A Survey of Security Issues in Hardware Virtualization

GÁBOR PÉK, Budapest University of Technology and Economics and KÜRT Co.
LEVENTE BUTTYÁN, Budapest University of Technology and Economics.
BOLDIZSÁR BENCSÁTH, Budapest University of Technology and Economics.

Virtualization is a powerful technology for increasing the efficiency of computing services; however, besides its advantages, it also raises a number of security issues. In this article, we provide a thorough survey of those security issues in hardware virtualization. We focus on potential vulnerabilities and existing attacks on various virtualization platforms, but we also briefly sketch some possible countermeasures. To the best of our knowledge, this is the first survey of security issues in hardware virtualization with this level of details. Moreover, the adversary model and the structuring of the attack vectors are original contributions, never published before.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protection

General Terms: Security

Additional Key Words and Phrases: Hardware virtualization, operating systems, security, malware

ACM Reference Format:
Pék, G., Buttyán, L., and Bencsáth, B. 2013. A survey of security issues in hardware virtualization. ACM Comput. Surv. 45, 3, Article 40 (June 2013), 34 pages.
DOI: http://dx.doi.org/10.1145/2480741.2480757

1. INTRODUCTION

Virtualization is a powerful technology for increasing the efficiency of computing services provided to private and business users in terms of performance, maintenance, and cost. Essentially, virtualization provides an abstraction of physical hardware resources that allows for the operation of the same services on a multitude of different physical hardware platforms. By controlling access to the physical resources, virtualization can also be used to run different services in parallel on the same physical hardware. It allows, for example, for the execution of multiple operating systems simultaneously on the same physical host. In this way, resources can be utilized more efficiently, and users can decrease their expenditures on computing services significantly. For these very same reasons, virtualization plays a key role in *cloud computing*, which allows distinct customers to hire online computing resources for their own purposes. By doing so, specific applications (Software as a Service—SaaS), development platforms (Platform

This work is connected to the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME", and the "Security focused services and products in cloud infrastructure environment" project at KÜRT Co., supported by the New Széchenyi Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002 and GOP-1.3.1-09/B-2010-0007). L. Buttyán was partially funded by the Hungarian Academy of Sciences through the support of the Academic Research Group on Information Systems (ID: 04-130).
Authors' addresses: G. Pék, L. Buttyán, and B. Bencsáth, Laboratory of Cryptography and System Security, Budapest University of Technology and Economics, BME-HIT, PO Box 91, 1521 Budapest, Hungary; corresponding author's email: pek@crysys.hu.

DOI: http://dx.doi.org/10.1145/2480741.2480757

as a Service—PaaS), or complete virtual machines with networking components and storage capabilities (Infrastructure as a Service—IaaS) can be requested from the cloud operator on a pay-per-use basis.

While the advantages of virtualization are pretty much clear to everyone, one must also be aware of the fact that it gives rise to a number of security issues. Some of those issues exist in traditional computing environments as well, but they need to be addressed with special care in virtualized environments, while some other issues are specific to virtualization and hence require novel solutions. One such example is multitenancy, which allows for cross-platform information flow between customers hiring virtual machines over the same physical host [Ristenpart et al. 2009] in an IaaS cloud. Other issues entitle adversaries to execute arbitrary out-of-the-guest code either locally [Wojtczuk 2008b] or remotely [Shelton 2005] without owning the required access rights. Virtualized storage systems also have special security requirements [Dwivedi 2003, 2004, 2005] in order to keep data secure. Due to the increasing popularity of using virtualization technologies, it is important to discuss these security issues and hence raise the awareness of the potential users of virtualized services and infrastructures.

For this reason, in this article, our objective is to provide a thorough survey of security issues in hardware virtualization. We focus on potential vulnerabilities and existing attacks on hardware virtualization platforms, but we also briefly sketch some possible countermeasures. As it turns out, the number of reported vulnerabilities and attacks on different virtualization platforms is quite large, so we structure the presentation of those based on their target; hence, we introduce vulnerabilities and attacks targeting the guest, the host OS, the hypervisor layer, the management interfaces, and the different networks within a virtual infrastructure. The detailed discussion of vulnerabilities and attacks is preceded by the definition of an adversary model and a short overview on virtualization detection and identification techniques.

