them and turn into production, leaving the infected system for analysis and evaluation.

Virtual machines offer a perfect environment for development and research. According to Silberschatz et al. [32], changing an operating system is a difficult task because they are complex programs and since they executes in kernel mode, the impact of changing a pointer can destroy the entire file system. Therefore, it is necessary to test all changes to the operating system. With virtualization, the system programmer can have their own virtual machine and system development or test is made on those virtual machines instead of on a physical machine. This reduce the system development time and cost, increasing the productivity.

Another benefit of virtualization is the possibility of having multiple operating systems running, even those systems that are obsolete and cannot be utilized by the newer hardware resources, may be supported to run on a virtual machine.

It is also useful for testing software solutions. Using virtualization, a company can try some solutions without the necessity of using many real servers. The same way it is useful for software developers to simulate the production environment the best they can using

virtual machines, and that way, debugging their application in an almost real environment. In addition to testing new solutions, virtual machines are useful to test new patches before applying them into production systems.

Virtual Machines are also very useful for forensic team research. It is possible to clone a potentially compromised host into a VM and do further investigation without the need of the physical machine. The investigation team can also take advantage of snapshots to return to a previous state. The same can be said for malware investigation team. It can be very useful to use a VM since it guarantees isolation and to have the ability to use the snapshot function. However, malware does not always have the same behavior inside a VM as on a real machine.

There are other benefits of using virtualization. Honeypots or honeynets [33, 34] are intrinsically related with server virtualizations and are normally used by organizations to attract possible attackers. Honeypots can be described as servers with fake information in an isolated network, simulating a real DMZ or Intranet in order to analyze the attackers' behavior. The logs generated on these systems is much less than the generated every day on other security systems as firewalls, IDS and IPS alerts or even system logs, but their value is normally high, because it include most likely scans, probes and attacks [35]. Detect intrusions using hypervisor is purposed on some papers [36, 37, 38, 39, 40] as a way to alert for rootkits running on the guest OS.

Chapter 3

State of Art

According to a study from Gartner [41], in August of 2009 about 16 percent of workloads are running in virtual machines, where VMware has 89% of the installation base followed by Microsoft with 8%. Analysts expect that the installed base of virtual machines will grow 10 times in the next four years. However, the expectation is that VMware will lose some of the market for Microsoft, Citrix and Red Hat. In the same study, Gartner expects that, in 2012, VMware only has 65% of the installation base, while Microsoft would achieve 27 percent of market followed by Citrix with 6 percent.

In this chapter we present the most current used virtualization products, describing the main components of each one and how they work.

3.1 VMware

Founded in 1998 by Diane Greene and Dr. Mendel Rosenblum along with two students from Stanford University and a colleague from Berkley, VMware is a well known company on the x86 virtualization market. In October of the same year, these five founders filed for a patent regarding new virtualization techniques. These techniques were based on a project called SimOS conducted at Stanford University. The U.S. Patent 6,397,242 was awarded on May 28, 2002 [42].

Their first product was VMware Workstation with the first version being released on February 8, 1999 for Windows and Linux. This is one of the most successful products from VMware for desktop and stays as a commercial product with its current version 6.5.x. VMware Workstation is a type 2 hypervisor, supported on top of a Host OS – either Windows

or Linux – and able to create virtual machines for a variety of guests OS, such as Solaris x86, Netware, FreeBSD, Windows and Linux.

In late 2000, they released their first version of the server virtualization platform called VMware GSX Server. As in the case of VMware Workstation, this GSX Server needs to be installed on top of an existing Windows or Linux operating system. In 2006, VMware GSX Server was renamed to VMware Server and it is now released as freeware.

In 2001, VMware release their first version of Elastic Sky X (ESX) [43]. This is their first server product with an approach different to that of the workstation version. In this case, VMware ESX does not require a host OS, but instead it has its own native hypervisor on a bare-metal system. This had the drawback to support less hardware but had the advantage of requiring less overhead to host each virtual system. On May 2009, it was released VMware ESX v.4.0. Figure 3.1 shows the many releases of VMware ESX Server. The interval between the major releases and the minor release has been growing, from 14 months between v.1.0 and v.1.5 to 26 months between 3.0 and 3.5. This is normally due the fact minor releases are the accumulation of many patches released since the major release, but also with added features.

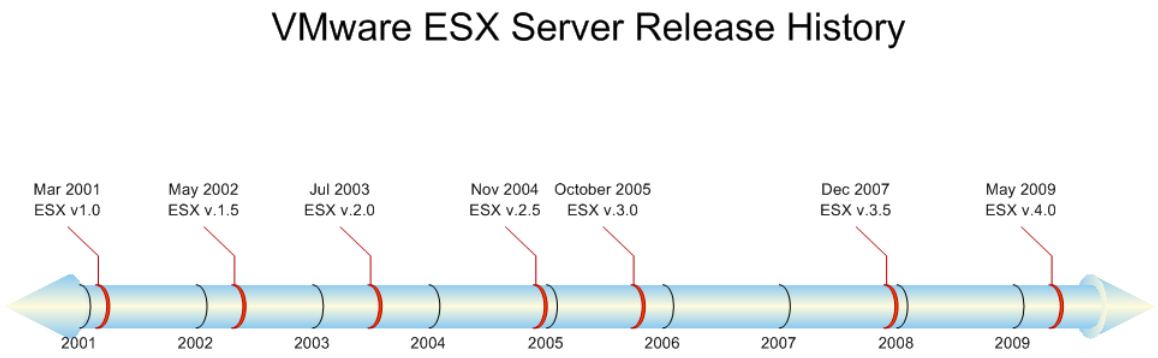


Figure 3.1: *VMware ESX server release history*

3.1.1 The ESX Platform

The core of VMware ESX has three main modules capable of regulating CPU affinity, memory allocation and oversubscription, network bandwidth throttling and I/O bandwidth control. Along with these, Virtual Machine File System completes the VMware ESX base platform. The three primary components of VMware ESX are:

Physical Host Server: This is related with the physical host server where VMware ESX runs on. VMware had a Hardware Compatibility List (HCL) that includes some of

the main servers' brands on the market, such as Dell, Hewlett-Packard, IBM among others. For support reasons, it is convenient to use ESX only on the servers reported on HCL.

VMkernel: The VMkernel is the center of the VMware ESX hypervisor, and it is a high performance operating system developed by VMware to run on the ESX host server. Although it has some similarities, it is not a Linux kernel. It does not share the Linux data structures or symbols neither depend on the Linux kernel for any services. The VMware ESX has a modular approach, loading various system modules. However, the version 3.x presents some innovations regarding this modular approach, such as allowing new devices to be added without the need of recompile the VMkernel [44].

The Console Operating System (COS): The service console has been upgrade from being based on a variant of Red Hat version 7.2 with ESX 2.x to Red Hat Enterprise Linux 3, Update 6 for ESX 3.0 [45] and Update 8 for ESX 3.5 [46]. The COS does not interact with system hardware, that is a job made by VMKernel. The COS function is to provide the executing environment for monitoring and administrating ESX. On versions before ESX 3.0, VMkernel would only load after COS was fully booted. In ESX 3.0 this was changed and now VMkernel runs before than COS. In fact, COS runs within a specialized VM with more privilege than the normal VMs.

Some other important components of VMware ESX are:

Virtual Machine File System (VMFS): The VMFS is a high performance cluster file system created by VMware. VMFS has many advantages compared to conventional file system. One of those is the fact that up to 32 ESX Servers [47] can concurrently read and write to the same storage by using per-file lock. It has some other important security features such as the fact that it allows live migration of powered-on virtual machines from a host server to another and by using distributed journaling, it is possible to recover VMs faster and more reliable in a case of a server failure. However, according to Scott Davis [48] from VMware, the choice between VMFS and NFS will depend of what type of storage infrastructure the organization is familiar with. If it uses block based storage, then it is recommended to use VMFS, but if it is already using a Network-attached storage (NAS), then it would be recommended to use NFS instead.

VirtualCenter: The VMware VirtualCenter is the management console used to control the virtualized enterprise environments. Is provides services such as access control, performance monitoring, and configuration.

On December 28, 2004, VMware submitted VMware ESX Server 2.5.0 and VirtualCenter 1.2.0 Target Of Evaluation (TOE) to Common Criteria for Evaluation Assurance Level (EAL) 2 conformance certification under the Operating Systems category. On March 27, 2006, Common Criteria confirmed the certification at EAL2. The evaluation was carried out in accordance with the US Common Criteria Evaluation and Validation Scheme (CCEVS) process. The criteria was judged as described in the Common Criteria for Information Technology Security Evaluation, Version 2.2 and International Interpretations. The TOE is Common Criteria 2.2 Part 2 extended and Part 3 conformant.

On May 20, 2008, VMware VI3 (ESX Server 3.0.2 & VirtualCenter 2.0.2) achieved Common Criteria certification at EAL4+ under the Canadian Common Criteria Evaluation and Certification Scheme (CCS). EAL4+ is the highest assurance level that is recognized globally by all signatories under the Common Criteria Recognition Agreement (CCRA) and achieved the lever at which it is likely to be economically feasible to retrofit to an existing product line. VMware is the only company having an x86 virtualization product to successfully complete the Common Criteria certification process. Currently, VMware ESXi 3.5 and VirtualCenter 2.5 and VMware Infrastructure 3.5 (VMware ESX Server 3.5 and VirtualCenter 2.5) are in process of certification.

Although it was not possible to confirm, there are some informations on the Internet reporting that VMware ESX 2.0 and ESX 3.0 have 100,000 lines of code. During my thesis research, I have questioned some of the key people in the virtualization world. One of the person I have exchanged emails regarding VMware was Keith Adams, an ex Senior Staff Engineer at VMware currently working on Facebook. I have questioned Keith Adams about if he thinks this is an approximated number. In his own opinion, he believes the line count for the VMkernel is higher, since it includes drivers for many storage and networking devices. Nevertheless, he could not confirm me this number, because it was information he did not have.

3.1.2 VMware ESXi

VMware ESXi was announced during VMworld 2007 and it is an integrated version of VMware ESX but without the COS. This is important in terms of security. VMware ESXi had RHEL-based COS replaced with BusyBox, which is a single binary that provides a minimal set of services. Many of the security patches for VMware ESX where related with security vulnerabilities on the Service Console (e.g. CVE-2009-1185, CVE-2009-0034, CVE-2009-0846). It is tempting to say that removing this component from VMware

ESXi will result in less security patches to be applied and so fewer downtimes for security patching. Only when this product gets more mature and more used we can see if this is correct or not.

Regarding my question if the core of VMware ESX and ESXi are similar, Keith answered me by saying that "the VMkernel and VMM are almost byte-for-byte identical".

3.2 Xen

The Xen project was first described in the paper "Xen and the Art of Virtualization" presented at SOSP in 2003 [49]. It was a project originally developed by the System Reseach Group at the University of Cambridge Computer Laboratory and was part of the XenoServers projects [50] which had the goal of build a public infrastructure for global-scale service deployment. The first public release of Xen 1.0 was made in October of 2003 and the project had a good evolution all over the years, getting maturity in virtualizing resources like CPU, memory, disk and network. For that, Xen had many project contributors, including AMD, HP, Intel, Novell, RedHat and XenSource.

XenSource, Inc. was a company founded by Ian Pratt, senior lecturer at Cambridge and lead of the Xen project, with the goal of supporting and developing the open source Xen project and to create a commercial enterprise version of the software. In 2005, XenSource release Xen 3.0, which was the first enterprise-class of Xen, supporting up to 32 processors. It was also the first version with built-in support for Intel's VT and with support for Physical Address Extensions (PAE) to support 32-bit host servers with more than 4GB of memory. At the time, ADM-V was not released yet, but eventually it was supported too. In order to compete more directly with VMware, XenSource released XenOptimizer, an integrated virtual infrastructure management platform competing with VMware's VirtualCenter and VMotion technologies.

However, XenSource was still missing the point, providing separated products when VMware was taking the market with their consolidated product ESX. In order to change this, XenSource released their first version of XenEnterprise 3.0, a product based on Xen v.3.0.3 and a directly compete with VMware ESX Server. This version had two important features: It included a new management and monitoring console based on XenOptimizer and it was the first Xen product supporting Windows guest operating systems. To this contributed the partnership made between XenSource and Microsoft.

The year of 2007 was very interesting regarding Xen. In August 2007, XenSource announ-

ced the release of XenEnterprise v4. This new version was more stable and added some new features, which made it become closer in feature parity to VMware ESX, but at less than half the cost. However, another important event happened in the world of Xen. At the end of October 2007, Citrix completed its acquisition of XenSource [51] and the Xen project moved to http://www.xen.org/.

Xen is an open-source hypervisor for both 32 and 64 bit process architecture that runs on top of the bare-metal. It allows to securely and efficiently run several virtual guest OS on the same host at the same time. Xen as many features as:

- Near native performance on the virtual machines
- Full support on x86 (32-bit) with and without Physical Address Extension (PAE)
- Full support on x86 with 64 bit extensions
- Support for almost all hardware with Linux drivers available
- Live migration of running virtual machines between two physical hosts with zero downtime
- Support of Hardware Virtualization extensions from Intel (Intel-VT) and AMD (AMD-V), allowing unmodified guest operating systems.

Initially, Xen would only support one mode of virtualization called paravirtualization (see Section 2.7.1 on page 18). In this mode, the guest OS must be modified and the kernel recompiled to support proper interaction with the Xen hypervisor. This had the drawback of limiting the choice of OS, since it would have to be open source, but had the advantage of improving its performance. Since version 3.0 of Xen, a new mode was introduced called full virtualization (see 2.7.1 on page 18). This mode was only possible with the addition of hardware virtualization extensions and so, the physical hosts must have an Intel-VT or AMD-V processors. Unmodified guest OS as Microsoft Windows can now run in full virtualization mode on Xen, with a minor performance penalty.

On the Xen technology, there are two important and distinct domains: Dom0 and DomU. The Dom0 is a special privileged domain also designated as Domain0. It is the first domain launched when the Xen's system is booted. This domain can be used to create and configure all the guest domains. These guest domains are called DomU because they are unprivileged domain.

The Dom0 has direct access to all hardware on the host machine and provides a simplified generic class device to each DomU. For instance, the network card is viewed as a generic network class device by the guest domain. Dom0 is the only one with a device driver which is specific to each physical device and then communicates with the guest domain using an asynchronous shared memory transport. However, Dom0 has the possibility to delegate the responsibility for a particular device to another domain. This can bring some stability and security advantages, since hardware drivers are the code in an operating system with most likely to fail or have bugs. Using a driver domain, which is a DomU that runs a minimal kernel and a backend for a particular device, it is possible to move the risk of the device management out of the Dom0. Furthermore, if it is necessary to restart the driver domain (e.g. from an error), this is not affect others domain, while if it was necessary to restart the Dom0 this would affect all the system.

A driver domain or a physical device driver running in Dom0 is called a backend, while the generic device accessed by each guest domain is called frontend driver. This technique will allow creating the illusion that each frontend have a generic device dedicated to that domain. The backend acts like a proxy, who understands the details of physical device and enclose the generic device requests from each frontend and encapsulate them according with the specification and send them to the corresponding hardware.

For security and stability reasons, Dom0 should be as small and simple as possible. It is recommended to use Dom0 only for administration of virtual machines on the system and restrict the user level code running on this domain. All other applications should run in a guest domain. Similarly, it is recommend to harden the Dom0, closing services and unnecessary network ports, making it less vulnerable to attacks or faulty software.

The Xen hypervisor and Dom0 acquires their privileged position in the boot process of the system. The only software that will run before them is the boot loader. Currently the most known boot loader for Linux and Solaris is GNU GRUB [52]. Xen hypervisor will be the first software to "get in touch" with the hardware and so from the moment will control the access to it from any other software. When selected on GRUB, Xen Hypervisor boots first and then starts Domain0. If there is guest domains already created, Dom0 can start them automatically, by consulting the configuration files normally located on `etc/xen`.

```
# Modified by YaST2. Last modification on Mon Aug 24 13:47:26
   WEST 2009
default 0
timeout 8
gfxmenu (hd0,1)/boot/message
```

```
##YaST - activate
###Don't change this comment - YaST2 identifier: Original name:
   linux###
title openSUSE 11.1 - 2.6.27.29-0.1 (default)
root (hd0,1)
kernel /boot/vmlinuz-2.6.27.29-0.1-default root=/dev/disk/by-id/
   ata-ST3160815AS_5RX0W316-part2 resume=/dev/disk/by-id/ata-
   ST3160815AS_5RX0W316-part1 splash=silent showopts vga=0x31a
initrd /boot/initrd-2.6.27.29-0.1-default
###Don't change this comment - YaST2 identifier: Original name:
   xen###
title Xen -- openSUSE 11.1 - 2.6.27.29-0.1
root (hd0,1)
kernel /boot/xen.gz
module /boot/vmlinuz-2.6.27.29-0.1-xen root=/dev/disk/by-id/ata-
   ST3160815AS_5RX0W316-part2 resume=/dev/disk/by-id/ata-
   ST3160815AS_5RX0W316-part1 splash=silent showopts vga=0x31a
module /boot/initrd-2.6.27.29-0.1-xen
```

Listing 3.1: *Excerpt from the GRUB configuration file on a OpenSUSE 11.1*

The Linux version used for the Xen Hypervisor was OpenSUSE 11.1 and on this distribution, the GRUB configuration file is located on /boot/grub/menu.lst (as Debian), while on other systems like Fedora and Gentoo Linux, it is located under /boot/grub/grub.conf or /etc/grub.conf. We can see that on Listing 3.1, in the case of Xen a different kernel is loaded (/boot/xen.gz) and while on the first section for the "openSUSE 11.1 - 2.6.27.29-0.1 (default)", the compressed bootable Linux kernel (vmlinuz) is executed on the kernel line. In the case of the Xen section, the vmlinuz is loaded as a module pointing for the same root and resume. The temporary file system used in the boot process of the Linux kernel (designated as initial ramdisk, or initrd) is also loaded as a module.

