

Towards Using a Trusted Computing Base for Security in Virtual Machines

Belinda Marie Deci
E00818003
IA 479, Winter 2008
Final Project Proposal
April 14, 2008

Table of Contents:

1. ABSTRACT	3
2. INTRODUCTION	4
3. BACKGROUND AND RELATED WORKS	5
3.1 VIRTUALIZATION METHODOLOGIES.....	5
3.2 VIRTUAL MACHINE MONITORS (HYPERVISORS).....	6
3.3 HARDWARE-ASSISTED VIRTUALIZATION.....	7
3.4 THE BENEFITS AND DRAWBACKS OF VIRTUALIZATION	8
3.5 TRUSTED COMPUTING AND VIRTUALIZATION.....	9
4. CONCLUSION AND FUTURE WORK.....	11
Works Cited.....	13

1. ABSTRACT

This proposal discusses the various virtualization methodologies that can be employed, the components of the virtualization architecture (including hardware-assisted virtualization), the benefits and drawbacks of virtualization, and Trusted Computing's relationship to virtualization solutions. It concludes with an analysis of security concerns in Virtual Machines and a case for a Trusted Computing Base in virtualized environments.

A growing number of organizations and individuals have chosen to deploy virtualization in their computing environments. The particular reasons for using virtual machines are many and varied, but there are some basic goals for using virtualization technology: increased security; higher productivity; use of a wider variety of applications and platforms; ease in testing and deployment of security solutions, new software implementations; and backup and recovery plans. The increase in use of virtualization has led to the development of new hardware technology, such as the production of Intel and AMD chips with on-board Hardware-assisted virtualization procedures. It has also led to growing concerns for the security of these virtualized machines and environments. This proposal lays out a plan to address these security concerns, including analyses and tests of various attack scenarios on Virtual Machines and environments and a rigorous analysis of implementation of a Trusted Computing Base within a machine utilizing various virtualization scenarios.

2. INTRODUCTION

In recent years, many organizations and individuals have begun taking advantage of and utilizing Virtual Machines (VMs) within their computing environments. Operating System virtualization, especially server virtualization, within an organization's Information Technology infrastructure can bring numerous benefits to the establishment's computing functions, including efficient and reliable backup and recovery, a safe environment for testing and deployment of new applications and systems, an ability to leverage the benefits of various operating systems simultaneously, and the creation of security boundaries within the network and within the machine itself. [7]

This has led to a flurry of activity in the development of virtualization technologies. Chip manufacturers have responded to the increased demand by producing CPUs with virtualization technologies built-in. Scores of Virtual Machine Monitors (VMMs), have been written and made available to the public and to organizations. Also known as hypervisors, these are a layer of software that controls the interactions between the virtual operating system (or "guest operating system") and the hardware.

Many concerns for the security of the Virtual Machines -the physical machines themselves, and the networks in which they participate- have accompanied this increase in the use of virtualization technology. These concerns stem mainly from the way a VMM intercepts processor calls by the guest operating system (paravirtualization), and from the way the new virtualization-capable chips (also known as hardware-assisted virtualization) define the authority levels of the VMM, the host operating system, and the guest operating system. It is generally claimed that Virtual Machines offer added security to the physical machine because the hypervisor defines new security boundaries through which all of the guest operating system's activities must go. It is yet unclear that this is the case, especially in light of the added layer of software involved in the operation of a hypervisor, and of the redefinition of kernel privilege level by hardware-assisted virtualization. [7]

This proposal seeks to further explore the determinate security of Virtual Machines, Virtual Machine Monitors, and hardware-assisted virtualization, with an analysis of VM security boundaries, VMM privilege level assignments, software security and the addition or expansion of a Trusted Computing Base (TCB) within the machines and the Virtual Machines. It will begin with an explanation of the background of and the technologies used in operating system virtualization, the benefits and concerns associated with virtualization, how Virtual Machine Monitors operate, how hardware-assisted virtualization works, and what is meant by a Trusted Computing Base (TCB.) It will outline past and ongoing research into the area of virtualization security, and make a case that signed code as part of the TCB of the VM is integral to the security of the machine. Finally, this proposal will outline the future direction of the research into securing hypervisors and Virtual Machines with signed and secure code on all levels, as part of the physical machine's TCB.