More specifically, the content of this survey is structured as follows: We first present a taxonomy of virtualization concepts in Section 2; this helps to understand the scope of our survey, which is focused on hardware virtualization. After that, in Section 3, we introduce an adversary model,where we distinguish different types of adversaries based on their available privileges and the targeted resources. Since, as a first step of an attack, the adversary needs to detect and identify the virtualized environment, we give a short overview on existing techniques for virtual machine detection and identification in Section 4. Then, we describe vulnerabilities and attacks aimed at compromising the guest, the host operating system, and the hypervisor in Sections 5, 6, and 7, respectively. In addition to studying the security issues at the main software layers, we also survey vulnerabilities and attacks on the management interfaces in Section 8, and on the various networks and storage services used in a typical virtual infrastructure in Section 9. Some miscellaneous threats related to virtualization are mentioned in Section 10. Finally, we provide a brief overview on possible countermeasures against the discussed attacks in Section 11 and conclude the paper in Section 12.

To the best of our knowledge, this is the first survey of security issues in hardware virtualization with this level of details. Moreover, the adversary model and the structuring of the attack vectors are original contributions, which have not been published before. We believe that they are sufficiently general to be used to classify not only existing but also future vulnerabilities and attacks in virtualized environments.

Note that the sources from which we compiled this survey are not limited to papers published in journals or conference proceedings, but we extensively relied upon the Common Vulnerabilities and Exposures (CVE) database hosted by MITRE (available online at `http://cve.mitre.org/`). References to CVE records do not appear in the reference list at the end of the article, but they can be easily resolved online.

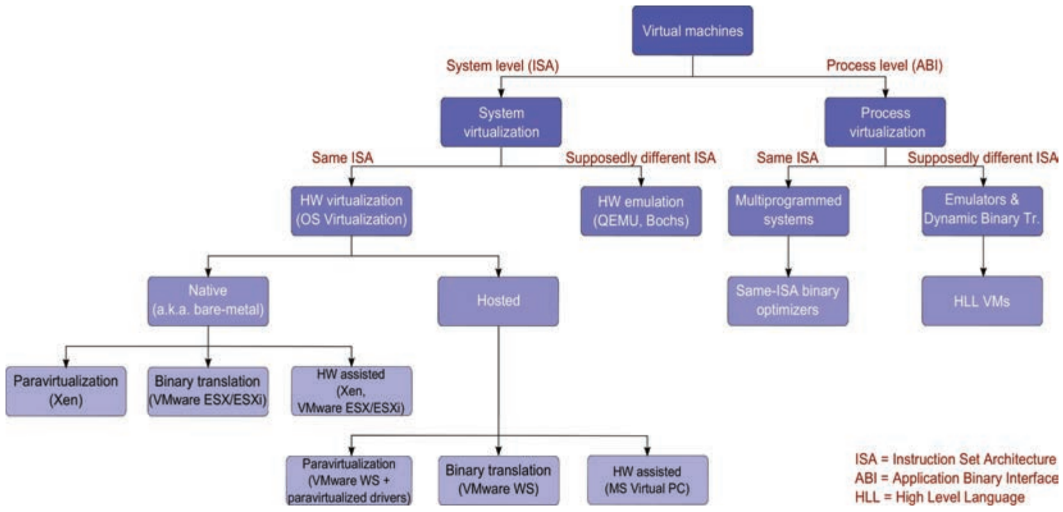

Virtual machines
System level (ISA)
Process level (ABI)
System virtualization
Process virtualization
Same ISA
Supposedly different ISA
Same ISA
Supposedly different ISA
HW virtualization (OS Virtualization)
HW emulation (QEMU, Bochs)
Multiprogrammed systems
Emulators & Dynamic Binary Tr.
Native (a.k.a. bare-metal)
Hosted
Same-ISA binary optimizers
HLL VMs
Paravirtualization (Xen)
Binary translation (VMware ESX/ESXi)
HW assisted (Xen, VMware ESX/ESXi)
Paravirtualization (VMware WS + paravirtualized drivers)
Binary translation (VMware WS)
HW assisted (MS Virtual PC)
ISA = Instruction Set Architecture
ABI = Application Binary Interface
HLL = High Level Language

Fig. 1. Taxonomy of virtualization concepts.