In the past, choosing an operating system to run in a guest domain was limited to some distributions, because it could use only the ones that were possible to port. Hardware support for virtualization reduced the limitations on the selection of that operating system, making possible to run closed source OS such as Microsoft Windows. In the case of the operating system for the Domain0, there are no restrictions, as long as the operating system supports Xen Hypervisor. In my case, I have selected OpenSUSE, but during my research, I have tried Debian, Ubuntu and CentOS. From all of them, I have choose OpenSUSE only because I was familiar with this distribution and because it was the only one providing the

most recent version of Xen (v. 3.3.1) at the time.

Differences between Xen and VMware

As we have seen before, VMware's architecture is based on direct execution and binary translation. It can run user-level virtual machine code natively on the hardware and is able to translate any privileged code. It is possible to run almost any operating system that would run natively on a x86 inside a VM without modification. Xen's architecture is different because it uses a paravirtualization technique that modifies the guest OS so that it knows that it is running inside a VM. Hardware assisted CPU virtualization techniques (Intel VT and AMD-V) allowed Xen - since version 3.x - to support unmodified or fully virtualized guest operating systems. Another important difference between them is the way they handle device I/O and the way VM I/O route each physical I/O device. This is very important because it has a direct impact on performance, portability and stability. Xen uses a split driver approach where the actual drivers reside in a service VM and special drivers that are inside other VMs will communicate back to that service VM. This has performance advantages but will have impact on closed-source and legacy operating system. VMware ESX has a different approach where the virtual devices drivers in VM communicate with the physical device drivers using ESX kernel. This approach also provides high performance but it has the advantage to support more operating systems, however it is necessary to port new device drivers into the ESX kernel.

Management is another important component, even more on enterprise virtualization. VMware provides management tools like VMware vCenter Server that allows deploying, monitoring, automating operations and managing of virtualized data centers. Novell and Red Hat have been integrating the management of Xen platforms into their operating systems with tools like YaST (Figure 3.2), Anaconda and management utilities like Virtual Machine Manager.

3.3 KVM

KVM stands for Kernel-based Virtual Machine and is a full virtualization software for Linux on x86 hardware, based on hardware virtualization extensions (Intel VT-X and AMD-V) and an adopted version of QEMU, using the Linux Kernel as the Hypervisor [53]. KVM does not run on CPUs without the hardware virtualization extensions. KVM consists of two modules:

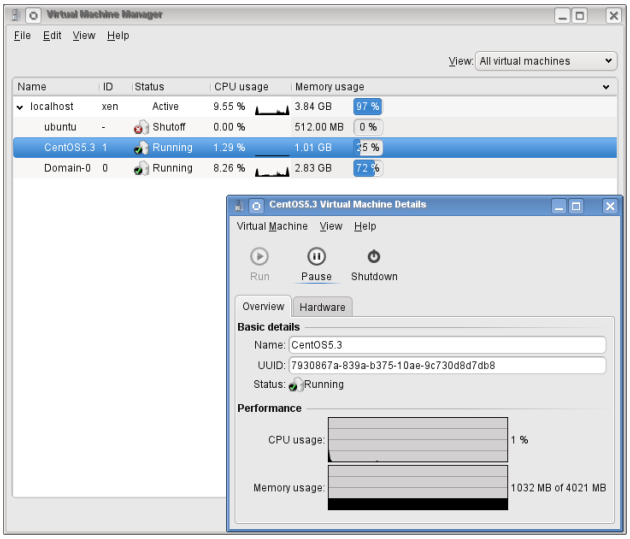


Figure 3.2: *OpenSUSE's virtual machine manager running CentOS on DomU*

- kvm.ko: A loadable kernel module that provides the core virtualization infrastructure
- kvm-[intel|amd].ko: A processor specific module for Intel or ADM

KVM takes advantage of code reuse. By using Linux Kernel 2.6.20 or later, it has the guarantee of being updated to the current hardware and it takes advantage of its scheduling and memory managing system. In addition, it uses also QEMU, to emulate the motherboard hardware (e.g memory controller, network interface, ROM BIOS). As the hardware emulation is based on QEMU, it is possible to move virtual machines between QEMU, KQEMU and KVM hosts.

A traditional UNIX process has two modes of execution: user mode and system mode. However KVM adds a new mode called guest mode. When guest software is running in such a 3-state process, I/O instructions are executed in user mode with the privileges of the user who owns that guest host process. All the non-I/O code runs in guest mode and the system mode is only used for special instructions or for transitions among modes [54].

The kernel module exports a character device (/dev/kvm) which adds the virtualization capabilities to user-space. With this device, a VM has its own address space with its own virtual disk, network adapter and display separated from any other VM that is running. Devices in the device tree (/dev) are normally common to all user-space processes, however this is different for /dev/kvm since each process that opens it sees a different map. This guarantees VM isolation [55].

Each VM is a single process of the host operating system (or hypervisor) and so all the standard Linux process management tools can be used. For instance, it is possible to pause,

resume or even kill a VM with the kill command or view the resource usage by it with the command top. Permissions are handled the same way as a file in Linux, meaning that the VM belongs to the user who started it and that has access to /dev/kvm, which does not necessary mean it needs root access. All the accesses are verified by the kernel, in the same way as any access to files by the user. This has the advantage that any Linux has the necessary standard tools to be used by any system administrator. KVM currently supports Linux VMs (x86 and x86_64) , Windows VMs (x86 and x86_64), BSD VMs (x86 and x86_64), among others. A complete description of the supported systems can be found on the "Guest Support Status" of the KVM site [56].

QEMU is platform virtualization solution that emulates an entire computer environment. Any I/O request from a guest OS are intercepted and emulated by the QEMU process. KVM provides memory virtualization through the /dev/kvm device. The physical memory that is mapped for the guest OS is in fact virtual memory mapped into the process. This is achieved by using shadow page tables [55].

3.4 QEMU

The terms virtualization and emulation are occasionally used to describe virtualization products. Sometimes people say, by the fact that they are running something that was supposed to run in another platform, then that is considered virtualization. For instance, it is possible to run ancient arcade games in a normal computer, using images of their ROMs. In this case, what we are running is an emulator program and not a virtualization program. Virtualization techniques normally run software that was compiled for the native instruction set of the physical hardware on where the virtual machine is running, while emulation techniques normally emulates all the physical environment, including processor. Microsoft Virtual PC and QEMU are two examples of emulation product.

QEMU stands for Quick Emulator and it is an open source emulator, which uses dynamic recompilation with the purpose to reduce the overburden caused by the emulation. It was developed by Fabrice Bellard and it can emulate a multiplicity of architectures, processors and related peripheral hardware [57]. It can run OS and software that was compiled for platforms such as 32 - and 64 - bit x86, 32 - and 64 - bit PowerPC, Motorola 68000, 32 - and 64 - bit SPARC, SH, MIPS, and ARM. It can virtualize a complete hardware environment for each of these processors and architectures, enabling the possibility to run unmodified guest OS for any of those. It is used as the base of various virtualization products, like KVM, Virtualbox ,KQEMU and even Xen.

According to Bartholomew [58], QEMU provides emulated versions of the following x86 hardware:

- i440FX host PCI bridge and PIIX3 PCI to ISA bridge.
- Cirrus CLGD 5446 PCI VGA card or dummy VGA card with Bochs VESA extensions (hardware level, including all nonstandard modes).
- PS/2 mouse and keyboard.
- Two PCI IDE interfaces with hard disk and CD-ROM support.
- Floppy disk.
- NE2000 PCI network adapters.
- Serial ports.
- SoundBlaster 16 card.
- PC BIOS from the Bochs Project.
- Plex86/Bochs LGPL VGA BIOS.

QEMU has two emulation modes: User mode emulation and complete computer system mode emulation. On user mode emulation, QEMU can run Linux or Mac OS X processes compiled for one architecture on another. An example of this mode is when a user runs a Windows API on Linux using Wine. On the complete computer system mode emulation, or just system emulation, is where QEMU can emulate a complete computer system, allowing running unmodified OS and user processes for a specific architecture and processor.

3.4.1 KQEMU

KQEMU was also written by Fabrice Bellard and is a Linux Kernel module to accelerate QEMU on x86 platforms. . It is a type 2 hypervisor, using dynamic recompilation. User mode code is run directly on the host's CPU while the kernel mode code uses processor and peripheral emulation. There is also the possibility of kernel emulation where a portion of the kernel mode code runs on the host's CPU. One of the advantages of KQEMU compared with KVM (which is another Linux kernel module) is that KQEMU does not need that the host CPU to support hardware virtualization.

It can run on X86 and the only operating system full supported is Linux. However, there are some experimental versions for FreeBSD and Windows XP

3.5 Microsoft Virtual PC

Around the same time VMware released their VMware ESX Server, Connectix was having a good partnership with Microsoft, because they provided a bundle of Microsoft's operating systems with the Connectix Virtual PC for Mac. In the beginning of 2003, Connectix gives the first public stop for the x86 server virtualization, releasing their first release candidate of their Connectix Virtual Server. The final release of this product never saw the daylight as Connectix because Microsoft acquired the intellectual property rights of both Connectix Virtual PC for Windows as Mac and as well as Connectix Virtual Server.

The released of Microsoft's first virtualization product was back in December 2, 2003 with Microsoft Virtual PC 2004. On the next year, Microsoft planned to release their Microsoft Virtual Server 2004, entering into the x86 server virtualization market, but the released was delayed due to the new Microsoft security initiative. Because of that, the final product was released in the middle of 2004 in two versions - Microsoft Virtual Server 2005 Standard Edition (limited to four physical processors) and Microsoft Virtual Server 2005 Enterprise Edition (supporting up to 32 physical processors).

Microsoft Virtual PC emulates x86 platform and was very popular when Apple Macintosh systems used PowerPC chips, because it was possible to install Microsoft Windows on their PPC systems. Currently, Apple Macintosh use Intel chips and Parallels Workstation and VMware Fusion conquered Apple users for their virtualization products.

Currently, Microsoft Virtual PC is already considered a virtualization product for Microsoft Windows operating systems but continues to be an emulation product for Mac OS X and has been deprecated for this one. It is free to download and use, but officially supports only Microsoft Windows operating systems.

3.6 Microsoft Hyper-V

The fact that VMware ESX Server and Xen are having more and more a deeper penetration on the server virtualization market made Microsoft to develop their own hypervisor technology, originally called Viridian and later renamed to Hyper-V.

Hyper-V is a type 1 hypervisor because it runs directly on the hardware of the host system and is responsible for sharing the physical hardware resources with multiple virtual machines. It is part of the Windows Server 2008 and it is a native 64-bit hypervisor that

can run 32-bit and 64-bit VMs concurrently. The main goal of Hyper-V on Windows Server 2008 is to enable the possibility to run multiple guest operating systems (also called partitions) on a single server hardware system.

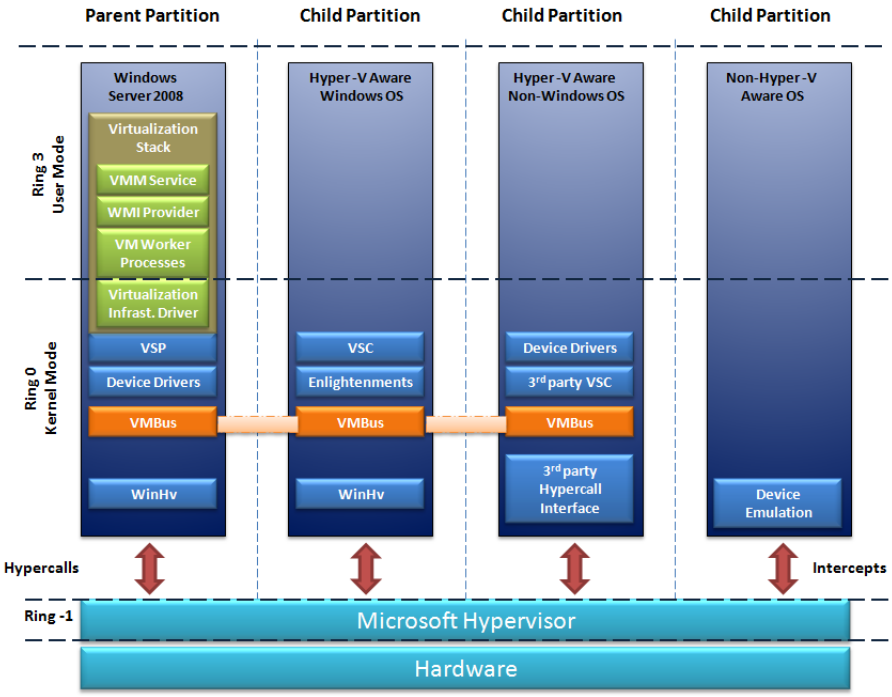


Figure 3.3: *Overview of hypervisor architecture*

As we can see on Figure 3.3, Microsoft hypervisor runs on top of bare metal, which makes this architecture a type 1 hypervisor. On top of the hypervisor, there is one parent partition and one or more child partitions. The concept of partitions is the same as the domains in Xen. It is a unit of isolation within the hypervisor with allocated physical memory address space and virtual processors. There are two types of partitions:

- Parent Partition: Is the controlling partition where the virtualization stack runs and is the one responsible for the management of other child partitions. Is also the one that manages and assign the hardware devices with the exception of processor scheduling and physical memory allocation since these are handled by the hypervisor.
- Child Partition: This is where the guest operating systems (and their applications) will run and they are created by the parent partition.

Partitions communicate with the hypervisor layer by using “hypercalls”. These hypercalls can be considered application programming interfaces (APIs) used by partitioned operating systems in order to take advantage of the optimization provided by the hypervisor.

We are going to see a brief description of each component of the virtualization stack on the parent partition:

- Virtual Machine Management Service: The VMM Service is responsible for managing the state of all virtual machines in child partitions. It is also responsible for controlling what operation can be performed on a virtual machine in a given state.
- WMI Provider: provides an interface for remote administration
- Virtual Machine worker processes: User mode process, which provides VMM Service from the Windows Server 2008 instance in the parent partition to the guest operating systems in the child partitions. For security reasons, each VMMS spawns a separate VM worker process for each running VM. This allows isolations in the sense that if one VM worker process fails, only the VM associated with it is affected.
- Virtualization Infrastructure Driver: Is the kernel-mode component of the virtualization stack and provides three principal management services (MS) for all child partition: Virtual processor MS, Memory MS and partition MS.

Beside Virtualization Stack, some other components are important for the Parent partition:

- Windows Hypervisor Interface Library (WinHv): is the kernel-mode dynamic-link library that loads inside the Windows Server 2008 instance running in the parent partition and inside the guest OS in any child partition although the guest has to be Hyper-V-aware.
- Virtual Service Providers: VSPs provides a way of publishing device services to child partitions. In order to do it, it provides I/O related resources to Virtualization Service Clients (VSC) running in the child partitions.
- Virtual Machine Bus (VMBus): This is a logical inter-partition communication mechanism between the parent partition and the child partitions. It is channel-based and its purpose is to provide high-speed communications mechanism between virtualized partitions. For instance, VSCs and VSPs use client/server communication for device functionality using VMBus.

Hyper-V-aware Operating Systems include Windows Server 2008, Windows Server 2003 SP2, Novell SUSE Linux Enterprise Server 10, Windows Vista SP1, and Windows XP SP3. Non-Hyper-V-aware Operating Systems include Windows Server 2000 and older versions of Windows.