3. BACKGROUND AND RELATED WORKS

3.1 VIRTUALIZATION METHODOLOGIES

Though the term *virtualization* has been around since the 1960's, it has recently gained a new level of popularity, with the increased use of Virtual Machines to simulate one or many different operating systems on a single physical machine. Often called platform virtualization, this process is performed on the physical machine by software, often a Virtual Machine Monitor (VMM) or hypervisor, which simulates most of, or the entire, computing environment to the layer above it, a Virtual Machine running a guest operating system. The Virtual Machine and the guest operating system see only the environment (including hardware resources) shown to it by the VMM, and therefore think and behave as if they are an actual physical machine.

There are several different types of virtualization that can be employed, depending upon the hardware and memory capabilities of the machine. [6]

- *Emulation* simulates the entire hardware state of the machine.
- *Full Virtualization* simulates enough of that machine's hardware to allow a guest operating system to run unmodified in isolation; examples of this style of virtualization include Parallels Desktop for Mac and the popular VMWare Workstation and Server.
- *Partial Virtualization* simulates multiple instances of much of the underlying hardware, to allow for resource sharing and process isolation, but does not allow for separate 'guest' operating system instances. This is mainly used by the host operating system to increase efficiency.
- *Paravirtualization* is a technique in which the virtual machine hypervisor presents a software interface to the guest operating system that is similar to the underlying hardware. The guest operating systems must be modified (either by the proprietor or by the hypervisor) to run at a different authority level than normal. Examples of this include the Xen and Denali hypervisors.
- *Hardware-assisted Virtualization* (sometimes called hardware-enabled virtualization) is a method in which the instruction sets in virtualization-capable processors have been written to allow hypervisors to run in a higher privilege (or protection) ring than in previous CPUs. This allows for the guest operating systems to run at their normal authority level (kernel mode), so there is no need for them to be re-written, or their processor calls to be re-interpreted by the hypervisor.

3.2 VIRTUAL MACHINE MONITORS (HYPERVISORS)

In all of the virtualization methods described above, the process requires software for implementation. With the exception of partial virtualization, which is mainly a method employed by host operating systems to increase computational efficiency, each method creates and manages a virtual machine via a hypervisor. This is an extra added piece of software which runs between the hardware and the guest operating system, generally at the kernel level, giving it, as in the case of a host operating system, kernel access to the hardware. The insertion of this VMM software adds another layer of complexity to the machine and its operations. [8]

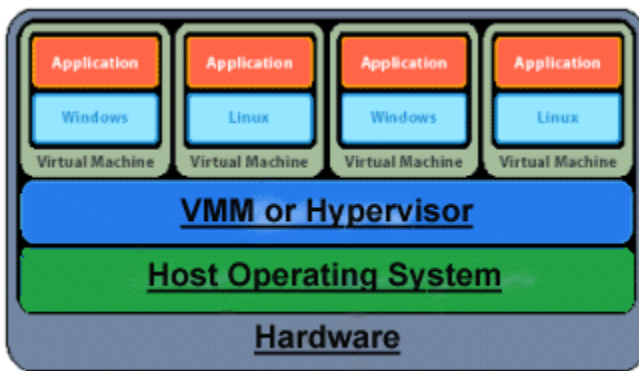


Figure 1: The Computing Stack with Virtualization
(Adapted from [14])

The hypervisor acts as a middle-man, so to speak, between the guest operating system and the hardware, functionally making all the memory and processing calls for the OS (often called *hypercalls*.) This separation of the guest OS and the hardware by the hypervisor is what constitutes the creation of a security boundary within the machine, and within the software.