2. OVERVIEW OF VIRTUALIZATION CONCEPTS

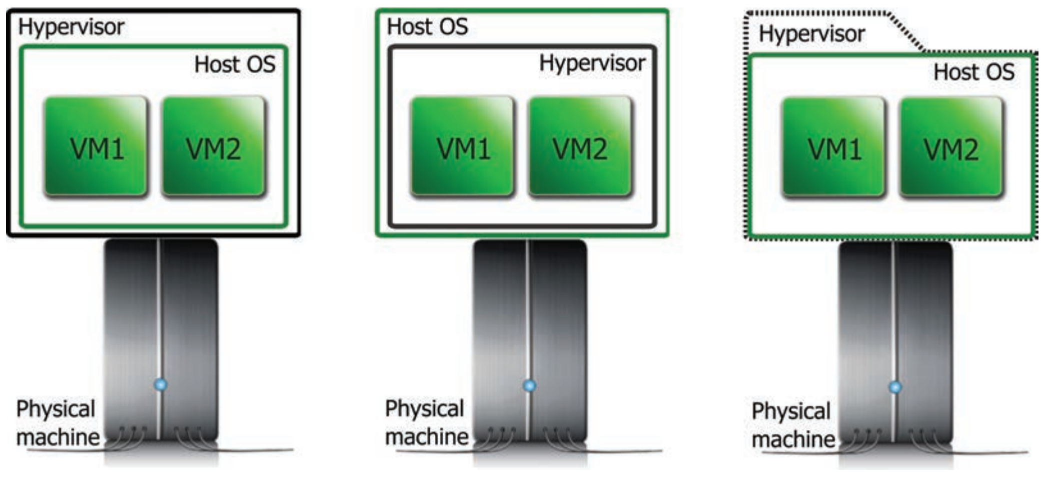
A virtual machine (VM) is an abstraction of computing resources presented to services to allow them to operate simultaneously on the same physical hardware infrastructure. VMs can be classified into two main categories: process and system virtual machines, as Figure 1 depicts. A process VM executes an individual process, and thus its lifetime is identical to the lifetime of the corresponding process. Examples for such VMs are found in well-known development platforms, such as Microsoft.NET and Java VM. Extending this concept, system virtualization corresponds to a VM that provides a complete operating system with multiple processes. The process or system being executed in a VM is called the *guest*, while the underlying platform that allows a VM to run is defined as the *host*. System virtualization provides a complex layer to services implementing both hardware virtualization[1] and hardware emulation. Hardware virtualization includes approaches where the hardware and the software are built upon the same instruction set, while hardware emulation can apply different instruction sets. In this article, we focus on the security issues of hardware virtualization. We note, however, that some platforms, for example, Xen [Takemura and Crawford 2007] and Kernel-based Virtual Machine (KVM) [KVM 2007] heavy-use hardware emulators, for example, QEMU [Bellard 2005; 2008], for device virtualization, therefore security of hardware virtualization cannot be discussed fully separately from hardware emulation.

Note that *operating system-level virtualized* platforms, such as OpenVZ [OpenVZ 2005], FreeBSD jail [FreeBSD 2000], and Oracle Solaris Containers [Oracle 2004], cannot be placed into Figure 1, as they do not allow for the running of VMs with a kernel version that is different from that of the host system. As a consequence, they do not provide true virtualization. KVM is also an interesting technology, as it virtualizes the Linux kernel but also supports hardware assisted virtualization.[2]

Another combined solution is the VMware ESX/ESXi platform, which either uses binary translation or hardware-assisted virtualization, depending on both the guest

[1]Virtualization of operating systems or computers.

[2]Enables a proprietary execution scheme: nonsensitive instructions are executed on the physical processor directly, while sensitive ones are intercepted by the hypervisor, residing at a higher privilege level (ring -1) and executed by the virtual processors implemented in the hypervisor.



(a) Native virtual platform. (b) Hosted virtual platform. (c) Our notation.

Fig. 2. Main types of hardware virtualization.

operating system and the processor architecture, in order to achieve optimized performance [VMware 2009a, 2011a]. Furthermore, both native (also known as bare-metal) and hosted virtual platforms can use paravirtualization[3], binary translation[4], or hardware-assisted virtualization to reach greater throughput and CPU utilization. For instance, VMware installs paravirtualized drivers, for example, paravirtual SCSI adapters (PVSCSI) for ESX/ESXi guests [VMware 2012], while the new MS Virtual PC exploits the rich feature set of hardware assisted virtualization [Microsoft 2012].

A unique and interesting idea is presented by NoHype [Keller et al. 2010], which assigns dedicated physical resources to VMs (processor core, memory partition, etc.) by removing the hypervisor. Therefore, NoHype eliminates the attack surface exposed by the hypervisor but offers a viable solution for secure cloud computing, as it supports multitenancy and resource utilization. See sources [Smith and Nair 2005; Adams and Agesen 2006; Scope Alliance 2008] for more detailed information about virtualization.