As we have seen on Figure 3.3, there are three types of child partitions:

- Child partitions hosting Hyper-V-aware Windows operating systems
- Child partitions hosting Hyper-V-aware non-Windows operating systems
- Child partitions hosting non-Hyper-V-aware operating systems, either Windows or other types

Beside WinHv and WMBus, child partitions hosting Hyper-V-aware Windows operating systems include two important kernel-mode virtualization components, which are virtualization service clients (VSC) and Enlightenments. VSC are devices residing in the child partition that use hardware resources provided by the VSP, which are in the parent partition. The communication between them is made over VMBus. VSCs are available automatically if the operating system installed on the child partition has integration services (IS) installed. Professional versions of Windows 7 (Business, Enterprise and Ultimate) as well as all versions of Windows Server 2008 R2 already come with integration components installed, but it is necessary to install as an extra for other operating system, like Windows Vista. Without IS installed, a child partition can only use emulated devices which is the case for the Child partitions hosting non-Hyper-V-aware operating systems. Enlightenments refer to the modification made to operating systems so these can run more efficiently as a guest within a hypervisor environment. Child Partitions Hosting a Hyper-V-Aware Non-Windows Operating System use third party VSCs to communicate over the VMBus, however it is necessary to have the Integration Services installed in the child partition.

Integration Services are key components that provide Heartbeat, Time Synchronization and Volume Shadow Copy Service among others. Hyper-V includes Integration Services for both x86 and x64 versions of the following Windows operating systems:

- Windows XP with Service Pack 3 (SP3)
- Windows Vista with Service Pack 1 (SP1)
- Windows 7
- Windows Server 2003 SP2
- Windows Server 2008
- Novell SUSE Linux Enterprise Server 10

- Novell SUSE Linux Enterprise Server 11

The version of Integration Services developed by Microsoft for the Novel SUSE Linux Enterprise Server is called Linux Integration Components for Hyper-V.

The third type of Child Partitions is related with those that cannot have Integrated Services installed on them. This can be because it is an older version of Windows (Windows Server 2000 or previous) or a third-party operating system not supported (like Red Hat Enterprise Linux 5.2). In this case, the operating system must use emulated devices, which has performance impact.

3.7 VirtualBox

VirtualBox is type 2 virtualization software package for x86 infrastructures originally developed by a German company called Innotek. On February 20, 2008 Sun completed the acquisition of Innotek [59] and since then, VirtualBox makes part of its Sun xVM virtualization platform.

VirtualBox uses software virtualization to run VMs, however, when running on Intel-VT and AMD-V capable CPUs, it provides the option to enable hardware virtualization on a per virtual machine basis. Prior to version 2.2, software was the option by default. On all the versions prior to that until the current version (3.0.10), the default option for hardware virtualization is enabled for the new virtual machines. The reason pointed by VirtualBox for this change is the fact that running a VM using the latest Intel and AMD processors with this option enabled is faster than using software virtualization in most of the situations.

VirtualBox, in its current version (3.0.10) supports the following guest operating systems:

- All family of Windows versions (from Windows 3.1 to Windows 7 – x86 and 64-bit).
- Linux (kernel 2.2, 2.4 and 2.6 – x86 and 64-bit)
- OS/2 Warp (OS/2 Warp 3, OS/2 Warp 4, OS/2 Warp 4.5, eComStation)
- Solaris and OpenSolaris – x86 and 64-bit
- BSD (FreeBSD, OpenBSD and NetBSD – x86 and 64-bit)
- DOS, Netware, L4 and QNX

For the host operating system, VirtualBox can be installed on Windows x86/AMD64, Mac OS X, Linux, Solaris and OpenSolaris x86/AMD64

3.8 Virtualization Solutions Comparison Matrix

The virtualization solutions described are, in our opinion, the most important products currently used. On Table 3.1, we present a comparison of various hardware and operating system-level virtualization solutions products that includes the previous one presented and the ones we did not described but we consider important. The column “Host OS” means different things depending of the type of the hypervisor:

- For the case of type 1 hypervisors, this column is related to the operating system supported for the special privileged area such instance by the dom0 (Xen) or parent partition (Hyper-V).
- For the case of type 2 hypervisors, this column is related to the operating system on where the hypervisor is installed.

3.9 Conclusion

This chapter described some different type 1 and type 2 server virtualization products that we consider to be the most used on x86 architecture. There are many solutions that a person could chose, some open source, others proprietary. VMware is the most used virtualization product, with an installed base of 89 percent. Therefore, we can say that VMware is for server virtualization what Microsoft is for desktop operating systems or what Cisco is for network.

Name	Virtualization	License	Architectures	Guest CPU	Host OS	Guest OS	Type
Xen	[F][P]	GPL	x86, x86-64, IA64, PPC	x86	Linux, Solaris, NetBSD	BSD, Linux, Solaris, Windows	1
KVM	[F][P]	GPL	x86, x86-64, IA64, PPC, S390	x86	Linux	Linux, Windows	1-2
UML	[P]	GPL	x86, x86-64, PPC	x86	Linux	Linux	2
L4Linux	[P]	GPL	x86, ARM, MIPS	x86	L4	Linux	1
QEMU	[F]	GPL, LGPL	x86, x86-64, IA64, PPC, ARM, MIPS, SPARC (kQEMU only x86/x86-64)	x86, SPARC, PowerPC, MIPS	BeOS, BSD, Linux, Mac OX, Solaris	BSD, Linux, Windows	2
OpenVZ	[O]	GPL	x86, x86-64, IA64, PPC, SPARC, ARM	Same as Host	Linux	Linux	2
VirtualBox	[F]	GPL/ Retail	x86, x86-64	x86	Windows x86/x86-64, OS X, Linux, Solaris and OpenSolaris	BSD, Linux, Solaris, Windows	2
VMware Server / Workstation	[F]	Server: Free Retail Workstation: Retail	x86, x86-64	x86	Windows, Linux	Windows, Linux, Novell Netware, Sun Solaris, MS-DOS, FreeBSD	2
VMware ESX(i)	[F]	Retail	x86, x86-64	x86	None - bare-metal	Windows, Linux, Novell Netware, Sun Solaris, FreeBSD	1
Microsoft Virtual PC	[F]	Retail	x86	x86	Microsoft Windows	IBM OS/2, Microsoft Windows	2
Microsoft Hyper-V	[F]	Retail	x86	x86	Windows 2008 w/Hyper-V Role, Windows Hyper-V Server	Windows, SUSE	1
z/VM	[F][P]	Retail	s390	s390	z/VM runs directly inside LPAR	Linux for System z, z/OS, z/TPF, z/VSE, z/VM	1
Parallels Workstation	[F]	Retail	x86	x86	Windows, Linux	Windows, Linux, FreeBSD, OS/2, eComStation, MS-DOS, Solaris	2

Legend: [F] - Full Virtualization, [P] - Paravirtualization, [O] - OS Virtualization, SMP - Symmetric MultiProcessor

Table 3.1: *Comparison of the features and performance of the various virtualization technologies available*

Chapter 4

Security of Virtual Machines

4.1 Some Important Concepts

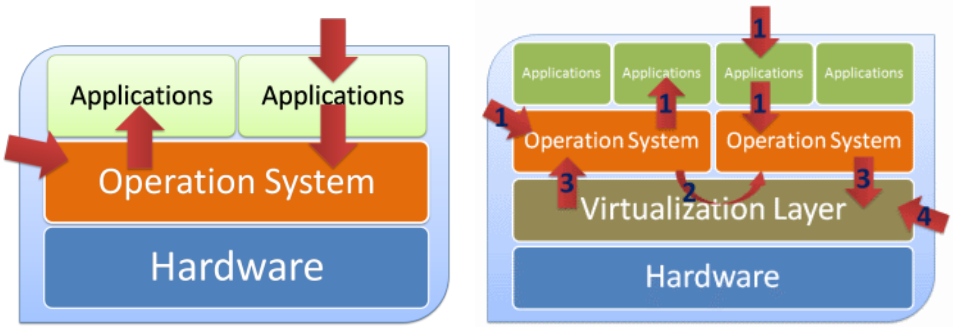
As we have seen in the previous chapter, Virtual Machine technology is going mainstream, implemented on every branch of the industry (e.g. Telecom, Finance), running critical services, which were previously implemented in isolated servers.

There are many reasons for its implementation: cost savings, server consolidation, disaster recovery and improved business continuity, among others. For instance, in order to have a database server, an email server and a webserver, running on different operating systems, it was necessary to have three servers, each running on a different operating system and service. Nowadays, with virtualization, it is possible to reduce the number of servers, being necessary, for instance, only one server running a hypervisor and three virtual machines.

However, industry pundits agree that a very important component has been neglegted: security. According to Gartner [60], "through 2009, 60 percent of production VMs will be less secure than their physical counterparts" and that "30 percent of deployments [will be associated] with a VM-related security incident". This can even be worst if we consider that 85 percent of VMware's customers are using virtualization for mission-critical production services.

Let us take as example the previous simple case. In the scenario with the three isolated servers, we had a traditional threat model (Figure 4.1a on page 46) where to compromise the host, the attacker would typically explore a vulnerability in an application and with that attack the operating system or she could explore some vulnerability on the operating system and then get access to the application. With virtualization, a new scenario is presented, with all the servers consolidated on a single server running three virtual machines. In

this case, a new threat model (Figure 4.1b on page 46) allows not only the same attacks as the traditional model but also opens the possibility to do new attacks and explore new vulnerabilities. Since each VM can be “exactly the same” as the real one (in the sense that it can have, for instance, the same number of CPUs, amount of memory, patches installed and configuration), an attacker can explore a vulnerability in an application the same way as it was done on the traditional threat model (arrows #1). Besides that, she has new targets to check for vulnerabilities. If the virtualization layer have vulnerabilities, she could launch an attack from a guest OS against the others VMs in the same host (arrows #2), or could attack the host by doing a denial of service exploring the host vulnerability in a virtual device . Doing this attack, she could attack the other guests in the same host (arrows #3). The impact of compromise the virtualization layer raises the risk’s level, since it is a critical asset shared by all the virtual machines. However, the attack can be remote too (arrow #4). For instance, recently it was discovered a vulnerability in some VMware products which allows remote attackers to execute arbitrary code on vulnerable hypervisors.



(a) *Traditional threat model* **(b)** *New threat model*

Figure 4.1: *The traditional vs new threat model*

Another important aspect related to security failures in virtualization regards misconfiguration resulting from human error. According to Gartner [60], "the security issues related to vulnerability and configuration management get worse, not better, when virtualized”. One of the reasons for this concern is the fact that replicating an insecure VM image is easier than before. VMs provide mobility similar to a normal file. For instance, they can be copied to other computers over a network or it can be carried on a portable storage media. Therefore, they can be deployed in many systems, some not managed by the local administrator, and so, it requires a bigger effort to discover those insecure images and correct them.

Many organizations still have the same approach in order to secure their VMs as if they were securing an operating system, using the same configuration guidelines and standards.

This is a ”normal” mistake, even more because there is a lack regarging virtualization on pilar documents as the information security standard ISO/IEC 17799:2005 and ISO/IEC 27002:2005.

4.1.1 Isolation

As we have seen in Chapter 2.3, one of the key issues in virtualization is isolation. This isolation will guarantee that one application in one VM cannot see applications running in a different VM, or that some process running in one VM can not affect the other VMs running in the same machine. If this security assumption is broken, then an attacker can have access to other VMs in the same machine or even to the host machine. That is why it must be carefully configured and maintained.

A workaround to isolation, possible to use in some software like VMware Workstation or Sun Virtualbox, is to share the clipboard. This way, it is possible to exchange data between the host and the guest machine. However, this workaround can open breaks to the security. According to Kirch [61], sharing the clipboard can “provide a gateway for transferring data between cooperating malicious programs in VMs of different security realms or to exfiltrate data to/from the host operating system.”

In the same whitepaper, the author refers some others isolation security breaks like the fact that in one VM technology (without mention which one), “the operating system kernel that provides the VM layer has the ability to log keystrokes and screen updates passed across virtual terminals in the virtual machine”. These logs (keystrokes and screen updates) are saved into files on the host machine, which allows external treatment (analyzing, monitoring, eavesdropping) of terminal connections inside the VM, even the encrypted one.

In order to guarantee isolation, a program running inside a VM A should only interact with another program in VM B on the same host the same way that a program installed on a physical machine A would interact with another program on other physical machine B. If a program on VM A is able to change memory or monitoring VM B, then this is considered a serious isolation break.

4.1.2 Controlling VMs from the Host

Hosts have the authority to control VMs and therefore can interact with them in many ways, depending of the VM technology used [61]:

- It can start, pause, stop (shutdown) and restart VMs
- It can change and monitor different VMs resources as CPU, memory, disk, among others.
- It can interact with the VM´s virtual disks, with operations like view, copy and potentially modify data stored.

Beside the previous properties, normally the host can also monitor the network traffic from or to VMs. Because of all these control that hosts machines have over the VMs, it is an important target of attack and so, needs to have a special attention.

4.2 Analyze of Security Vulnerabilities in Virtualization

4.2.1 Attacks from the Guest to the Host

There can be some external or internal factors that can compromise isolation as miss-implementation/configuration or some bug in the virtualization software. A dangerous attack can be made if isolation between the host and the VMs is compromised. That attack is called ”VM escape” and happens when a program can bypass the virtual machine layer from inside a VM and get access to the host machine. The host machine is the root of all the VMs, and so if a program escapes from the virtual machines privileges it will get root, allowing to control all the VMs and the traffic between them. This kind of attack are normally possible by exploiting bugs on the VMM combined with improperly configuration of the host/guest interaction. However, current VMMs do not offer perfect isolation, although they claim to. Many bugs have been found in all popular VMMs, some of them allowing ”VM escape”, like the VMware Workstation 6 CVE-2007-4496 [62] bug discovered by Rafal Wojtczuk which allows authenticated users with administrative privileges on a guest operating system to corrupt memory and possibly execute arbitrary code on the host operating system via unspecified vectors.

What happens if isolation failed and a VM can monitor another VM? As we have seen before, isolation plays a crucial role in virtualization. Therefore, it is considered a security flaw if isolation is overcome, and a VM can monitor another without any difficult and specific configuration. Moreover, this is not a probable thing to happen neither easy to achieve. Most of the modern CPUs have mechanisms of memory protection, which can be enforced by the hypervisor, and they are responsible for memory isolation. This way, if

well implemented (and again, the idea of many of the security flaws happens because they are not well implemented), memory protection should avoid VMs to see memory used by others VMs.

Even if they do not share memory (forced by the memory protection), there is something "shared" by them and this can be another point of attack, depending how it was implemented. There is no network protection implemented by hardware as there is with memory and so, this is something that requires special care. The more secure way would be by using a dedicated physical channel for each host-VM link. This way, each guest VM could not sniff what the others are sending or receiving. However, sometimes we can find VM platform linked to the host using "virtual hub" or "virtual switch". If a "virtual hub" is being used, the VM guests are able to sniff the packets in the network, while in the case of using "virtual switch", VM guests can do a Address Resolution Protocol (ARP) spoofing [63], redirecting the packets to them, and in this way, be able to sniff packets going to and coming from other guest VM.

According to the author [61], in case of necessity of using "virtual hub" or "virtual switch", two mechanisms could be implemented. It could be used authentication of network traffic and also, in order to avoid ARP poisoning, "enforce limits on what Ethernet MAC address is used on a VM's virtual network interface", although there are some doubts among the community regarding if is possible to avoid this attack if VMs are on the same VLAN/Port group [64]. Port groups define how VM connections are made through the virtual switch. Using port groups it is possible to configure bandwidth limitations and VLANs tagging policies for each member port. It is also possible to aggregate multiple ports in order to provide a local point for virtual machines to connect to a network.

Recently, a bug was found in VMware's virtualization software [65] which would allow a guest to write to arbitrary memory. Althought a patch was already released, bugs like this are very serious and can put in risk a VM architecture.

4.2.2 Remote Management Vulnerabilities

It is common in the current VM Environments to have a management consoles that manages the virtual machines. Normally, commercial products have their own. For instance, VMware uses VMware vSphere to manage the Hypervisor, while Citrix XenServer can use XenCenter. These consoles bring new facilities for administrators to manage their machines, but also open new vulnerabilities. Compromising a management console allows an attacker to control all the virtual machines managed by it. These type of technology

normally communicates with the VMM using HTTP/HTTPS which mean the VMM has to have a service running accepting HTTP connection. Xen, for instance, has the XenAPI HTTP Interface that had a Cross-site scripting (XSS) vulnerability, which allowed running a script code in a user's browser session in context of an affected site.

HyperVM [66] is a multi-tiered, multi-server, multi-virtualization software which allows to create and manage different Virtual Machines (Xen or OpenVZ) each with each Virtual Private Server (VPS) having its own operating system. In June 8 of 2009, 100,000 hosted websites were affected by a zero-day SQL injection hole [67] in the HyperVM 2.09 . With this attack, it was possible to gain root privileges allowing the attacks to run sensitive Unix commands as "rm -rf", which forces a recursive delete of all files in the current and sub directories. As a result, many clients were affected and some lost their data forever, because they did not have any backup.