There are a multitude of hypervisors, or Virtual Machine Monitors, available today, such as VMWare Server and Workstation, the Xen hypervisor (open-source), VirtualBox, Denali, VirtualPC and others. Most can operate using full or paravirtualization techniques, and can take advantage of hardware-assisted virtualization. What chiefly separates each of them is the host operating system that they run on top of, and the goals of virtualization that each addresses. Some hypervisors aim to increase security, some are intended to increase computational efficiency and decrease latency in the virtual machine, and still others operate to create and manage multiple different guest operating systems on a single platform.

3.3 HARDWARE-ASSISTED VIRTUALIZATION

To further enable efficiency in virtualization technologies, processor manufacturers, such as Intel and AMD, have recently begun making and marketing virtualization-enabled CPUs. These processors aid in the deployment of virtual machines and speed the functioning of the Virtual Machines by virtue of the fact that their instructions sets have been modified from the

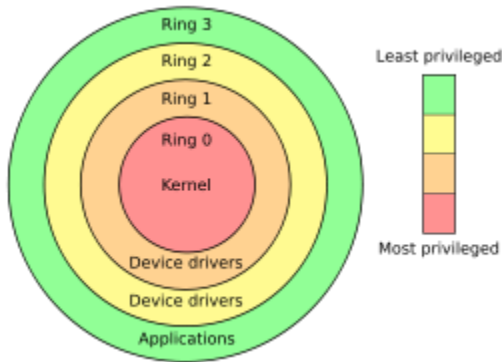


Figure 2: Protection Rings [10]

way that all previous x86 processors have been programmed.

Traditional x86 operating systems expect to have direct access to the CPU and other hardware, such as memory, known as kernel mode. This is typically described as running in Protection Ring 0. When an operating system is running in a Virtual Machine, and its processor and memory calls are being handled by the hypervisor, everything shifts out a Ring, with the hypervisor running in Ring 0, and the guest operating system in Ring 1 (user mode.)

Most operating systems are not written to run in Ring 1, and so must either be rewritten, or the hypervisor must handle the translation (called *trap and emulate*) of the guest operating system's calls to the hardware. This can become very code-intensive, and computationally-expensive for the hypervisor and the CPU. [2]

With hardware-assisted virtualization, the traditional x86 instruction set of the CPU has been modified, so that when a CPU becomes aware that a VMM is running just above it, the VMM runs in Ring -1, so that the guest operating system can run in Ring 0. Additionally, the chips have been modified to store OS and hardware state information in a protected memory space that only the VMM can access. This means that no privileged state data can be leaked to the host OS or guest OS. This speeds up the performance of the hypervisor, and of the Virtual Machine. [2]

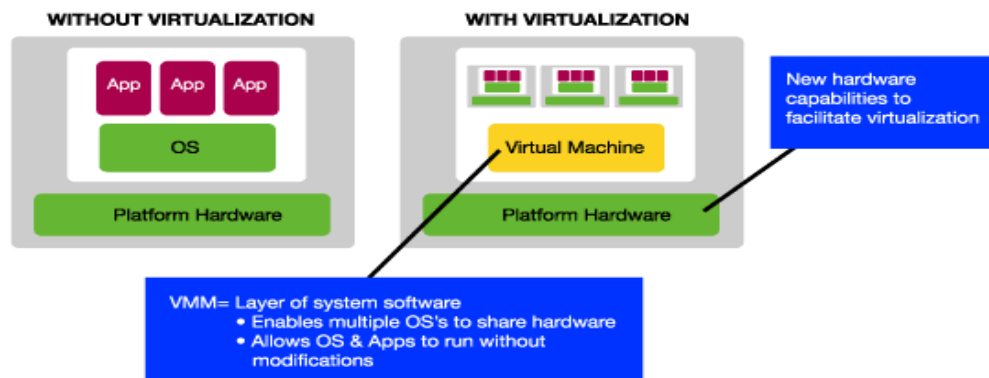


Figure 3: Hardware-Assisted Virtualization [15]