Hardware virtualization allows the sharing of hardware resources via hypervisors, which are software components that intercept all hardware access requests of the VMs and mediate these requests to physical devices. In many cases, a hypervisor is commonly referred to as the *host*, which should not be confused with the *host operating system* responsible for managing guest VMs and controlling hardware resources. We use a high-level system model depicted in Figure 2(c) in order to handle various system virtualization platforms uniformly. Note that this is a certain type of generalization, as the location of these layers differs from implementation to implementation. Later, in Section 2.3, we explain the figure in details. Here we separate three software layers: guest VMs, the host operating system, and the hypervisor. In the rest of the section, we discuss the main components of a hardware virtualized environment extended by network virtualization functionalities.

[3]Paravirtualization is a type of virtualization that requires modified guest OSs to replace their nonvirtualizable instructions to special hypervisor system calls (hypercalls).

[4]Binary translation is the only solution that uses neither CPU hardware virtualization extension nor OS assist to virtualize sensitive or privileged instructions. Similarly to the concepts of other binary translators, the VMM (or hypervisor) translates privileged OS requests and caches them for repeated use, while user-mode instructions are executed natively.

2.1. Guest Environments

Each single virtual machine, which comprises a stack made up from an operating system and its applications, defines a guest environment. All the guests are isolated from each other, while they use the same virtual platform provided by the hypervisor. The interface published by the physical platform is uniform and used by all the VMs to access real hardware resources. An important feature of this environment is that guests must run at reduced privilege level in order to allow the hypervisor to control their operations.

2.2. Host (Privileged) Operating System

A host operating system always has a more privileged role than guest VMs to be able to manage them and control hardware resources either directly or via the hypervisor. Note that management functionalities are not bound to host operating systems; therefore, we discuss them separately in Section 2.4. The host operating system can either be a native operating system, for example, in the case of VMware WS, or a privileged VM, for example, in the case of Microsoft Hyper-V [Microsoft 2008] or Xen. In addition, certain virtual platforms, for example, VMware ESXi [VMware 2007], Citrix XenServer [Citrix 2007], do not employ real host operating systems; only a small hypervisor and a management interface are provided to allow administrators to remotely configure the virtual server. Note that the term *host operating system* can be confusing in case of native platforms, as it refers to a privileged guest VM (e.g., Xen (dom0), Hyper-V (root partition)). However, we use this nomenclature in order to handle specific security issues (e.g., guest-to-host OS escape) conveniently in the rest of the article. See Sections 2.4 and 8 for more details.

2.3. Hypervisor and Virtual Machine Monitor

As has been introduced, the hypervisor is a software component that intercepts all hardware access requests of the VMs and mediates these requests to physical devices. A hypervisor typically implements a virtual machine monitor (VMM) component as well to manage VM hardware abstraction.

The tasks of the VMM include the selective control of hardware resources, the provision of replicated platforms, and the sharing of hardware resources between guest operating systems. Moreover, VMMs can manage and maintain the applications on the guests. If a guest invokes privileged instructions, they are first intercepted by the VMM, which checks the authenticity of instructions and performs them on behalf of the guest. It is the VMM's responsibility to ensure that these aforementioned operations remain transparent for guests. It is important to emphasize that there exist hypervisors, for example, SecVisor [Seshadri et al. 2007], that do not involve VMM functionalities.

Hypervisors and also hardware virtualization come with two main types: native (type 1) and hosted (type 2). Native (bare-metal) hypervisors (see Figure 2(a)) run directly on top of the hardware; thus they have full control over hardware resources such as CPUs, memory, devices, and so on. Therefore, they have to provide an abstraction layer for guests by adding virtual processors that allow guest codes to be executed. Furthermore, CPU scheduling, interrupt handling, I/O control, and memory management are also essential tasks to fulfil. A hosted hypervisor (see Figure 2(b)) runs as a process on top of an existing host operating system. It monitors the requests of the guest and dispatches them to an appropriate API function. Thus, the architectural position of the hypervisor and the host operating system is swapped in case of native and hosted virtual platforms. This is the reason why the boundaries of the hypervisor are indicated with a dotted line (see Figure 2(c)) in the rest of the article. Examples of native virtual platforms include VMware ESX/ESXi, Xen, Microsoft Hyper-V, while hosted virtual platforms include

VMware Workstation/Server/Player [VMware 1999], Microsoft Virtual PC [Microsoft 2006], Oracle VirtualBox [Oracle 2007], etc. At the same time, KVM raises hot debates whether it is a hosted or a native (bare-metal) platform [VirtualizationReview 2009]. Note that VMware ESXi replaced VMware ESX a few years ago and only legacy systems still use it, however, most of their concepts are the same. See VMware [2009c] for more information about VMware ESX/ESXi.