4.2.3 Denial of Service

A Denial of Service (DoS) has the goal to make a computer resource not available to its intended users. In virtual machine architecture, resources as CPU, memory, disk and network are shared between the host and the guests. It is then possible for a guest machine to impose a denial of service (intentional or not) to others guest which would also affect the host by taking all the possible resources of the system. When other guests try to request a resource, the system will deny that access since there is no resource available. VMware has been shown to suffer from several DoS vulnerabilities(4.2). A good approach to prevent this attack from a guest is to limit the resources VMs can access. Most of the current virtualization technologies have the mechanisms necessary to limit the resources allocated to each guest machine. With the correct configuration of the host virtualization, this attack can be minimized.

4.2.4 Virtual-Machine-Based Rootkit (VMBR)

The concept of Rootkits appeared in the Unix world. They were developed to replace standard Unix tools with versions that gave a user root or super-user privileges, while allowing their activity to remain invisible to other users. Rootkits had the particular ability to hide itself, being very difficult for users or processes to discover their existence. This amazing ability soon would raise eyebrows on hackers, because it would be an ideal way to cover their devious activities. A rootkit is normally designated as a program designed to hide not

only itself, but also other programs and all its associated resources (e.g. processes, files, ports).

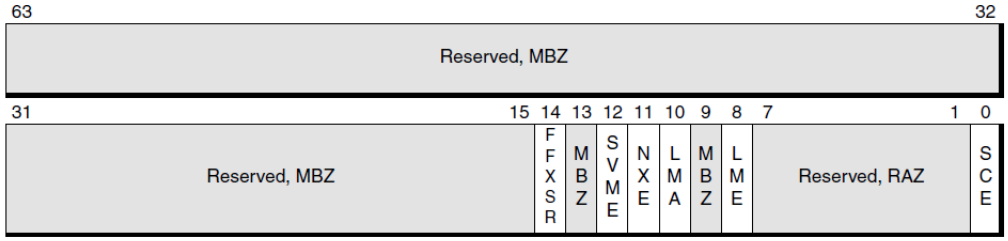
A rootkit tries to escalate privileges, with the goal of running in ring 0 which is the operating system's kernel mode. If it succeeded, can easily terminate applications run in ring 3 (user mode), by any normal user, including root. Virtualization adds a higher privilege ring (ring -1) and if a rootkit could compromise that ring, it gain control of the whole physical environment on which the system runs. Rootkits hiding in this layer are considered VMBR and they are even harder to discover and be removed than then ones on kernel mode. Two example of this type of rootkits are BluePill and SubVirt. A third one was also released called Vitriol, created by Dino Dai Zovi using Intel VT. I am now describing the first two.

4.2.4.1 Blue Pill

Blue Pill is a VMBR rootkit created by Joanna Rutkowska for COSEINC in 2006 to exploit the AMD64 Secure Virtual Machine (SVM) Extensions - also known as Pacifica - on AMD Athlon 64. The main idea was that it should install itself without necessary any intervention of the machine, and would move the operating system into the virtual machine. This rootkit was itself a hypervisor that would allow to control the guest OS and cannot be detected using any integrity scanner. Joanna Rutkowska and Alexander Tereshkin released a new version of BluePill in 2007. It was redesigned because the original version is property of COSEINC. This new version was able to work on the AMD SVM and Intel VT.

Blue Pill uses the concept of "thin hypervisor" to control the operating system. This thin hypervisor is based on hardware virtualization provided by the SVM [23] and VT [68]. SVM makes part of the AMD-Virtualization (also called AMD-V) and it is an extension of the AMD 64 architecture, which provides hardware support to improve performance and facilitate implementation of virtualization introducing two modes designed host mode and guest mode and a new instruction VMRUN.

As in the case of the AMD64-based processor, which boots in legacy x86 mode in order to be compatible with 32-bit operating system, the AMD-V processor boots up in legacy "guest mode", until a compatible VMM is turned on and the VMRUN instruction is issued. When this happens, the processor shifts into host mode, similar to what happens when the 64-bit operating system would active the x64 mode on an AMD64 previously started as 32-bit. The "turn on" operation that will enable the VMM it is a bit called Secure Virtual Machine Enable (SVME). The extended-feature-enable register (EFER) is a model-specific register (MSR) with an address of C000_0080h and it can only be read or write by privileged software.



Bits	Mnemonic	Description	R/W
63–15	Reserved, MBZ	Reserved, Must be Zero	
14	FFXSR	Fast FXSAVE/FXRSTOR	R/W
13	Reserved, MBZ	Reserved, Must be Zero	
12	SVME	Secure Virtual Machine Enable	R/W
11	NXE	No-Execute Enable	R/W
10	LMA	Long Mode Active	R
9	Reserved, MBZ	Reserved, Must be Zero	
8	LME	Long Mode Enable	R/W
7-1	Reserved, RAZ	Reserved, Read as Zero	
0	SCE	System Call Extensions	R/W

Figure 4.2: *Extended feature enable register (EFER)*

As we can see on the Figure 4.2, on the bit 12 we can find the SVME bit. This bit has to be set to 1 before any SVM instruction is executed.

The VMRUN instruction is the keystone of SVM. The VMM calls the VMRUN instruction that has an implicit addressing mode of [rAX]. The VMRUN will then take the Virtual Machine Control Block (VMBC). This VMBC, in a simplified way, describes a guest to be executed and has the settings that determine what actions cause the guest to exit to host. The system switches into guest mode, booting up its own operating system that will run in its own ring 0 privileged-mode instructions, and its own applications at the ring 3. This guest will run until it takes an action that causes an exit to the host or if it clearly (legal or illegal) calls the VMM (using the VMMCALL instruction). When this happens, the information about the intercepted event is written on the VMCB and the host resumes at the instruction following VMRUN, as we can see on the Figure 4.3.

Anything made by the guest operating system that causes an interrupt or execute a ring 0 command will make the VMM taking control of it in the Host mode. Therefore we can say that using host mode is more secure and efficient than using virtualization carried out exclusively in software [69].

Intel VT works in a similar way to AMD-V and so it is not going to be detail the process, only during the explanation of Blue Pill.

Currently, all of the most important open source virtualization projects (e.g. Xen, KVM, VirtualBox) and comercial solutions (e.g. VMware, Parallels, Oracle VM) supports both AMD-V and Intel VT.

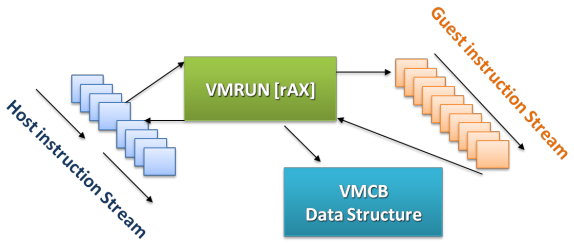


Figure 4.3: *Workflow of the host and guest mode*

The idea regarding Blue Pill is that the attacked machine has the process running in Ring 0 privileged-mode like the kernel-mode drivers. Then it will check if it has support for hardware VM, and if it does, enable it in the extended features enable register (AMD-V: EFER.SVME, Intel VT: CR4.VMXE). Then it is allocated a memory region for storing host and guest data on transitions and the VMBC (or VMCS in case of Intel VT) is prepared in order to treat the #VMEXIT handler. After this, the hypervisor will execute the guest-entering instruction (AMD-V:VMRUN, Intel VT: VMLAUNCH), and this way will control the native operating system, which will have no clue that is running inside a hypervisor.

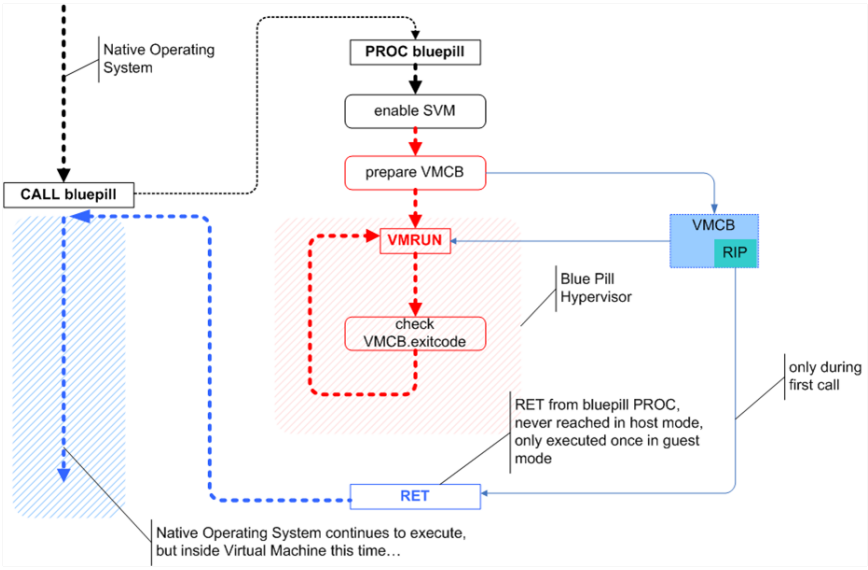


Figure 4.4: *The Blue Pill idea (simplified) [1]*

4.2.4.2 SubVirt

SubVirt [70] is a VMBR rootkit created by researchers at Microsoft Research and the University of Michigan. The name is a combining portions of the words subvert and virtual. Subvert is one of the abilities that rootkits use to trick the operating system into believing

the rootkit does not exist.

SubVirt makes use of VirtualPC or VMware in its own area of disk space that is totally undetectable and off-limits to the host operating system. SubVirt operates at a level below the host kernel and remains inaccessible to the host operating system. The original host operating system is placed inside a virtual machine. The boot sequence is modified by the kernel module to load original operating system inside the Virtual PC (or VMware in case of a Linux).

This rootkit survives a restart, but needs an inicial restart in order to be installed.

Differences Between SubVirt and Blue Pill

As previously said, SubVirt is a restart-surviving rootkit. In fact it needs an initial restart in order of the SubVirt's installation process takes control before the original operating system boots. In contrast, Blue Pill does not require any restart and can be installed 'on-the-fly'.

SubVirt was implemented on x86 hardware, which does not allow to achieve 100 percent virtualization. There are number of sensitive instructions, which are not privileged on an x86 hardware, like SIDT/SGDT/SLDT. Blue Pill relies on AMD64 SVM technology.

SubVirt is based on one of the commercial VMM: Virtual PC and/or VMWare. Both of these applications create virtual devices, which can be easily detected by the guest machine. Blue Pill is a thin hypervisor, and the hardware is accessible without loss of performance (e.g. the 3D graphics card stays with the same performance as if there was no hypervisor).

4.2.4.3 Detecting VMBR and Ways to Protect Against These Attacks

When BluePill was presented for the first time, Joanna declared it was 100 percent undetectable. That was proven to be wrong. Later on the same year, many investigators claimed that, by using external timing, they could detect Blue Pill. They argue that executing some instruction in a none virtual machine, it takes some certain time, but doing the same operation inside a virtual environment, it will take much longer. Based on this, they can tell if it is in a virtual machine or not. Although this is true, without any baseline comparison, as the time required for the same machine to run the same number of interactions of the same instructions before and after the system has been virtualized, then it will not be possible to compare correctly, and by that, detect that it is running inside a hypervisor is almost impossible.

In 2007, Keith Adams et al. [71] suggest that by using translation lookaside buffers (TLBs) it was possible to detect Blue Pill. In the paper, the authors claim that, because VMM and guest virtual address mappings competes for the same pool of TLB, the guest OS could detect that it is running inside a VMM since the size would be smaller.

However, not all VMBR are hard to detect. For instance, in order to survive reboots, SubVirt change the master boot record of the hard disk, and so it can be detected, even for "off line" detection. Nevertheless, it is a fact that detecting a VMBR is a hard job to do, although not impossible. In the same paper, the authors suggest that people should not focus on the detection but in the prevention, because it is easier.

The simpler way would be the suggestion of Microsoft by disabling hardware virtualization extensions by default for client-side systems. This solution in fact would prevent VMBR to install, but it would also disable legitimate usages of virtualization. A small workaround to this is provided by AMD-V, which allows the SVM to be re-enabled using a 64-bit key introduced by the user for the valid hypervisor. If a malicious hypervisor would attempt to install, it would fail because it does not have the necessary key.

Another possibility would be using Trusted Platform Module (TPM) provided by Intel's Trusted Execution Technology (formerly LaGrande) or AMD's SVM. This would allow the CPU to reboot into a trusted state using a TPM verified secure loader.

Two more things could be done in order to avoid VMBR. A simple approach could be by doing a safe boot from CD. The second idea could be by installing a trusted hypervisor first. This idea has the drawback that it requires IOMMU/VT-d support, otherwise the VMBR could write on the hypervisor's page in memory and again, take control of the system [72].

Some security expertises have seen VMBR as impractical to implement in production systems. The main reason is related with system resources. Virtual machines place quite a drain on those, particularly memory and disk space. For instance, it would be hard an administrator miss the disk space occupied by a VMBR like SubVirt, which is installed with its own operating system. The technology advance can make these detections more difficult, since servers have more and more memory and disk space, although also VM tend to be bigger and occupy more. Nevertheless, if they turn out to be a reality outside the labs, then this type of malware can become extremely dangerous.

4.2.5 The Intrusion Detection/Prevention Approach

VMware, as one of the leaders on virtualization, has an important role to play on VM security. Therefore, they implement advanced techniques in order to provide transparent

traffic analysis and threat interception. Recently they released VMsafe APIs, which can alert about an on-going attack, in the same way as an IDS, or can terminate open malicious sessions, like an IPS. However, VM security is not only limited to this. In an article about Taming Virtualization, Carbone et al. [73] proposed GuardHype. With the focus on VMBR prevention, GuardHype acts like a hypervisor for hypervisor, because it controls the access of hypervisors to the hardware's virtualization extensions. Using the same technique as Blue Pill, it emulates the CPU's virtualization extensions, allowing third-party hypervisors to run unmodified on top of it. During my investigation, I have exchanged some emails with Carbone. He told that currently there is no implementation of GuardHype, but that Phoenix (the BIOS manufacturer) has a project called HyperSpace which implements a very similar concept to the one proposed by them. HyperSpace has a "small hypervisor which supports nested virtualization and an additional security domain where security applications can be deployed", he told me. Therefore, in his opinion, in the near feature it would be thinkable to imagine GuardHype as being a component of our BIOSes. There has been some study in this field to enable hypervisors to detect any malicious modification inside a VM. Both Manitou [74] and Patagonix [39] use a hypervisor to detect and identify stealthily executing binaries on a computer system.

4.2.6 The Revert to Snapshots Problem

A disk "snapshot" is a mechanism used by some well known VMMs (e.g. VMware and VirtualBox) which allows the administrator to take a snapshot of the Guest machine at a certain point in time. What it does is to preserve the disk file system and the system memory, allowing the administrator to revert to the snapshot in case of necessity.

They can be lifesavers but they can bring some security problems too. For instance, revert to a snapshot can:

- Insert an un-patched vulnerable machine online again
- Re-enable previous disabled accounts or passwords
- Use old security policies (e.g. firewall rules, antivirus signatures)

However, it can have other security problems. Galfinkel et al. [75] alerts for the problem of using snapshots on systems with one-time password system like S/KEY [76]. The consequence of revert a snapshot can be as serious as the attacker had previously sniffed password that can now use, and this way compromise the security of the infrastructure.

Another problem of using snapshots can be found on systems that use protocols that rely on the "freshness" of their random number source. Authentication algorithms is indeed a problem to be concerned when enabling snapshots. Zero Knowledge Proofs of Knowledge (ZKPK) [77] are not secure if the same random nonces are used more than once, and therefore, many of the authentication protocols derived from ZKPK such as Fiat-Shamir authentication [78] or Schorr authentication [79] will be insure if a snapshot is reverted. However, not only the cryptographic protocols are affected. For instance, the reuse of TCP initial sequence number could allow an attacker to do a TCP hijacking attack [80].

4.2.7 Vulnerability Analysis of VMware ESX and Xen

When thinking about vulnerabilities, hypervisors are one of the most sensitive pieces of software on a computer, as they are the door to access virtual machines. They are used for many propose and in many different environments, by professionals and amateurs. Hypervisors, as said before, should have the less possible lines of codes to make it lighter to run but also to allow the less vulnerability. However, are they really secure? How many vulnerabilities were reported and what would be the impact if it was exploitable?

In [81], the authors do an analysis of the number of patches released for the VMware ESX. Their study was about the number of patches released for this version based on the information on the VMware's website, which only had information from VMware ESX 3 and above. Moreover, the authors group these patches in critical, security and general patches. This gives an idea of the number of patches but does not give an idea of how secure is the product. Our approach is different. We study the number and severity of CVEs reported for both VMware ESX and Xen. The database available of Common Vulnerabilities and Exposures (CVE) is very detailed and with a good registry of old versions of both products. This gives a good idea of the security for along the years. We also compare both products, trying to understand if any of them can be considered more secure and which one has higher severity vulnerabilities reported.

While there are other hypervisors that one could consider, VMware ESX and Xen ultimately represent the forefront of efforts and claims with respect to hypervisors and security, and likely suffer higher levels of scrutiny by security researchers than other hypervisor solutions.