3.4 THE BENEFITS AND DRAWBACKS OF VIRTUALIZATION

Organizations and individuals choose to employ virtualization in their computing environments for a host of reasons, and the methods that they use to implement virtualization will, generally, support those reasons. Virtualization in a server environment can lower hardware costs for the organization, while at the same time speed backup and recovery processes. Deployment of virtual machines is very quick, especially when the organization uses a pre-configured virtual machine ‘template’ for its servers. Testing of new applications, software packages, security configurations and patches, to name only a few instances, can be done safely and securely on a virtual machine network, built to be an exact replica of the organization’s production environment, without having to endanger the actual network. [5]

Companies use virtualization to leverage the unique strengths and features of differing operating systems, and networks utilize virtualization to take advantage of the security boundaries that the technology creates. Academic institutions have created whole virtualized networks, to assist in the education of their server and operating systems configuration classes. [3]

However, virtualization comes at a cost, mostly in processor and memory access speed. Because the VMM must either trap-and-emulate guest operating system calls or the guest OS must be recompiled (if it has been rewritten) at run-time, this results in a noticeable performance hit. [1] Hardware-assisted virtualization technology was developed to solve just this problem, but may have resulted in the unintended side-effect of decreasing security in the machine. This is one of the trade-offs of increasing Virtual Machine performance: since the guest OS now runs in Ring 0 (kernel mode), even when in a Virtual Machine, its calls share the same privilege level as the host operating system, and, at times, the same memory space. Theoretically, this could lead to data leakage or transference of viruses, malware or infections between the two operating systems, in either direction.

This concern, then, leads us back to more traditional computer security concerns: can a compromised host operating system infect an otherwise stable guest operating machine running in a virtualized environment, due to the possibility of data leakage, or vice versa? Because organizations increasingly use virtualized servers in large and/or distributed networks, there is the possibility that a compromised virtual server can affect the entire network.

An example of the malicious use of an element of virtualization to gain control over machines and computing resources comes in the form of a new type of rootkits, called VM-based rootkits (VMBRs). These are stealthy, small, opaquely-installed malicious hypervisors, which can quickly insert themselves below the host operating system after a system reboot. They are almost completely undetectable, and the machine’s operating system does not know it is running in a virtualized environment. [4] The creation of these rootkits illustrates the return to ‘traditional’ computer security concerns, even with the separation and segmentation of resources that virtualization is intended to provide.

3.5 TRUSTED COMPUTING AND VIRTUALIZATION

These security concerns and performance issues naturally lead us to ask the question: How can we assure that the code that we are running from within the guest operating system and from the hypervisor is secure and doing what we intend for it to do? The answers can be found within the Trusted Computing domain.

With Trusted Computing, the computer will consistently behave in specific and predictable ways, and those behaviors will be enforced by the machine's hardware and, optimally, by its software. This begins by building the machine, from the hardware up, with trusted components, which are trusted because their development and life-cycle follow Trusted Computing standards.

The start-point for Trusted Computing hardware is a Trusted Platform Module. The TPM is a microcontroller that stores keys, passwords and digital certificates, which is affixed to the motherboard. It ensures that the information stored there is made more secure from external software attack and physical theft. Security processes, such as digital signature and key exchange, are protected. Critical applications and capabilities such as local protection of data are thereby made much more secure. TPM capabilities also can be integrated into other components in a system, such as virtualization and hypervisors. [9]

To achieve security within a machine running many virtualized machines can be regarded in the same way that security within a network is achieved: by securing the connections between the two end points (whether it is two different physical machines or a Virtual machine and the host machine.) Security threats have only increased as connectivity and network complexity have risen. To address this problem, the industry created a special interest group (SIG), called the Trusted Computing Group (TCG). The TCG's mandate was to specify a comprehensive approach to enterprise security based on compatible technology building blocks. [11] This approach has included standards and specifications for software as well as hardware, and includes Trusted Server specifications. These specifications can be easily and readily applied to software in a normal client computing stack, and to hypervisors, which will allow for a secure, reliable Trusted Computing Base for the physical machine and the Virtual Machine.