2.4. Management interface

In order to configure and manage guest VMs, vendors provide management interfaces for their products. These allow privileged users to create, delete, and modify both virtual machines and virtual infrastructures. Management interfaces reside at various levels in the software stack depending on the virtualization technology. One possible way of classifying management interfaces (and also virtualization technologies) is to examine whether they can be bound to a host operating system or not. In the first case, the management interface is a component of the host operating system, for example, VMware ESX, KVM, Xen, Microsoft Hyper-V, hosted virtual platforms. In the second case, the virtualization technology provides a management console (and a small local console with decreased functionality) to which the administrator can remotely connect with a management client, for example, VMware vSphere Client (formerly VI Client), so as to configure VMs of the virtual server. Examples for this include technologies such as VMware ESXi, Citrix XenServer, Microsoft Hyper-V Server. Furthermore, a management interface might also have a Web front end for an increased availability. These different management interfaces allow administrators to manage a virtual infrastructure at different levels. See Section 8 for more details.

2.5. Network

Virtual servers are not meant to be isolated nodes running multiple VMs simultaneously, but they are networked in order to add higher efficiency and performance. In virtual networks, one can connect VMs in the same way as one does with physical networks; in addition, networking is feasible within a single virtual server host and across multiple hosts, too. That is, network virtualization abstracts traditional networking components, for example, switches, routers, etc, as virtual appliances[5] in order to interconnect VMs. For example, virtual switches can reside in the hypervisor (VMkernel in case of VMware) or in the host operating system (Dom0 in case of Xen), each of them is mapped to a physical interface of the virtual server host and interconnect multiple VMs to communicate over the same protocol. More precisely, hypervisors provide host-only (internal-to-host) virtual networks that allow for network communication between guests on the same virtual server. This is realized by means of virtual switches inside the hypervisor (or Dom0 in case of Xen). However, these internal networks can have operational disadvantages, because traditional network tools designed for physical switches and nodes may not work properly with them.

Virtual security appliances (VSA) typically consist of a hardened operating system and a security application, for example, IDS, firewall, anti-virus software, and they aim at guaranteeing the same level of trust as their physical counterparts. In some cases, they can be plugged into the hypervisor in order to make the virtual platform more secure.

Similarly to physical networks, one can also construct VLANs[6] in virtual networks. In addition, VMs can come with one or multiple virtual Ethernet adapters, each with distinct IP addresses and MAC addresses providing flexibility, scalability and redundancy.

[5]A prepackaged software component that runs inside a VM with specific functionality.
[6]Logical grouping of stations or switch ports to form separate LAN segments.

In the following, we distinguish multiple types of networks that typically appear in well-designed virtual infrastructures. See Figure 3 for an illustration of the different types of networks as they may appear in a virtual infrastructure. We further distinguish virtual servers from virtual management servers, as the latter have management interfaces for remotely controling other VMs.

—*Public Network (PUN)*. A public network provides an interface to reach VMs and their exposed services from the public Internet.
—*Production Network (PDN)*. A production network carries the data traffic that belongs to customer VMs in the internal network of the virtual environment.
—*Physical Management Network (PMN)*. Virtual server hosts provide physical management ports which allow administrators to manage the systems remotely. Thus, they can reboot, halt, and reinstall their system. Note that physical management communication differs from its virtual counterpart.
—*Virtual Management Network (VMN)*. Refers to any virtual network that carries traffic between VMs, support services (e.g., databases, directory services, update services, etc.), and management interfaces. Basically, it transports management information to remotely configure a virtual infrastructure through added services. Note that a VMN typically enables customers to configure multiple VMs that can even run on top of different virtual servers.
—*Restricted Virtual Management Network (RVMN)*. A restricted virtual management network allows for only one VM to be manipulated; thus the customer cannot carry out infrastructure-level configurations, such as managing virtual switches, VM migration, and so on.
—*VM Migration Network (VMMN)*. For easier system deployment virtual solutions support VM migration between virtual server hosts. That is, the state of VMs can be snapped and moved to another virtual host to create the same identical virtual environment.
—*Physical Production Storage Network (PPSN)*. It is responsible for transmitting selected customer data located on the physical hard disks of virtual servers to a network-based storage, for example, network attached storage (NAS) or storage area networks (SAN).
—*Physical Management Storage Network (PMSN)*. Used to transport physical management traffic to archive them on a network-based storage dedicated to management data. Note that we suppose a separate storage for management data, as it is highly sensitive; thus, physical separation is suggested here.
—*Virtual Storage Network (VSN)*. It is used to transport customer data, for example, virtual machine disks, to a network-based storage for the sake of a more resilient, redundant, and convenient data storage, recovery, and backup.