Questioned about if it was correct to compare VMware ESX with Xen, Keith Adams answered me saying "the closest products in spirit and capabilities are probably ESXi and Xen. I think ESX is fair to compare, too, if you exclude security problems from the user-level

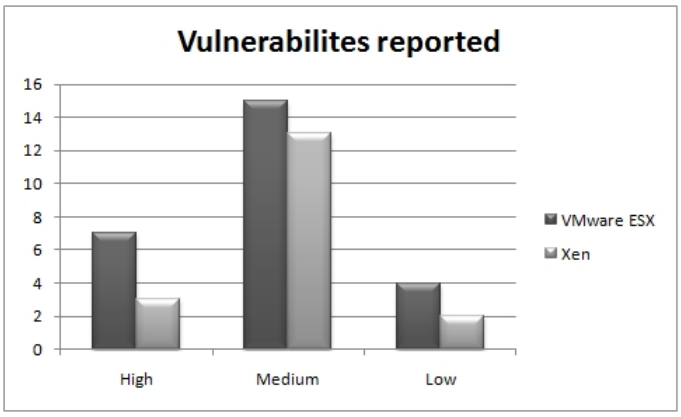


Figure 4.5: *Severity of the vulnerabilities reported*

component of ESX; the user-level portion of ESX is essentially Linux, so it inherits all of its security advisories".

With that context in mind, I am going to undertake a security analysis of these two solutions regarding software vulnerabilities. I considered VMware ESX and VMware ESXi as the same product, and considered Xen or Citrix Xenserver as the same product.

This analysis was based on the CVEs [62] reported on the National Vulnerability Database [82] and it was made in August of 2009.The CVE naming conventions and process is known worldwide as being the most comprehensive list of vulnerabilities across software products of all types.

VMware released VMware ESX 1.0 (Elastic Sky X) in 2001 and has subsequently released VMware ESX 1.5 (2002), 2.0 (2003), 2.5 (2004), 3.0(2006), VMware ESX 3.5 (2007) and VMware ESX 4.0 (2009) . The first public release of Xen was in 2003 and since then, new releases have been made. The current stable version is 3.4.

The Common Vulnerability Scoring System (CVSS) does the following equation in order to find the CVSS Severity Score: (0.6*Impact +0.4*Exploitability-1.5)*f(Impact). The value of f(Impact) can have one of the following two values: If Impact is zero then f(Impact) is also zero, otherwise it will be 1.176. The value of Impact and Exploitability is more complex and its explanation can be found in [83].

Since the release of VMware ESX 1.0 in 2001, VMware has fixed 26 vulnerabilities in supported ESX products – 7 HIGH severity, 15 MEDIUM severity and 4 LOW severity. Xen had the first public release in 2003, however the first CVE is dated from 2007. From that date until June of 2009 it was reported 18 vulnerabilities regarding Xen –3 HIGH severity, 13 MEDIUM severity and 2 LOW severity.

The graphic on the Figure 4.5 shows that, although these two products were first released

Month/ Year	CVSS Severity	Impact Subscore	Exploitability Subscore
12/03	7,2	10	3,9
12/05	7,6	10	4,9
12/05	6,8	6,4	8,6
12/05	2,1	2,9	3,9
12/05	4,9	6,9	3,9
12/05	4,3	2,9	8,6
07/06	5	2,9	10
07/06	3,6	4,9	3,9
04/07	5	2,9	10
04/07	6,6	10	2,7
01/08	7,5	6,4	10
06/08	4,4	6,4	3,4
06/08	6,9	10	3,4
06/08	9	10	8
06/08	7,2	10	3,9
09/08	2,1	2,9	3,9
10/08	5	2,9	10
10/08	6,8	10	3,1
11/08	6,9	10	3,4
11/08	9,3	10	8,6
12/08	7,2	10	3,9
02/09	4,7	6,9	3,4
04/09	6,8	10	1
04/09	2,1	2,9	3,9
04/09	4,6	6,9	3,1
06/09	4	6,9	1,9

Table 4.1: *VMware ESX CVSS severity analysis from 2003 until 2009*

with two years difference, VMware is the product with more vulnerabilities, especially the ones classified as HIGH. However, Xen as a number of MEDIUM severities very close to the ones of VMware. We are going to analyze each product individually.

VMware ESX As we can see on Table 4.1, from the 26 occurrences since 2003, 7 had CVSS Severity score equal or above 7 which is considered severity High. In addition, 6 had a score above 6 (close to the High severity) and 15 are considered of severity Medium.

Figure 4.6 shows the incidence reported for VMware ESX and their severity. Around 70 percent of vulnerabilities considered High was reported in 2008 while the ones considered Medium were equality dived between 2005 and 2009. On those seven high vulnerabilities,

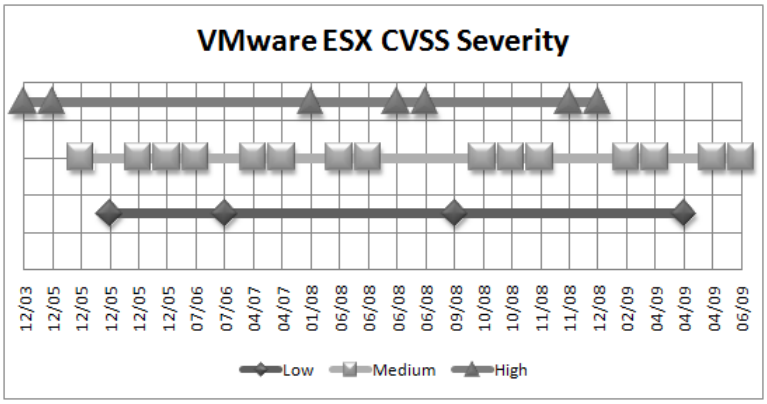


Figure 4.6: *VMware ESX CVSS severity from 2003 until 2009*

around 60 percent were network exploitable while 40 percent were local exploitable.

In order to have a deeper analysis of the vulnerabilities reported, it was made a study regarding all the VMware ESX's CVE analyzed, grouped by the type of attack and the access vector.

As a result, all the Low Severity CVEs are classify as Information Disclosure, while most of the High Severity CVEs are Host Privilege Escalation exploitable by network. Most of these are possible because the newer versions of ESX have a web server in order to use the client vSphere to management the server and so have a new door for attacks. For the same reason is able to do XSS attacks against the ESX.

XEN As we can see on the table 4.3, from the 18 occurrences since 2003, 3 had CVSS Severity score equal or above 7 which is considered severity High. In addition, 3 had a score near 7 (close to the High severity) and 13 are considered of severity Medium.

Figure 4.7 shows the incidence of the severity. All the vulnerabilities considered High was reported in 2008 while the ones considered Medium were equally divided between 2007 and 2008 (with six reported in each year). On those three high vulnerabilities, all were local exploitable.

In order to have a more deep analyze of the vulnerabilities reported, it was made a study regarding all the Xen's CVE analyzed, grouped by the type of attack and the access vector.

As a result, most of the vulnerabilities reported are related with DoS, some of them considered of severity High. Many of these DoS would affect services on the host. It was considered VM Escapes only the vulnerabilities that explicit execute code on the Host and not the ones that access memory in order to crash the hypervisor.

Attack	Access Vector	CVE	Impact
Denial of Service	Locally exploitable	CVE-2008-4916(M-GH) CVE-2009-1805(M-OG) CVE-2008-4914(M-GH) CVE-2008-4917(H-GH) CVE-2008-2100(H-GH) CVE-2007-1271(M-O?) CVE-2005-4773(M-OH)	Violates the condition of Isolation and allows an attacker to consume all the resources available.
	Network exploitable	CVE-2008-4309(M) CVE-2007-1270(M)	A remote attacker who issued a specially crafted request could cause a service to crash.
VM Escape	Locally exploitable	CVE-2009-1244(M) CVE-2008-2100(H)	Violates the condition of Isolation allowing code execution on the host system from the guest system
Guest Privilege Escalation	Locally exploitable	CVE-2008-4915(M) CVE-2008-4279(M) CVE-2007-5671(M) CVE-2007-1271 (M)	Allows an user in a guest machine to gain privileges
Host Privilege Escalation	Locally exploitable	CVE-2008-2100(H) CVE-2008-0967(M) CVE-2003-1291 (H)	Allow users with non-privileged accounts to gain root privileges
	Network exploitable	CVE-2008-2097(H) CVE-2007-5360(H) CVE-2008-4281(H)	
Information disclosure vulnerability	Locally exploitable	CVE-2005-3620(L) CVE-2006-3589(L) CVE-2009-0518(L) CVE-2008-2101(L)	An attacker can access sensitive information
	Network exploitable	CVE-2006-2481(M) CVE-2005-4583 (M)	
XSS Attack	Network exploitable	CVE-2005-3619(M) CVE-2005-3618(H)	An unauthorized user could construct a specially crafted URL that might change some information or do Cross Site Request Forgery

Legend: (L) – Low, (M) – Medium, (H) – High, (OH) – Only on Host, (GH) – Guest operating system can affect the Host, (OG) – Only on Guest

Table 4.2: *Analysis of each VMware ESX CVE reported*

Month/ Year	CVSS Severity	Impact Subscore	Exploitability Subscore
03/07	4,3	2,9	8,6
09/07	6,9	10	3,4
10/07	6	9,2	2,7
11/07	4,7	6,9	3,4
11/07	4,7	6,9	3,4
12/07	4,6	6,4	3,9
12/07	2,1	2,9	3,9
04/08	4,3	2,9	8,6
05/08	2,1	2,9	3,9
05/08	7,2	10	3,9
05/08	4,9	6,9	3,9
06/08	5	2,9	10
07/08	4,3	2,9	8,6
08/08	6,8	6,4	8,6
10/08	7,2	10	3,9
11/08	6,9	10	3,4
12/08	7,2	10	3,9
05/09	5	2,9	10

Table 4.3: *Xen CVSS severity analysis from 2003 until 2009*

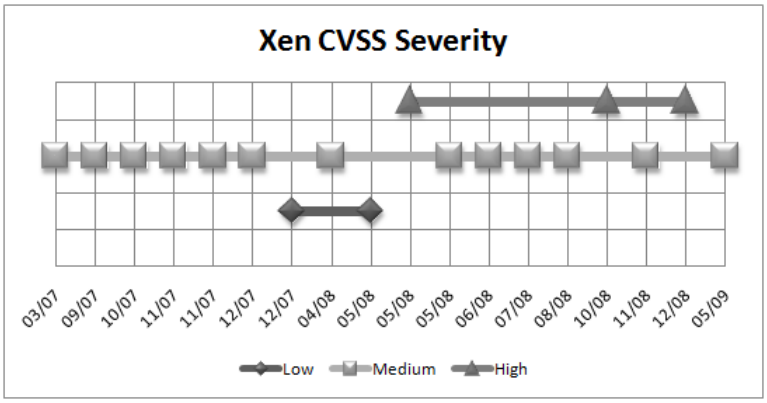


Figure 4.7: *Xen CVSS severity from 2003 until 2009*

Attack	Access Vector	CVE	Impact
Denial of Service	Locally exploitable	CVE-2008-5716(H-GH) CVE-2008-4405(H-GH) CVE-2008-3687(M-GH) CVE-2007-5498(M-GH) CVE-2008-1944(H-OH) CVE-2008-1943(L-OH) CVE-2007-5907(M-GH) CVE-2007-5906(M-GH) CVE-2007-3919(M-OH) CVE-2007-4993(M-GH)	Violates the condition of Isolation and allows an attacker to consume all the resources available.
	Network exploitable	CVE-2009-1758(M-OG) CVE-2008-1952(M-OG) CVE-2008-1619(M-OH)	A remote attacker who issued a specially crafted request could cause a service to crash.
VM Escape	Locally exploitable	CVE-2007-4993(M) CVE-2008-3687 (M)	Violates the condition of Isolation allowing code execution on the host system from the guest system
Guest Privilege Escalation	Locally exploitable	CVE-2008-3687(M)	Allows an user in a guest machine to gain privileges
Host Privilege Escalation	Locally exploitable	CVE-2008-4993(M) CVE-2008-1944(H) CVE-2008-1943(L) CVE-2007-3919(M)	Allow users with non-privileged accounts to gain root privileges
	Network exploitable		
Information disclosure vulnerability	Locally exploitable	CVE-2007-6207(L) CVE-2007-6416(M) CVE-2007-0998(M)	An attacker can access sensitive information
	Network exploitable		
XSS Attack	Network exploitable	CVE-2008-3253(M)	An unauthorized user could construct a specially crafted URL that might change some information or do Cross Site Request Forgery

Legend: (L) – Low, (M) – Medium, (H) – High, (OH) – Only on Host, (GH) – Guest operating system can affect the Host, (OG) – Only on Guest

Table 4.4: *Analysis of each Xen CVE reported*

4.2.8 Conclusion

Most of the vulnerabilities in VMware ESX have been discovered since 2006 and 70 percent of vulnerabilities considered High was reported in 2008. In the case of Xen, the first vulnerability was reported in 2007 and 100 percent of the High severity vulnerabilities were discovered in 2008. The reason for this, in my opinion, is that with the increase in popularity, relevance and deployment of virtualization, also vulnerability discovery gain a new liveliness with the goal on finding ways to exploit virtualization technologies.

Virtualization does not mean security or replace security. In fact, virtualization brings a more complex and risky security environment. Virtualization adds a new layer to what we had before. Operating system and applications always coexisted. Now they are being packed in a box called virtualization. However, there are vulnerabilities in operating systems and applications no matter if they are running on a virtual environment or not. Combining to that, we are now adding vulnerabilities with the virtualization software, not forgetting VMBRs. Moreover, using virtualization, we are consolidating our virtual environment in one physical target were by using a DoS or VM Escape, it is possible to exploit one system, accessing and controlling other virtual systems on that target or even the server itself.

Chapter 5

VM Solutions for Online Banking and e-Commerce

Talking about end user's computers is also talking about security. Although nowadays, personal computers and even office computers are much more protected against virus and malwares than five to ten years ago, the truth is that its security level is still low. From my own experience, while I was responsible for a Public key infrastructure (PKI) solution, I went to some desktops in a public institution, and on some of them, despite the fact it was installed an antivirus software, the user had a lot of add-ons (toolbars) installed on the Internet Browser (IE) and some other uselessness tools on the desktop background. It was the typical user that clicks "Yes" to any window asking permission for an operation. Later, in another project, I was responsible for an antivirus project of another public institution and our days was the management of the antivirus of a big farm of machines, most of them desktops. Although there were majority updated in terms of antivirus signature file, there were always some virus found every day. In addition, when we discovered a computer that did not have antivirus software installed, the number of virus in that machines was normally high. Virus and malwares increase every day and are more and more intelligent in the way they infect a machine and spread to others but also in the social engineering they do to deceive the end user to install them. I consider myself a little paranoid about home security, for instance in my Windows XP, I have an antivirus software, an host intrusion prevention system (HIPS) software and firewall software. Moreover all the software installed (including the operating system) are totally updated to the last stable version. However, some years ago, I have almost installed a virus in my machine received in a normal mail supposedly sent by one of the persons I trust. This virus was not detected by the antivirus product but the HIPS alerted me by the fact of that simple file was trying to write in a

specific directory and on the registry. At that time I figured out that something was wrong and I canceled all the actions the virus was trying to do, and alerted the person who sent me the email for that fact. Other common friend received the same email and he lost almost all the files he had on his computer. With that, he learns to be more careful the next time he receives an email, but as any human being, time turns you softer when facing a lower risk. Normally, the home user only realizes its computer is infected when its start to act uncommonly, but some virus can be installed and does not show any presence. Some of them are the ones that attack home banking and they only start to do something when it detects the user is accessing a specific online banking website.

According to Rachwald [84] "people who are not particularly tech savvy have a tough time differentiating between good online security practices and bad online security practices". Although I know some cases where this does not apply, I believe this is the common rule. Physical security is more intuitive for people than the IT world. A person would not leave her home key on the door or left alone in an easy place to find, but more easily, they write their password on a post-it and put it under the keyboard or even on the monitor.

In October 1994, Stanford Federal Credit Union introduced the first online banking service in United States. Since then, banks have come to understand that security is a major concern, and they have to evolve with it, adapting their solution to face new attacks. The number of users has been growing over the years. In 2006, the number of online banking consumers in the US grew to 44 millions. According with a study of Ponemon Institute made in 2006, 34 percent of customers would change their bank after one breach and 45 percent would leave after two breaches. This indicated how serious would be the impact of an attack to one of the top 10 online banks. Nowadays, phishing is still a problem and normally they can be done in two ways:

- Redirect the user to a different webpage with the same layout as the original one, using a link received by an email, or using a virus that redirects the user to that webpage.
- Allowing the user to access the legitimate webpage but use a virus installed on the computer to deface the webpage, asking for some extra information the legitimate webpage would not ask like the data of the online banking consumer's TAN list.

Virtualization could have a role to play in order to protect the consumer. We are going to describe some solutions for a homer user to use online banking or ebay in a more secure way.