A trust chain in a machine is built from the hardware up, all the way through the software stack: it starts with *trustworthy* hardware components, which means that the behavior of the component (which includes security behavior) is demonstrably compliant with its stated functionality. To then build the trust chain, the hardware must find that the components in the next layer, the host operating system, can be *trusted*. In the Trusted Computing model, this trust relationship relies on secure code and code-signing: the operating system code has been written securely, which is verified by its compliance to Trusted Computing standards as well as by its behaviors' predictable activity. To supplement this, the user and then the hardware

verify trustworthiness by using code-signing, a hash or key which is verified at launch of every executable process or module of the software.

In a Virtual Machine environment, this Trusted Computing Base can be implemented quite easily, by simply beginning with trusted hardware, and then running signed and trusted operating systems and applications, from the boot loader to the host operating system, through the hypervisor and the guest operating system(s). If a machine is using a TPM, the user could theoretically bypass the step of verifying that the software he is installing is written securely, as the TPM will monitor that software's behavioral compliance, but since we are speaking of securing the software and building a trust chain, this is neither optimal nor recommended.

There are, however, a couple of obstacles to this method of building a Trusted Computing Base within a machine running Virtual Machines. First, there is the understandable performance degradation that comes when the hardware needs to verify that each piece of code is trusted, which, on top of the well-documented slow-downs that accompany use of Virtual Machines, can be significant. Second, the overwhelming majority of software available today is not written securely, and verified by code signing. Lastly, it has been shown that there are known attacks against TPM hardware that have successfully cleared the hash values stored, clearing the way for malicious code to write its own values to the TPM to fool it into believing 'everything's OK.' [12]

4. CONCLUSION AND FUTURE WORK

It is common thought today that, because a machine is running in a virtualized environment, it is, by default, more secure. After working our way through the logical examination of virtualization, how it is implemented and the components involved, it can be seen that there are several security areas that bear further analysis with regard to Virtual Machine Monitors and Virtual Machines. This analysis is especially vital today, as organizations deploy entire networks of virtualized servers, often running business- or mission-critical applications and services. Are these Virtual Machines really as secure as they are believed to be? Adding virtualization logically adds a layer of complexity to a machine and to a network, and conventional wisdom states that with complexity comes an opportunity for errors, holes, and malicious activity. In other words, the more parts there are, the more parts there are to break. This proposal intends to look for the answers to several security questions, concerning virtualization itself, hardware-assisted virtualization, and a Trusted Computing Base for virtualization technology.

The first is the question of the effect of ‘traditional’ security hazards on Virtual Machines and the networks in which they are active: how do viruses, worms, Trojan Horses, and other malware behave within a Virtual Machine? A future focus of this research will be a comprehensive testing of this question, from every direction: can a worm on a Virtual Machine get onto the host operating system, and from there, onto the network? Can a virus from the network get onto the Virtual Machine? These are important questions to ask and answer, in light of the recent rise in use of Virtual Machines as networked servers, in many cases as mission-critical end-points.

Another topic that must be explored is the security consequence of deploying virtualization with the assistance of hardware, in which the hypervisor, the host operating system and the guest operating system all enjoy escalated Protection Ring privileges. Future research will investigate the effect that this has on the security boundaries of the physical machine, and answer the question of if there can then be data leakage between the three entities mentioned above.