More details about the security issues of these network segments are discussed in Section 9.

2.6. Storage

Storage virtualization abstracts away physical storage components and provides a single storage device that can be reached directly or over the network. However, this conceptual simplicity comes with the price of enhanced data management and distributed access requirements. Due to the high complexity and diversity of different physical storage subsystems (Fibre Channel (FC) SAN, iSCSI SAN, direct attached storage (DAS), and NAS), virtualization has to cover a huge semantic gap in order to present them uniformly to guest OSs. For example, VMware implements datastores that enable the allocation of storage space for VMs to transparently access a wide variety of physical storage technologies. A VMware datastore physically can either be

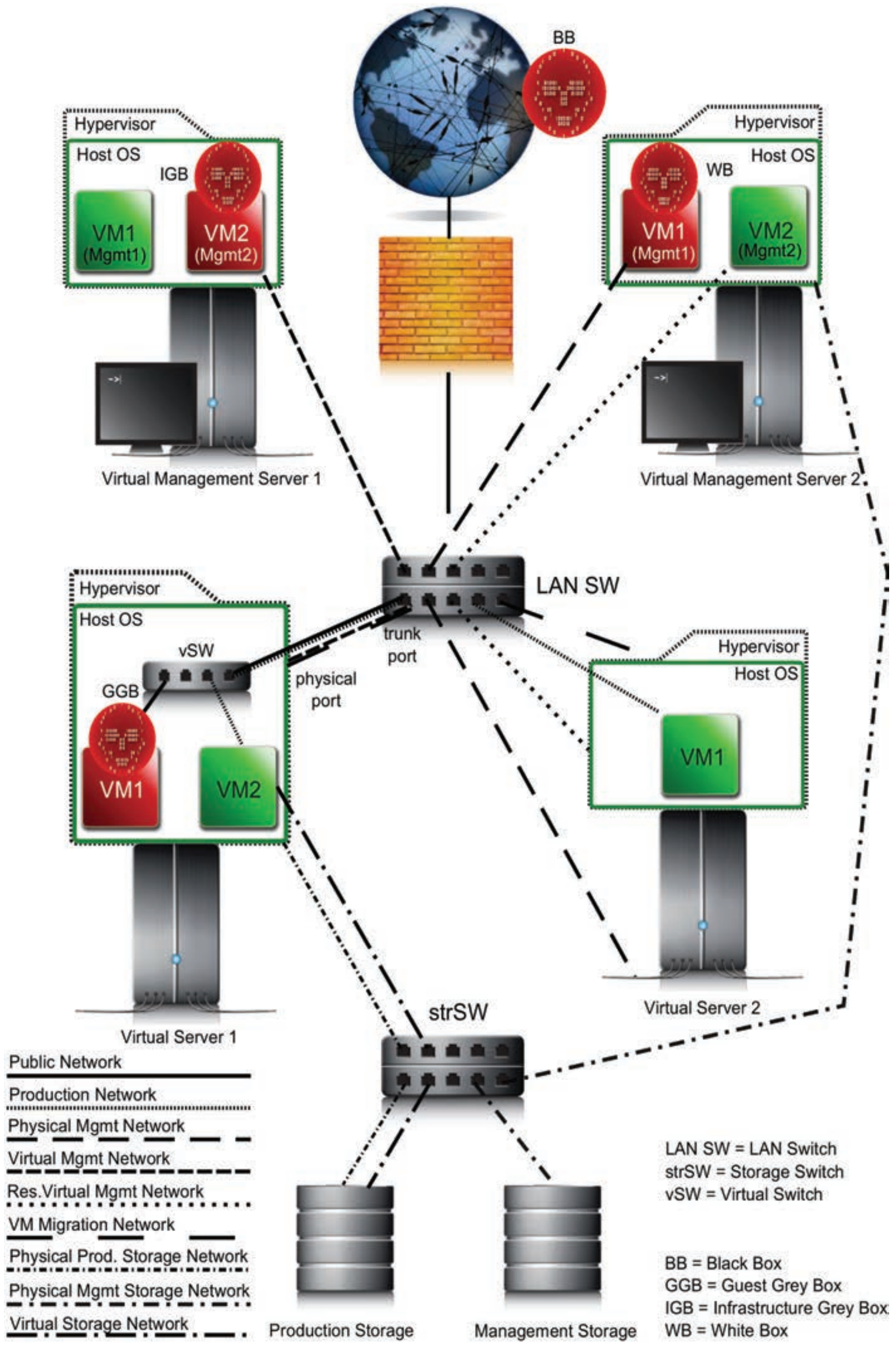


Fig. 3. System architecture including different types of networks and adversaries.