5.1 The Three Colors Solution

The idea of this solution is to provide an isolated environment where the user could perform sensitive tasks as online banking, or e-commerce or semi-sensitive task as updating its own blog or even non sensitive task as surfing in the internet, or downloading P2P files. Since these are isolated environment, it is safe to do any of this operation because it will not affect the host system neither the other virtual machines. We have organized the solution to use three virtual machines, each one with a different purpose. The one we are going to use only for online banking is the one that requires more security attention and because of that, we are going to designate as the Red VM. If the user does some online shop using eBay or Amazon, it will require a safe VM but with few lower requirements and so will designate this one as the Yellow VM. The Green VM is the one where the user can surf freely on the Internet, accessing anywhere, even the websites considered dangerous. Table 5.1 resumes the purpose of each virtual machine. The Red VM, as said before, it will be used exclusively for online banking. The Yellow VM it is going to be used for fewer sensitive operations but those that still need some care. The list presented on the Yellow VM column is not restricted to the items presented. The user could do some other activities that he identifies as sensitive operations. The Green VM is where the user should have all the freedom to do anything.

Red VM	Yellow VM	Green VM
Online Banking	e-Commerce	Check e-mails on yahoo/gmail/hotmail
	Update blog / website	Googling
	SSH connections	Search for information of exploits
	Stock Market websites	Youtube videos
	Other sensitive operations	Online games
		Adult websites
		IRC channels
		etc...

Table 5.1: *Resume of the purpose of the three virtual machines*

We are going to describe the solution purposed using only open source software available free and that any user could install on his computer. Moreover, this solution can be used with any common desktop operating system such as Microsoft Windows, Mac OS X or Linux.

5.1.1 Design of the Solution

For each VM it is necessary an operating system hardened to run faster and have higher security level. Since Microsoft Windows requires licenses for each installation, this operating system is not an option. Therefore, we choose Linux to be installed on each VM. The question now was which distribution to use and how it is going to be used. There are many options and the home user can use any of them, each one with its benefits and drawbacks. Some of them are described on table 5.2.

Option	Pros	Cons
LiveCD from any Linux/BSD Distribution	It is easy to download and run; Fresh run each time	It brings more software than needed and it runs in memory
Install a full Linux	The user choose the distribution it is more used and install it	It requires some knowledge to install only the necessary packets.
SUSE Studio	Customizable distribution with only the packages the user wants	Some steps are necessary and requires some knowledge

Table 5.2: *Comparison of different solutions*

Any distribution is good if the user is already familiar with it, however for the Red and Yellow VM it is recommended a hardened Linux since we only need the kernel, a Windows Manager to run the browser, a browser and a pdf reader. This will make the VM to be light and run with few requirements, fulfilling better its goal. Because of this, we have chosen Novell's SUSE Studio [85].

The solution adopted is a type 2 virtualization and the software chose was Sun VirtualBox because it is free, small and there are versions for Microsoft Windows, Linux and Mac. The design of the solution is described on Figure 5.1.

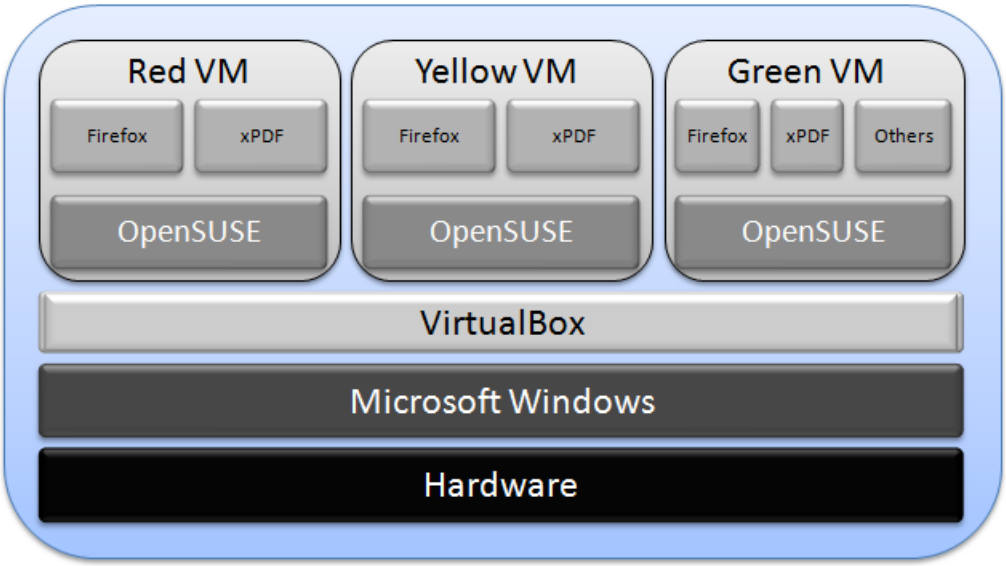


Figure 5.1: *The solution's components*

We are going to describe the steps used to install the software and to run the VM but for further detail, we recommend to consult the Sun's VirtualBox Manual [86]. We are going to describe the steps necessary in order to install and run the three VMs. I am assuming the user is running Microsoft Windows with administrations rights and knows how to install an application.

5.1.2 Setup of the Solution

The first step is to customize the Linux distribution we are going to use with only the browser and few more things. We want to simplify the process and not use advanced configurations like replace the /boot/grub/menu.lst file with one that does not show the Failsafe option, or replace introduce some add-ons on Firefox. The goal is to produce something any user could do without many "blind" steps.

The user should access to SUSE Studio website [85] and create an account. At the time of writing this thesis, SUSE Studio was an invite-only service, but I have request an invitation that was received in less than 10 minutes. Once the user's invite arrives, he can sign into SUSE Studio with his Google or Yahoo account, or any OpenID provider. Once the users sign successfully, it can start the creating of the customized VMs.

5.1.2.1 Creating the Red VM

1. Go to "Home" and click on the "Create new appliance" link in the upper-right.

2. On the subsection "OpenSUSE 11.1" (which is the current version as the thesis is written) click on the link "Minimal X"

3. Go to the bottom of the page and make sure the option "32-bit" is select on "Select your architecture"

4. On the subsection "Name your appliance", introduce "Red VM" and click on "Create appliance"

5. After this, the Start menu will show asking you to navigate through the tabs to generate the appliance. The first is software, and the user should click in the tab "Software" or in "Switch to the Software tab to continue" link.

6. Based on the selection the user has made previously, SUSE Studio already selected the Linux basic to boot and the few packages as the Windows Manager. However, we want to install also the web browser and a pdf reader. OpenSUSE 11.1 comes with version 3.0.x version of Firefox, but we want the most recent one, which is at the time the version 3.5.x. For this, it is necessary to add the Mozilla repository, by clicking on "Add repositories" on the section "Software sources".

7. Now we need to write "Mozilla" under "Add and remove repositories" and click on "+ add" for "Mozilla openSUSE_11.1".

8. Another repository is necessary to add in order to have the guest tools of VirtualBox already installed. It is necessary to search for "VirtualBox" and then and click on "+ add" for "Virtualization:VirtualBox 11.1"

9. Click on the link "back to the software overview" on the top of the page.

10. Now we are able to add the last version of Firefox and XPDF. The use should now go to "Search for software" and find for firefox.

11. Choose "MozillaFirefox" by click "+add" on the left of it. If all goes well, the button should change for "remove" and a "mark" should appear. Note also that a new section appeared on the left saying "Software changes" and showing "Added MozillaFirefox".

12. In order to add XPDF, the user must search for "xpdf" and select it the same way he have done on the previous step.

13. Some online bank's website requires flash player. In order to add Flash, the user must search for "flash-player" and select it the same way he have done on the previous step.

14. It is also necessary to add the Guest Tools of VirtualBox, and so it is necessary to search and add "virtualbox-ose-guest-tools" as we have done in the previous step.

We have found later an important bug when we use IceWM with auto login. What happens is that after the auto login, the keyboard does not work anymore and few less important bugs happened too. After deep investigation, it was found that the problem was on the login manager and because of this, it was necessary to add KDE Display Manager (KDM), which is a graphical login interface. After this replace, all the bugs noticed before no longer existed. The drawback of this solution is that adds 50MB of packages. Because of this bug, an extra step is necessary that is search for kdebase3-kdm and add it.

1. The next step is related with the configuration and so it is necessary to click on "Configuration" on top of the page, next to Start.

2. The first sub tab that is selected by default is "General". In this tab it, the user can select the language he wants, the keyboard layout he will use (if it is a keyboard from US then he should choose English (US) but if it is a Portuguese keyboard then he should choose Portuguese) and some other definitions as default time zone and network. In this screen, the configuration purposed are:

- Default locale
  - Language: English (US)
    - Keyboard Layout: English (US)
  - Default time zone
    - Region: USA
    - Time Zone: Eastern (New York)
  - Network
    - Discover network setting automatically (DHCP)
  - Firewall

  - Enable firewall
    - Open SSH port (22): disabled
    - Open HTTP ports (80,443): disabled
- Users and groups:
  - Login: root
    - Password: [user's password]
    - Group: root
    - Home directory: /root
    - Shell: /bin/bash
  - Replace the user tux by a new one.
  - Login: onlinebanking
    - Password: [user's password]
    - Group: users
    - Home directory: /home/onlinebanking
    - Shell: /bin/bash

1. The next sub tab to configure is "Personalize". In this section, it is possible to configure the logo and the background. We have selected the "Carnegie Mellon University" logo and for background, the "FCUL C8 Building" - which won the Valmor award – with a red layer.

2. On the "StartUp" sub tab, we can select the runlevel. The user should select:

- 5: Graphical Login

1. We can skip the sub tab Server, because we will not install any MySQL database.

2. On the "Desktop" sub tab, the following options should be selected:

- Automatic desktop user log in
  - Automatic log in user: onlinebanking – enabled
  - Autostart desktop programs
    - Command: firefox

        - Start for User: onlinebanking
        - Comment: Starts Mozilla Firefox
        - Enabled: Checked

1. On the “Storage & Memory” the following configurations are purposed:

- Virtual appliance
    - Ram size: 512 MB – If the user has 1GB or more of RAM, them this is the purposed valued. If the user has an old PC with less than 1GB, then we suggest at least 256 MB for the RAM size value.
        - Virtual disk size: 2 GB
    - Disk Image
        - Swap partition: 512 MB

1. On the “Scripts” sub tab, the following options should be:

- Run script at the end of the build: Disabled
    - Run script whenever the appliance boots: Disabled

1. Now the user should click on the “Overlay files” tab. In this section no entries should exist. If there is any, the user should delete it.
2. On the “Build” tab, the user should select:

- Format: VMware/VirtualBox (.vmdk)
    - Version: 1.0.0

1. Then click on the button “Build”. After this, a progress bar should appear. It is necessary to wait until the VMware/VirtualBox image is created and a Download link is available.
2. When the download link is available, it is necessary to click on it and save the file on the user’s hard drive. The file will have around 300MB.

3. After it completes its download, it is necessary to uncompress the tar.gz. Microsoft XP does not provide a program for this, and so we recommend an open source file archive called 7-ZIP [87].

At this point, the creating of the Red VM is made. However, is it still necessary the Yellow VM and the Green VM.

5.1.2.2 Creating the Yellow VM

For the Yellow VM we can clone the Red VM and only change the name, the personalization and the user name.

1. Go to the "Home" tab and put the cursor on top of "Red VM". A yellow box will appear showing the "Clone" link. Click on that link.
2. Rename the appliance name:

- Appliance Name: Yellow VM

1. Go to the "Configuration" tab and under "General" the user should change the user "onlinebanking" for the new one which can be:

- Login: onlineshopping
    - Password: [user's password]
        - Group: users
        - Home directory: /home/ onlineshop
        - Shell: /bin/bash

1. On the sub tab "Personalize", the user should change the background for something different, so it can easily identify it is in the yellow VM. We have selected the "Carnegie Mellon University" logo and for background, the "FCUL C8 Building" with a yellow layer.
2. On the sub tab "Desktop" the user should check if the user is equal to the one introduced on the step 3:

- Start for User: onlineshopping

1. The user can go now directly to the tab “Build” and do the same as for the Red VM:

- Format: VMware/VirtualBox (.vmdk)
    - Version: 1.0.0

1. Then click on the button “Build”. After this, a progress bar should appear. It is necessary to wait until the image is created and a Download link is available.
2. When the download link is available, it is necessary to click on it and save the file on the user’s hard drive. The file will have also around 300MB.

5.1.2.3 Creating the Green VM

To create the Green VM, we are going to use other template. The following steps should be made:

1. Go to “Home” and click on the “Create new appliance” link in the upper-right.
2. On the subsection “OpenSUSE 11.1” (which is the current version as the thesis is written) click on the link “KDE 4 desktop”.
3. Go to the bottom of the page and make sure the option “32-bit” is select on “Select your architecture”
4. On the subsection “Name your appliance”, introduce “GreenVM” and click on “Create appliance”
5. After this, the Start menu will show asking you to navigate through the tabs to generate the appliance. The first is software, and the user should click in the tab “Software” or in “Switch to the Software tab to continue” link.
6. On the “Software” tab click on “Add repositories” on the section “Software sources”.
7. Now we need to write “Mozilla” under “Add and remove repositories” and click on “+ add” for “Mozilla openSUSE_11.1”.

8. Another repository is necessary to add in order to have the guest tools of VirtualBox already installed. It is necessary to search for “VirtualBox” and then and click on “+ add” for “Virtualization:VirtualBox 11.1”

9. Click on the link “back to the software overview” on the top of the page.

10. The Green VM will be used for the end user surf on the Internet without any restriction and so, some extra programs may be necessary. We are going list the programs we purpose for this VM that should be installed like we did for “MozillaFirefox” and “XPDF”:

- Mozillafirefox
    - OpenOffice
    - Flash-player
    - OpenOffice_org-base
    - kde4-kate
    - xpdf
    - virtualbox-ose-guest-tools

1. The next step is related with the configuration and so it is necessary to click on “Configuration” on top of the page, next to Start.

2. The first sub tab that is selected by default is “General”. In this tab it, the user can select the language he wants, the keyboard layout he will use (if it is a keyboard from US then he should choose English (US) but if it is a Portuguese keyboard then he should choose Portuguese) and some other definitions as default time zone and network. In this screen, the configuration purposed are:

- Default locale
    - Language: English (US)
        - Keyboard Layout: English (US)
    - Default time zone
        - Region: USA
        - Time Zone: Eastern (New York)

- Network
  - Discover network setting automatically (DHCP)
- Firewall
  - Enable firewall
    - Open SSH port (22): disabled
    - Open HTTP ports (80,443): disabled
- Users and groups:
  - Login: root
    - Password: [user's password]
    - Group: root
    - Home directory: /root
    - Shell: /bin/bash
  - Replace the user tux by a new one.
  - Login: freeuser
    - Password: [user's password]
    - Group: users
    - Home directory: /home/freeuser
    - Shell: /bin/bash

1. The next sub tab to configure is "Personalize". In this section, it is possible to configure the logo and the background. We have selected the "Carnegie Mellon University" logo and for background, the "FCUL C8 Building" - which won the Valmor award – with a green layer.

2. On the "StartUp" sub tab, we can select the runlevel. The user should select:

- 5: Graphical Login

1. We can skip the sub tab Server, because we will not install any MySQL database.

2. On the "Desktop" sub tab, the following options should be selected:

- Automatic desktop user log in

- Automatic log in user: freeuser – enabled
- Autostart desktop programs
  - Command: firefox
  - Start for User: freeuser
  - Comment: Starts Mozilla Firefox
  - Enabled: Checked

1. On the “Storage & Memory” the following configurations are purposed:

- Virtual appliance
  - Ram size: 512 MB – If the user has 1GB or more of RAM, them this is the purposed valued. If the user has an old PC with less than 1GB, then we suggest at least 256 MB for the RAM size value.
    - Virtual disk size: 3 GB
  - Disk Image
    - Swap partition: 512 MB

1. On the “Scripts” sub tab, the following options should be:

- Run script at the end of the build: Disabled
  - Run script whenever the appliance boots: Disabled

1. Now the user should click on the “Overlay files” tab. In this section no entries should exist. If there is any, the user should delete it.
2. On the “Build” tab, the user should select:

- Format: VMware/VirtualBox (.vmdk)
  - Version: 1.0.0

1. Then click on the button “Build”. After this, a progress bar should appear. It is necessary to wait until the VMware/VirtualBox image is created and a Download link is available.

2. When the download link is available, it is necessary to click on it and save the file on the user's hard drive. The file will have around 300MB.

3. After it completes its download, it is necessary to uncompress the tar.gz. Microsoft XP does not provide a program for this, and so we recommend an open source file archive called 7-ZIP [87].

5.1.3 Running the R/Y/G VMs

We have concluded the creating of the three VMs. We are going to describe now how to use them as a systematic guide.

1. The first step is to go to Sun's VirtualBox webpage [86] and download the Windows version of VirtualBox. The version for x86 or 64-bit is the same. This software is the hypervisor type 2 we are going to use in order to run the VMs.