The largest matter to be addressed by the future work of this proposal will be to fully compare and document two Virtual Machine implementations: one built in the common method, and the other built with a full Trusted Computing Base and full trust chain. An analysis will be made of the differences in cost, ease-of-use (installation and management), performance, and adherence to the tenets of Information Assurance (Confidentiality, Integrity, Availability) and, additionally, Non-Repudiation.

Though our main focus for this research is a documentation of the security impact, we believe that to ignore such associated issues as performance slowdowns and latency would be imprudent. As organizations choose more and more to implement virtualization in their computing environments, performance and cost often play a large role in IT decision-making.

Research into the area of utilizing a full Trusted Computing Base in virtualization implementations has already begun. Hand, Murray and Milos at the University of Cambridge have investigated the use of TCB with the Xen hypervisor, with promising results. [13] This proposal seeks to further the exploration into this topic by expanding the testing to commodity and other software hypervisors and operating environments. As previously stated, organizations use a wide variety of virtualization technologies, computer architectures and operating system software to accomplish their virtualization goals: an analysis of the feasibility and result of using TCB on this variety of components, not just the Xen hypervisor, is warranted, and can be invaluable for an organization's IT decision-making process.

Works Cited:

- [1] Adams, K., & Aageson, O. (2006). A Comparison of Software and Hardware Techniques for x86 Virtualization. *ACM ASPLOS '06*. San Jose, CA: ACM.
- [2] Beal, V. ' . (2007, April 12). *Understanding Hardware-Assisted Virtualization*. Retrieved March 01, 2008, from www.webopedia.com/Did_You_Know?:
http://www.webopedia.com/DidYouKnow/Computer_Science/2007/hardware_assisted_virtualization.asp
- [3] Border, C. (2007). The Development and Deployment of a Multi-User, Remote Access Virtualization System for Networking, Security, and System Administration Classes. *ACM SIGCSE 2007* (pp. 576-580). Covington, KY: ACM.
- [4] Carbone, M., Zamboni, D., & Lee, W. (2008). Taming Virtualization. (S. O. Saydjari, Ed.) *IEEE Security & Privacy* , 6 (1), 65-67.
- [5] Collier, G., Plassman, D., & Pegah, M. (Fall 2007). Virtualization's Next Frontier: Security. *ACM SIGUCCS Conference on User Services*. 35, pp. 34-36. Lake Buena Vista, FL: ACM.
- [6] Crosby, S., & Brown, D. (2006-2007, December/January). The Virtualization Reality. *ACM Queue* , 34-41.
- [7] Farrell, S. (2008). Security Boundaries. *IEEE Internet Computing* , 12 (1), 93-96.
- [8] Fong, L., & Steinder, M. (2008). Duality of Virtualization: Simplification and Complexity. *ACM SIGOPS Operating Systems Review*. 42, pp. 96-97. New York, NY: ACM.
- [9] Group, T. C. (2008). *TPM:FAQs*. Retrieved March 12, 2008, from Trusted Computing Group: <https://www.trustedcomputinggroup.org/faq/TPMFAQ/>
- [10] Hertzprung. Privelege Rings Diagram. http://en.wikipedia.org/wiki/Image:Priv_rings.svg.
- [11] Kay, R. L. (2006). *How To Implement Trusted Computing*. Endpoint Technologies Associates.
- [12] Lawson, N. (2007, July 16). *TPM Hardware Attacks*. Retrieved March 30, 2008, from root labs rdist: <http://rdist.root.org/2007/07/16/tpm-hardware-attacks/>
- [13] Murray, D. G., Milos, G., & Hand, S. (2008). Improving Xen Security Through Disaggregation. *4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* (pp. 151-169). Seattle, WA: ACM.
- [14] Vi411. VMWare Server Virtualization. <http://www.vi411.org/wp-content/uploads/2006/07/gsx-1.gif>. Vi411.org.
- [15] XBitLabs. Intel's Virtualization Explained. <http://www.xbitlabs.com/images/news/2005-01/virtualization.gif>. XBitLabs.