a virtual machine file system (VMFS) volume or a directory mounted as a network file system (NFS). VMFS is a clustered file system (CFS) that can be accessed simultaneously by multiple physical servers. Furthermore, VMware supports raw device mappings (RDM) technology that allows virtual machines to read and write to an existing volume (iSCSI or FC) directly[7] [VMware 2008a]. A VMFS volume can contain several virtual disks, representing a virtual hard drive for a virtual machine that can be distributed on one or more physical storage subsystems [VMware 2010]. Each virtual disk appears as an SCSI device for the virtual machine without taking into account the real physical connection (RAID, SCSI, iSCSI, NFS, or FC). Any file access to the virtual hard drive initiated from the guest OS results in the physical access to the virtual disks [Scarfone et al. 2011]. Note that various vendors use various file format extensions for virtual disks: .vmdk for VMware, .vhd for Xen and Microsoft Hyper-V, .vdi for Oracle VM VirtualBox, and so on.

In the following, we discuss the types of physical storages supported by VMware ESX/ESXi that can either be accessed through a VMFS volume or an NFS-mounted directory.

—*Local*. Stores files locally on a direct attached storage (DAS) that can be an internal or an external SCSI device and is abstracted as VMFS volume. Note that other DAS devices (ATA, SATA, eSATA, SAS, etc.) are also presented as SCSI devices to the guest.
—*iSCSI SAN*. Stores files remotely on Internet SCSI (iSCSI) storage devices that can be accessed over TCP/IP networks. iSCSI storages are abstracted as VMFS (RDM) volumes.
—*Fibre Channel (FC) SAN*. Stores files remotely on a Fibre Channel SAN through a VMFS (RDM) volume. FC was originally defined as a fast data transfer technology in order to replace SCSI/iSCSI. An FC SAN uses *fabrics* (switched topologies) to interconnect physical storage nodes and client nodes (e.g., virtual servers in our case).
—*Network Attached Storage (NAS)*. Stores files remotely on file servers that can be accessed over TCP/IP networks using NFS (*NIX) or CIFS (Windows).

3. ADVERSARY MODEL

In this work, we classify attackers into two main categories: network and local attackers, though the two may overlap. In our taxonomy, we also adopt the traditional hacking models, *black box* and *grey box*, where the former refers to an attacker without any network or physical access to the target system, while the latter supposes an internal attacker with certain privileges (LAN or computer access). We further distinguish *white-box* attacks, where a malicious user is an insider who has administrative privileges. In the following, we define the corresponding terms for virtualized environments.

3.1. Network Adversary

We suppose that a network adversary is able to sniff, block, inject, or modify the network communication; however, he is unable to break cryptographic primitives. That is, we build upon the traditional Dolev-Yao network threat model [Dolev and Yao 1981]. In the remainder of the article, we refer to attacks initiated by a network adversary who resides outside of the target system with no prior information about it as *black box* (BB) attacks.

[7]RDM can work either through a VMFS mapping file or directly without using the datastore.

3.2. Local Adversary

The distinction of local adversaries requires a more fine-grained approach, as their capabilities highly depend on the access rights granted to them. For this reason, we define three separate attack models according to the access rights of the local adversary.

—*Guest Grey Box* (GGB). An adversary who owns privileged rights to one designated guest VM can initiate guest-grey-box attacks against the virtual infrastructure. Note that he does not access any management interfaces and does not have any prior information about other customers or the organization of the target system.
—*Infrastructure Grey Box* (IGB). In an IaaS model, customers typically hire multiple VMs at the same time, which they can control through a management interface (e.g., VMware vCenter) provided by the cloud operator. Thus, a malicious customer may get access to multiple VMs that he can control at will. A local adversary with these rights can initiate infrastructure-grey-box attacks to compromise other customers or the whole system.
—*White Box* (WB). The malicious user can be an employee of the cloud operator, as well, who has either physical access to internal resources (e.g., virtual server hosts) or he can manage certain servers remotely. This is a practical problem, as large systems are supervised by multiple administrators. Thus, it is important to examine the attack that a malicious privileged user can carry out against the system.