2. After installing, the user can access it on Start-> Programs -> Sun VirtualBox on Windows XP and run VirtualBox. When it starts for the first time, it will ask for a free registration that the user can fill. Then, you will have access to the Sun xVM VirtualBox console.

3. The first step now is to create a new Virtual Machine. For this, we click on "New".

4. A welcome window will pop up. We click on "Next".

5. On the VM Name and OS type we introduce:

- Name: Red VM
    - OS Type:
        - Operating System: Linux
        - Version: openSUSE

1. Click on "Next".

2. On the Memory window, we introduce 512 MB as the Base Memory Size.

3. On the Virtual Hard Disk, we select:

- Boot Hard Disk (Primary Master): Enabled
    - Use existing hard disk: Selected.
    - If this is the first time the user is using VirtualBox, then a message saying <No Media> appears under the option selected. Press on the folder icon on the right.
    - A new window opens with the Virtual Media Manager.
    - Select the option “Add”.
    - Browse to the folder you have uncompressed the "[color] VM tar.gz" file and choose the file "[color]_VM.i686-1.0.0.vmdk".
    - Click on Select
    - The user will return to the previous window, with the hard disk selected. Click on Next

1. On the Summary, the user should review the configuration that will be used. If all is correct, then should click in Finish.
2. Select the VM by doing one click on top of it and on the left panel, click on "General" and go to "Advanced".
3. If the VM selected is a Red or Yellow VM, then change the "Shared clipboard" option to "Guest to Host"
4. If the VM selected is the Green VM, then change the "Shared clipboard" option to "Disabled"
5. To run the virtual machine, it is necessary to select it on the left panel and click “Start”.

These same steps should be applied for the Yellow VM and Green VM. At the end of this step, the user have all the three VMs ready to run on the VirtualBox.

5.1.4 Taking Snapshot

Now that we have a clean installation of the three VMs, some extra cares are needed to enhance the security. The virtual machine would need to be kept up to date and some steps should be taken:

- On the Mozilla Firefox Browser
    - Install the "NoScript" browser plug-in
    - Install the "Perspectives" browser plug-in from CMU Perspectives webpage [88].
    - Set the home page to the banking/shop web site to be used. Remove all bookmarks and add only those that are needed for the online banking or e-commerce operation.
    - Disable the option "Remember passwords for sites" on Edit -> Preferences -> Security.

After doing these steps, and before the user do anything else, it should take a snapshot of the VM. To do this, it should do:

- Menu Machine -> Take Snapshot.
    - Snapshot Name: Clean installation
    - Snapshot Description: This is a clean state of the virtual machine.

The goal of this snapshot it to use it each time the user wants to do online banking or e-commerce operations. This way, he can use a clean VM. It is necessary to modify the appliance virtual machine configuration file to revert to its original state after each shutdown. After the “take snapshot” operation, the user should go to:

- Menu Machine -> Close
- On the “Close Virtual Machine”
    - Select “Power off the machine”
        - Revert to the current snapshot: Enable

For now on, the user just has to start the machine using this snapshot. The restore is fast and the VM will be secure. However, there can be updates for this machine (as new version of Firefox or kernel updates) and for security reasons they should be applied. In this case, the user should revert to the current clean snapshot, update the system using the password created for the user root and then take a new snapshot, using this snapshot as the current state.

These same steps should be applied on the Red VM, Yellow VM and Green VM. However, since it is always reverting to the taken snapshot, all the files downloaded or created on any of the VMs will be lost.

5.1.5 Using the Three Color Solution

Now it is possible to use any of the VMs created in a safe way. When the user wants to do online-banking, it would only have to start the VirtualBox, then run the Red VM and do the online banking operation he needs. When the user finishes its online baking operations it should access the VirtualBox menu on the same windows (pressing the right Ctrl to leave the focus from inside the window) and select Machine -> Close and verify if the option "Revert to the current snapshot" is enabled. The same procedure should be done for when the user needs to use the Yellow or Green VM.

Nevertheless, some extra cares should be followed:

- For the Red VM, only use HTTPS connections. This is particularly important if the host machine is using unsafe networks such as a hotel/airport wireless network.
- It should be avoided running more than one VM at a time since each virtual machine will consume memory (512 MB by default).
- Every time the user runs one of the VMs, it will run the cleaned snapshot created on the previous subsection. Therefore, if the user needs to keep something saved on the Red or Yellow VM, then he can drag&drop the file from the VM to the host machine. This procedure should only be used for pdf or html files saved by the user from a trusted website (such as online banking extract or Amazon receipt page).
- The host machine must not be used for web surfing or any other activity that could put in risk its security.

5.1.6 Security Analyses of this Solution

The solution of using VMs for online banking and e-commerce operation is based on the assumption that the host machine is reliable. According with the reports from Forrester of 2007 [89], 95 percent of enterprise desktop runs Microsoft Windows. It is well known the vulnerabilities that explore this operating system, but although this may seem a strong assumption, it is not that hard to achieve. Let us assume the computer received a fresh installation (without being connected to the Internet), and then it was installed and configured correctly a personal firewall and an antivirus software. After having all these software installed, the administrator forced a full updated of them (preferentially still offline) including the operation system. At this point, we can say the computer has a high security level.

One of the problems about Microsoft Windows security is the wrong configuration of user's privilege. Normally the current user has administrator privileges, which allows him to install, and change anything in the computer and with that, the user can install virus or malwares. If the user's account is configured without these privileges, the computer will be less unprotected and so more secure.

Our assumption for the three color solution is that the host machine is reliable because it is well installed and well configured. If not, then there is some attack vectors such as if the host machine is compromised the virtual machine is vulnerable. For instance, a key logger on the host machine could capture account credentials typed into the virtual machine. In table 5.3, we have summarized the security problems an user can face and how this solution mitigates them.

Security Problem	How to mitigate
If the browser have the bank account login memorized, anyone who have access to the computer could start the VM and access the bank account	Each time the VM restarts, it will use a clean snapshot and therefore, even if the user has saved the password on the browser on the last time he used the VM, this information will be lost.
The user received an email with a link that goes to a fake site operated by criminals.	Our solution does not protect the activities the user would do in its host machine, but since the user has a Firewall, what he could do is to configure a rule in his host that would block access to the legitimate bank website. This way, he would know that if he pressed a link and that link would show the webpage of his bank it would be a fake one.

An attacker access the user's computer and with the credentials of that user, replace the snapshot with one compromised.	The solution to this problem (not presented in the setup of the solution) could be: The user have the snapshot file encrypted, and each time it would use, would have to decrypted it. This way, if it was replaces, unless the password used to decrypt would be the same, the user would detect the original snapshot was changed. The solution was configured to do auto login and the snapshot was proposed to be taken after that, in order to make the user's life easier. However if the user had to enter its login each time he starts the VM, and then if the snapshot was changed he would notice because the system would not ask the password to login.
The user installs a malware or keylogger on the VM	The VMs are secure against this kind of attacks because the user does not have permission to install programs as root although there are keylogger for Linux that can run in user mode. However, this means the user would have to download it and execute it. Nevertheless, the VM should be used only to access the online banking and e-commerce websites. In addition, each time the user starts the VM it will use a snapshot it was taken in a clean environment. What could be done extra is to configure a firewall on the guest machines to allow only access to the IP's corresponding to the online bank websites.
The user installs a malware or keylogger on the host	This would break the assumption that the host machine is reliable and the user account is well configured. Our solution cannot do much against it.

An attacker could do a DNS poisoning attack against the user	The solution as we described before does not mitigate this problem. What could be done extra is to configure a firewall on the guest machines to allow only access to the IP's corresponding to the online bank websites.
The attacker could replace files on the host system with some that would compromise the VMs, for example, replace of some of the VirtualBox's files with some changed by the attacker.	This would break the assumption that the host machine is reliable. The solution does not mitigate any attacker that can access locally or by network on the system and change files.
Physical attacks such as clone the disk so that the attack can try to break the VMs encrypted file and after replace the legitimate one with the one changed by the attacker.	This solution does not mitigate physical attacks.

Table 5.3: *Resume of the security problems and how the purposed solution deals with them*

From this analysis, there are only few attacks this solution does not mitigate. Those attacks are possible mainly because we are dependent of the host's system security. This is the main disadvantage of this solution. The advantage is that the user does not have to restart its machine to use it and therefore it is more users friendly.

5.2 The Read-Only Bootable Media Solution

Another security method that allows secure online backing or e-commerce is to use bootable read-only media. This can be by using a CD/DVD or USB Flash, although the currents USB Flash do not allow to protect against write as the old version did and so, they should not be used. It should only be used read-only media that cannot be changed after creation.

As we have done for the "Three color solution", this bootable system should be created and configured with only the services and applications required to perform the necessary operations. It is possible to create a LiveCD using the SUSE Studio on the option Build – Create appliance. LiveCD is a CD or DVD with a bootable operating system, normally Linux. One important configuration is not allowing this bootable media to access the local hard drive. Since this is a bootable ready-only medium, it would be necessary to restart the machine and have the option "Boot from CD" activated and configured to run before the

booting from the hard disk. This way, any malware on the local machine would not affect the user when using the bootable media. However, this LiveCD can have vulnerabilities, which can compromise the computer each time it is bootable, as in the AVI model [90]. For instance, if the LiveCD has a vulnerability that an attack can explore, then an Intrusion may happen, compromising the system. It is important to generate a new LiveCD from time to time (for instance, every 2 weeks). SUSE Studio offers a robust update process because if the user generates a new LiveCD with a different version, it will use the most updated packages and so there is the trust that the LiveCD built is updated.

Another way could be having a master installation created inside a VM. This VM would be handled secure and only used to update the master installation. Each week the user would run this VM and would update it using the available updating mechanisms on that distribution. If there were any updates applied, then the user should create a new LiveCD from it. If not, it means the LiveCD in use is still updated. This VM should be securely stored when not in use, like in a safe external disk.

In table 5.4, we have summarized the security problems an user can face and how this solution mitigates them.

Security Problem	How to mitigate
If the browser have the bank account login memorized, anyone who have access to the computer could start the VM and access the bank account	Since this runs a LiveCD, each time the user restart the computer it will loose any memorized credentials.
The user received an email with a link that goes to a fake site operated by criminals.	Our solution does not protect the activities the user would do in its host machine, but since the user has a Firewall, what he could do is to configure a rule in his host that would block access to the legitimate bank website. This way, he would know that if he pressed a link and that link would show the webpage of his bank it would be a fake one.
An attacker access the user's computer and with the credentials of that user, replace the LiveCD with one compromised.	This solution does not mitigate physical attacks.

The user installs a malware or keylogger on the LiveCD	The LiveCD are secure against this kind of attacks because the user does not have permission to install programs as root although there are keylogger for Linux that can run in user mode. However, this means the user would have to download it and execute it. Nevertheless, the LiveCD should be used only to access the online banking and e-commerce websites.
The user installs a malware or keylogger on the host	This solution is host system independent, therefore any virus installed on the host machine will not affect the LiveCD.
An attacker could do a DNS poisoning attack against the DNS server used by the user.	The solution does not mitigate this problem.
The attacker could explore a vulnerablity of the browser on the LiveCD.	The LiveCD should be updated (creating new LiveCDs) and the user should not go to other webpages beside the ones for online banking or e-commerce.
Physical attacks such as replace the LiveCD with one compromised.	This solution does not mitigate physical attacks.

Table 5.4: *Resume of the security problems and how the purposed solution deals with them*

This solution has some advantages, comparing with the previous one. The most important is that it does not rely on the security of the host machine. Even if the computer is full of virus and malwares, by the fact that we are rebooting and running it with a read-only bootable medium protects the user from this security problems.

The disadvantages is that managing the master installation and create/distribute the CDs has an administrative overhead and a cost. Other disadvantage is the fact it requires the user to reboot the computer and have some specific configuration selected on the BIOS. However, these disadvantages are small comparing with the security advantages. Perhaps the most problematic disadvantages of using this solution is if there is no DHCP Server, although the workaround for this is the user to take note of the network configuration used by the host operating system and then configure the LiveCD with the same network configuration.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis is a result of study work about virtual machines, their principal characteristics and differences, and the security impact of their utilization.

We initially studied the history of the virtual machines, following their evolution from the origins, in 1960s until nowadays with its implementation in the x86 architecture. We also presented the different components of virtualization, focusing on the problem of the x86 architecture that natively did not support virtualization and how some virtualization software companies worked around this problem. We have made a study about the current state of art of the main server virtualization products, describing their principal characteristics and systems supported.

The use of virtualization brings many advantages, such as reduction of the hardware resources needed with direct impact on cost efficiency, but also security advantages. The latter benefit is commonly used to spread the word on virtualization, but we wanted to demystify this myth, presenting some of the security problems that server virtualization brought and their impact. We choose two of the main server virtualization products commonly used by companies and universities and conducted a vulnerability analysis, using as reference the CVEs reported. The conclusion from that analysis is that both products show a similar security risk, and they require an extra attention to have their security patches applied.

Desktop security always brought some concerns to IT administrators and also end users, which had their computers infected by virus, worms, or their credentials stolen by some keylogger or phishing attack. We have described some solutions to this problem, using

virtualization. The first solution presented is based on type 2 hypervisor. The user creates and customizes three different virtual machines using SUSE Studio, following some steps described. At the end, the user will have three different VMs, which can use for critical operation that requires isolation but also secure environments. For this, the user will use one of the VM which we designated as Red VM that will be used for online banking only. For operations such as e-Commerce or consulting stock market websites, it will be used a Yellow VM. A third VM is created with the purpose to be used for operations considered unsafe such as webmail access (e.g. yahoo, hotmail) or online games. Taking advantage of using virtualization, we have the guarantee of isolation and therefore, each VM will not affect the others, neither the host machine, and this way, we have a private isolate environment that we will use. A functionality available on the hypervisor software we will use is the possibility to create snapshots of the virtual machines. This allows the user to have a clean environment each time it starts the VM. The second solution requires the user to reboot its machine and run a read-only bootable media. In this scenario, virtualization would be used to maintain a master version of the bootable media installed within a virtual machine, which allows to be updated, and this way create new versions of the read-only bootable media with the last updates applied. It was done a security analysis, describing how the solution can mitigate some of the security problems faced by a user.

I believe that, in a near future, virtualization will start being regarded as a desktop security-enabling technique rather than just a server workload consolidation mechanism, as it is by most of the IT community presently. Perhaps, with the commoditization of cloud computing, we will only have on our desktop computer a virtualization layer that would interact with our desktop hardware and a desktop VM that would be on some cloud computing. The future will tell us how right can be this idea.

6.2 Future Work

6.2.1 Virtual Machine Security

One of our goals was to do a security analysis of VMware ESXi using a tool called PREDATOR [91]. VMware ESXi provides a management console accessed by the VMware vSphere Client using HTTPS. We have eavesdropped on the communication and did some analysis of the traffic exchanged between the client and the server, and we detected that the structure passed is XML. The Listing 6.1 show the initial message exchanged between the VMware vSphere Client and VMware ESXi.

```
Method: GET
URL:    /client/clients.xml
status: 200
MIME type       XML
Request:
"GET /client/clients.xml HTTP/1.1
User-Agent: VMware VI Client/4.0.0
Host: cmu-pc157
Connection: Keep-Alive

Response:
"HTTP/1.1 200 OK
Date: Tue, 21 Jul 2009 19:39:01 GMT
Content-Type: text/xml
Content-Length: 315

<ConfigRoot>
  <clientConnection id=""0000"">
    <authdPort>902</authdPort>
    <version>4</version>
    <exactVersion>4.0.0</exactVersion>
    <patchVersion>1.0.0</patchVersion>
    <apiVersion>4.0</apiVersion>
    <downloadUrl>https://*/client/VMware-viclient.exe</
        downloadUrl>
  </clientConnection>
</ConfigRoot>"
```

Listing 6.1: *Message exchanged between the VMware ESX and vSphere Client*

Our goal was to identify the type of communication used and use PREDATOR in order to find vulnerabilities on it. However, this tool does not have the HTTPS inspection implemented and therefore was not possible to continue with our experience.

6.2.2 The Virtual Machine Read-Only Bootable Media Solution

The solution presented on 5.2 has the disadvantage of replace the CD each time an update is necessary. If we could somehow make this update automatically without the replace of

the read-only medium, it would make the solution almost perfect. The solution that we are going to describe is called Virtual Machine Read-Only Bootable Media solution and relies on a VM to boot an update system.

This solution would boot from a LiveCD and the first thing to run would be a Hypervisor, like Xen, that would then start its own Dom0. This Dom0 would connect to a webserver, using HTTPS and would compare the version of the guest VM file available on that webserver with the one it has on the LiveCD. If it is the same or older it would launch the one it has on the LiveCD, otherwise it would download the new version of the guest VM, it would validate its integrity, and then it would launch that VM. This file is compressed and signed by using the Kr of the webserver/company. After download, Dom0 has the Ku correspondent and so can validate its integrity.

This solution could be used for online banking and this LiveCD could be distributed by the banks. In Portugal, the majority of the banks use a matrix card to identify the user asking some random data on that matrix. Instead of printing that matrix on plastic cards, the bank could use a CD Business Card as the medium for the LiveCD.

The advantage of this solution, as mentioned before is that does not require an update to the medium CD each time there is new released of the VM. However, the download file must be small in order to take few seconds to download and does not take many bandwidth on the bank network. There are some Linux version like TinyCore [92] and KolibriOS [93] that can provide a small distribution that could have less than 10MB.

An issue regarding this solution could be the download size of the guest VM. Another problem of this solution is the same of the previous one, because is dependent of a DHCP server, although there is the workaround of manual network configuration.

Bibliography

[1] Joanna Rutkowska. IsGameOver(), Anyone? Black Hat Conference, 2007. Retrieved July 26, 2009.

[2] Randy Perry Al Gillen, Tim Grieser. Business value of virtualization: Realizing the benefits of integrated solutions. Technical report, IDC, July 2008.

[3] OpenDNS. Phishtank, June 2009. URL `http://www.phishtank.com/stats/2009/05/`. Retrieved November 10, 2009.

[4] Robert P. Goldberg. Survey of virtual machine research. *Computer*, pages 34–45, 1974.

[5] Christopher Strachey. Time sharing in large fast computers. In *International Conference on Information Processing*, pages 336–341. UNESCO, June 1959.

[6] John McCarthy. Reminiscences on the history of time-sharing. volume 14, pages 19–24, Piscataway, NJ, USA, 1992. IEEE Educational Activities Department.

[7] J. Howlett. The atlas computer laboratory. *Annals of the History of Computing, IEEE*, 21(1):17–23, Jan-Mar 1999. ISSN 1058-6180.

[8] Derrick Morris, Frank H. Sumner, and Michael T. Wyld. An appraisal of the atlas supervisor. In *Proceedings of the 1967 22nd national conference*, pages 67–75, New York, NY, USA, 1967. ACM.

[9] Barbara S. Brawn, Frances G. Gustavson, and Efrem S. Mankin. Sorting in a paging environment. *Commun. ACM*, 13(8):483–494, 1970. ISSN 0001-0782.

[10] Peter J. Denning. Performance evaluation: Experimental computer science at its best. In *SIGMETRICS '81: Proceedings of the 1981 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, pages 106–109, New York, NY, USA, 1981. ACM Press. ISBN 0897910516.

[11] Stuart E. Madnick and John J. Donovan. Application and analysis of the virtual machine approach to information system security and isolation. In *Proceedings of the workshop on virtual computer systems*, pages 210–224, New York, NY, USA, 1973. ACM.

[12] VMware, Inc. Vmware milestones, 2009. URL `http://www.vmware.com/company/mediaresource/milestones.html`. Retrieved November 22, 2009.

[13] Amit Singh. An introduction to virtualization. ISSN `http://www.kernelthread.com/publications/virtualization/`. URL `http://www.kernelthread.com/publications/virtualization/`.

[14] Nadir Kiyanclar. A survey of virtualization techniques focusing on secure on-demand cluster computing. *ArXiv Computer Science e-prints*, November 2005. Provided by the SAO/NASA Astrophysics Data System.

[15] Gerald J. Popek and Robert P. Goldberg. Formal requirements for virtualizable third generation architectures. *Commun. ACM*, 17(7):412–421, 1974. ISSN 0001-0782.

[16] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[17] Jim Hurst. Operating system protection and rings, February 2007. URL `http://www.giac.org/resources/whitepaper/architecture/92.php`. Retrieved November 06, 2009.

[18] Jim Smith and Ravi Nair. *Virtual Machines: Versatile Platforms for Systems and Processes (The Morgan Kaufmann Series in Computer Architecture and Design)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 1558609105.

[19] Mendel Rosenblum and Tal Garfinkel. Virtual machine monitors: Current technology and future trends. *Computer*, 38(5):39–47, 2005. ISSN 0018-9162.

[20] Richard L. Sites, Anton Chernoff, Matthew B. Kirk, Maurice P. Marks, and Scott G. Robinson. Binary translation. *Commun. ACM*, 36(2):69–81, 1993. ISSN 0001-0782.

[21] Kristy Andrews and Duane Sand. Migrating a cisc computer family onto risc via object code translation. In *ASPLOS-V: Proceedings of the fifth international conference on Architectural support for programming languages and operating systems*, pages 213–222, New York, NY, USA, 1992. ACM. ISBN 0-89791-534-8.

[22] Rich Uhlig, Gil Neiger, Dion Rodgers, Amy L. Santoni, Fernando C.M. Martins, Andrew V. Anderson, Steven M. Bennett, Alain K?gi, Felix H. Leung, and Larry Smith. Intel virtualization technology. *Computer*, 38(5):48–56, 2005. ISSN 0018-9162.

[23] AMD. *AMD64 Architecture Programmer's Manual Volume 2: System Programming*. Number 24593. September 2007. URL `http://www.amd.com/us--en/assets/content_type/white_papers_and_tech_docs/24593.pdf`. Retrieved July 23, 2009.

[24] Carl A. Waldspurger. Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36(SI):181–194, 2002. ISSN 0163-5980.

[25] VMware, Inc. *Resource Management Guide Update 2 and later for ESX Server 3.5, ESX Server 3i version 3.5, VirtualCenter 2.5*. VMware, Inc., 2009.

[26] G. Milos, D. G. Murray, S. Hand, and M. Fetterman. Satori: Enlightened Page Sharing. In *Usenix*, 2009.

[27] Oracle Corporation. Partitioning. Technical report, Oracle Corporation, 2002. URL `http://www.oracle.com/corporate/pricing/partitioning.pdf`. Retrieved November 15, 2009.

[28] John Scott Robin and Cynthia E. Irvine. Analysis of the intel pentium's ability to support a secure virtual machine monitor. In *SSYM'00: Proceedings of the 9th conference on USENIX Security Symposium*, pages 10–10, Berkeley, CA, USA, 2000. USENIX Association.

[29] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. In *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 2–13, New York, NY, USA, 2006. ACM. ISBN 1-59593-451-0.

[30] Adam Lackorzynski Björn Döbel Alexander Böttcher Hermann Härtig, Michael Roitzsch. L4 - virtualization and beyond. Korean Information Science Society Review, 2008.

[31] Jenni Susan Reuben. A survey on virtual machine security. In Jukka Manner and Laura Takkinen, editors, *Security of the End Hosts on the Internet, Seminar on Network Security Autumn 2007*. Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, 2007.

[32] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. John Wiley & Sons, Inc., New York, NY, USA, 2001. ISBN 0471417432.

[33] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis. Detecting targeted attacks using shadow honeypots. In *SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium*, pages 9–9, Berkeley, CA, USA, 2005. USENIX Association.

[34] Xuxian Jiang and Dongyan Xu. Collapsar: a VM-based architecture for network attack detention center. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 2–2, Berkeley, CA, USA, 2004. USENIX Association.

[35] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. ISBN 0321108957.

[36] Kenichi Kourai and Shigeru Chiba. HyperSpector: virtual distributed monitoring environments for secure intrusion detection. In *VEE '05: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, pages 197–207, New York, NY, USA, 2005. ACM. ISBN 1-59593-047-7.

[37] Ashlesha Joshi, Samuel T. King, George W. Dunlap, and Peter M. Chen. Detecting past and present intrusions through vulnerability-specific predicates. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 91–104, New York, NY, USA, 2005. ACM. ISBN 1-59593-079-5.

[38] Tal Garfinkel and Mendel Rosenblum. A virtual machine introspection based architecture for intrusion detection. In *In Proc. Network and Distributed Systems Security Symposium*, pages 191–206, 2003.

[39] Lionel Litty, H. Andrés Lagar-Cavilla, and David Lie. Hypervisor support for identifying covertly executing binaries. In *SS'08: Proceedings of the 17th conference on Security symposium*, pages 243–258, Berkeley, CA, USA, 2008. USENIX Association.

[40] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction. In *CCS '07: Proceedings*

of the 14th ACM conference on Computer and communications security, pages 128–138, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-703-2.

[41] Thomas J. Bittman. Virtualization with VMware or Hyper-V: What You Need To Know. Gartner Webinar, August 2009.

[42] Scott W. Devine, Edouard Bugnion, and Mendel Rosenblum. Virtualization system including a virtual machine monitor for a computer with a segmented architecture, February 2002.

[43] Eric Siebert. A brief history of VMware. IT Knowledge Exchange, February 2009. URL `http://itknowledgeexchange.techtarget.com/virtualization-pro/a-brief-history-of-vmware-2/`. Retrieved August 6, 2009.

[44] Edward Haletky. *VMware ESX server in the enterprise: designing and securing virtualization servers*. Pearson Education, Inc., 1st ed. edition, 2008.

[45] VMware, Inc. Best Practices for VMware ESX Server 3, June 2006. URL `www.vmware.com/pdf/esx3_best_practices.pdf`. Retrieved September 01, 2009.

[46] VMware, Inc. Details of What's New and Improved in VMware Infrastructure 3 version 3.5. URL `www.vmware.com/support/vi3/doc/whatsnew_esx35_vc25.html`. Retrieved September 01, 2009.

[47] VMware, Inc. Configuration Maximums - VMware Infrastructure 3, January 2009. URL `http://www.vmware.com/pdf/vi3_301_201_config_max.pdf`. Retrieved September 09, 2009.

[48] Scott Davis. VMFS vs. NFS for VMware Infrastructure?, September 2008. URL `http://blogs.vmware.com/storage/2008/09/vmfs-vs-nfs-for.html`. Retrieved November 17, 2009.

[49] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM. ISBN 1-58113-757-5.

[50] Keir A. Fraser, C Keir A. Fraser, Steven M. H, Steven M. H, Timothy L. Harris, Timothy L. Harris, Ian M. Leslie, Ian M. Leslie, Ian A. Pratt, Ian A. Pratt, K A Fraser,

T L Harris, I M Leslie, and I A Pratt. The xenoserver computing infrastructure. Technical report, 2003.

[51] Citrix Systems. Citrix Completes Acquisition of XenSource, October 2007. URL http://www.citrix.com/English/NE/news/news.asp?newsID=683171. Retrieved August 31, 2009.

[52] Allen Riddell. Gnu grub - gnu project - free software foundation (fsf), 2008. URL http://www.gnu.org/software/grub. Retrieved October 26, 2009.

[53] Anthony N. Liguori Ryan A. Harper, Michael D. Day. Using kvm to run xen guests without xen. In *2007 Ottawa Linux Symsposium*, pages 179–188, June 2007.

[54] Dan Duchamp and Greg De Angelis. A hypervisor based security testbed. In *DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, pages 3–3, Berkeley, CA, USA, 2007. USENIX Association.

[55] M. Tim Jones. Discover the linux kernel virtual machine, April 2007. URL http://www.ibm.com/developerworks/linux/library/l-linux-kvm/. Retrieved August 6, 2009.

[56] KVM. Guest support status. URL http://www.linux-kvm.org/page/Guest_Support_Status. Retrieved September 04, 2009.

[57] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, page 41, Berkeley, CA, USA, 2005. USENIX Association.

[58] Daniel Bartholomew. Qemu: a multihost, multitarget emulator. *Linux J.*, 2006(145): 3, 2006. ISSN 1075-3583.

[59] Sun Microsystems, Inc. Sun Welcomes Innotek, 2008. URL http://www.sun.com/software/innotek/. Retrieved October 30, 2009.

[60] Neil MacDonald. Security considerations and best practices for securing virtual machines. Gartner, Inc., March 2007.

[61] Joel Kirch. *Virtual Machine Security Guidelines Version 1.0*. The Center for Internet Security, September 2007.

[62] The MITRE Corporation. Common Vulnerabilities and Exposures List. URL http://cve.mitre.org/cve/. Retrieved July 07, 2009.

[63] Sean Whalen. An Introduction to Arp Spoofing, April 2001.

[64] VMware Communities. Esx 3.0.1 : protect from mitm attack, June 2007. URL http://communities.vmware.com/message/683223. Retrieved November 18, 2009.

[65] VMware Security Advisory. VMSA-2009-0006, April 2009. URL http://www.vmware.com/security/advisories/VMSA-2009-0006.html. Retrieved August 06, 2009.

[66] LXLabs. Hypervm, 2009. URL http://www.lxlabs.com/software/hypervm/. Retrieved August 06, 2009.

[67] Dan Goodin. Webhost hack wipes out data for 100,000 sites, June 2009. URL http://www.theregister.co.uk/2009/06/08/webhost_attack/. Retrieved August 06, 2009.

[68] Intel Corporation. *Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide, Part 2*. Intel Corporation, March 2009. URL http://download.intel.com/design/processor/manuals/253669.pdf.

[69] Alan Zeichick. Processor-Based Virtualization, AMD64 Style, Part II. June 2006. URL http://developer.amd.com/documentation/articles/pages/630200615.aspx. Retrieved July 24, 2009.

[70] Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, and Jacob R. Lorch. Subvirt: Implementing malware with virtual machines. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 314–327, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2574-1.

[71] Tal Garfinkel, Keith Adams, Andrew Warfield, and Jason Franklin. Compatibility is not transparency: Vmm detection myths and realities. In *In: Proceedings of the 11th Workshop on Hot Topics in Operating Systems (HotOS-XI*, May 2007.

[72] Hagen Fritsch. Analysis and detection of virtualization-based rootkits. Master's thesis, Technische Universitat Munchen, 2008.

[73] Martim Carbone, Diego Zamboni, and Wenke Lee. Taming virtualization. *IEEE Security and Privacy*, 6(1):65–67, 2008. ISSN 1540-7993.

[74] Lionel Litty and David Lie. Manitou: a layer-below approach to fighting malware. In *ASID '06: Proceedings of the 1st workshop on Architectural and system support for improving software dependability*, pages 6–11, New York, NY, USA, 2006. ACM. ISBN 1-59593-576-2.

[75] Tal Garfinkel and Mendel Rosenblum. When virtual is harder than real: security challenges in virtual machine based computing environments. In *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*, pages 20–20, Berkeley, CA, USA, 2005. USENIX Association.

[76] Neil Haller. The S/KEY One-Time Password System. In *In Proceedings of the Internet Society Symposium on Network and Distributed Systems*, pages 151–157, 1994.

[77] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, pages 427–436, Oct 1992.

[78] U. Fiege, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 210–217, New York, NY, USA, 1987. ACM. ISBN 0-89791-221-7.

[79] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4 (3):161–174, 1991.

[80] S. M. Bellovin. Security problems in the tcp/ip protocol suite. *SIGCOMM Comput. Commun. Rev.*, 19(2):32–48, 1989. ISSN 0146-4833.

[81] Ron Oglesby and Dan Pianfetti. Patch Tuesday for VMware, 12 2007. URL `http://www.virtualization.info/2007/12/patch-tuesday-for-vmware.html`. Retrieved November 22, 2009.

[82] National Institute of Standards and Technology. National vulnerability database. URL `http://nvd.nist.gov/`. Retrieved August 06, 2009.

[83] National Vulnerability Database (NVD). Draft cvss v2.10 equations, March 2007. URL `http://nvd.nist.gov/cvsseq2.htm`. Retrieved August 6, 2009.

[84] Rob Rachwald. Is banking online safer than banking on the corner? *Computer Fraud & Security*, 2008(3):11 – 12, 2008. ISSN 1361-3723. URL http://www.sciencedirect.com/science/article/B6VNT-4S3H495-H/2/237237fd0b772f1eb9ee5ffacba73a3a.

[85] Novell, Inc. SUSE Studio, 2009. URL http://susestudio.com/. Retrieved October 31, 2009.

[86] Sun Microsystems, Inc. *Sun VirtualBox User Manual*, version 3.0.10 edition, 2009. URL http://www.virtualbox.org. Retrieved October 31, 2009.

[87] Igor Pavlov. 7-Zip, 2009. URL http://www.7-zip.org/. Retrieved November 03, 2009.

[88] Dan Wendlandt and Ethan Jackson. Perspectives : Improving SSH-style Host Authentication with Multi-path Network Probing, 2009. URL http://www.cs.cmu.edu/~perspectives/firefox.html. Retrieved November 21, 2009.

[89] Thomas Mendel. Enterprise desktop and web 2.0/saas platform trends. Technical report, Forrester Research, Inc., March 2008.

[90] Paulo Verissimo and Luis Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.

[91] João Antunes, Nuno Ferreira Neves, and Paulo Jorge Veríssimo. Detection and prediction of resource-exhaustion vulnerabilities. In *ISSRE '08: Proceedings of the 2008 19th International Symposium on Software Reliability Engineering*, pages 87–96, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3405-3.

[92] Robert Shingledecker. Tiny core linux, 2009. URL http://www.tinycorelinux.com/. Retrieved November 04, 2009.

[93] KolibriOS Project Team. Kolibrios, 2009. URL http://www.kolibrios.org/. Retrieved November 04, 2009.