Table I gives a detailed overview of the possible attack vectors according to the access rights of the adversary. One can see vertically the possible targets, while horizontally the adversary's access privileges are indicated in accordance with the type of attack he can initiate. In this article, we structure the discussion of these threats according to the targeted system. These can be the guest OS, the host OS, the hypervisor, the management interfaces, and various networks (communication, management, and storage). Figure 3 depicts adversaries who can launch various attacks (BB, GGB, IGB, WB) depending on their access rights to resources.

4. VIRTUAL MACHINE DETECTION

Since the appearance of hardware virtualization technologies, a considerable amount of effort has been devoted to making virtualized environments transparent. First of all, transparency is important for interoperability reasons, that is, to ensure that legacy software runs smoothly in the virtual environment. A second, more important reason for our discussion stems from the fact that anti-virus vendors heavily use virtualization to identify state-of-the-art exploits and rootkit techniques. There are several VMM-based malware detectors [Sidiroglou et al. 2005; Dagon et al. 2004] and malware analyzer environments [iSecLab 2007; Song et al. 2008; Offensive Computing 2003; Dinaburg et al. 2008] which leverage transparency in order to detect the presence of malware and observe its behavior. Therefore, detecting if an execution environment is virtualized became an important objective for malware writers, as they can prepare their code to refrain from running or to behave differently in a virtualized environment.

In recent years, virtualization has gained a huge slice in the server deployment market as more and more people and organizations turned to use it. Hence today, it is no longer in the interest of malware writers to avoid virtualized environments, because virtualization is so wide-spreadly used. Thus, malicious programs tend to operate normally in virtual environments in order to spread widely and cause large-scale infections. Consequently, anti-virus vendors keep moving their products into the hypervisor address space, for example, McAfee DeepSAFE [McAfee 2011], to operate at higher privilege levels (ring -1) than the infected VMs and stay transparent. In this situation, instead of simple detecting virtualization, malware writers have

Table I. Threat Model

	ACCESS/ATTACK		
Target	*Internet / BB*	*Guest OS / GGB,(IGB)*	*Host OS / WB*
Guest OS	CVE-2012-0392 CVE-2012-0392	[Ristenpart et al. 2009] [Suzaki et al. 2011]	
Self Guest		CVE-2009-2267	
Host OS	[Shelton 2005]	CVE-2007-4591 CVE-2007-4593 CVE-2007-4993 [Wojtczuk 2008] [Kortchinsky 2009]	
Self Host			CVE-2010-4295 CVE-2010-4296
Hypervisor		[Wojtczuk 2008b]	[Wojtczuk 2008b]
Mgmt if.		CVE-2007-4993	
Comm., Mgmt., and Storage Networks		[Dwivedi 2004]	[Oberheide et al. 2008] [Dwivedi 2003] [Dwivedi 2005] [Dwivedi 2004]

	ACCESS/ATTACK		
Target	*Mgmt if. / IGB*	*Networks / GGB,IGB*	*Physical,PMN / WB*
Guest OS	CVE-2009-3731	[Oberheide et al. 2008]	
Self Guest			
Host OS	CVE-2009-3731		
Self Host			
Hypervisor			[Wojtczuk and Rutkowska 2009] [Wojtczuk et al. 2009]
Mgmt if.		CVE-2009-3731 CVE-2009-0518	
Comm., Mgmt., and Storage Networks		[Oberheide et al. 2008] [Dwivedi 2003] [Dwivedi 2005] [Dwivedi 2004]	

Note: See vertically the possible targets that can be compromised by an adversary with access privileges enumerated horizontally. Note that the type of attack (BB—black box, GGB—guest grey box, IGB—infrastructure grey box, WB—white box) that an adversary can launch is remarked after each access privilege. Cells contain references to practical examples.

the additional objective of identifying the virtualized environment in order to initiate attacks that are specific to the given virtual environment, for example, by exploiting known vulnerabilities in a given hypervisor product. We summarize the main results on the detection and identification of virtualization in a tabular form. Table II contains references to practical examples that demonstrate that both a remote and a local adversary residing in the guest OS can detect and identify the virtualized environment. Different techniques use different information obtained from the environment, ranging from simple version information (e.g., the VMware "get version" command [Kato 2003; Klein 2008; Holz and Raynal 2005]) to resource discrepancies (e.g., PCI IDs and storages). It has been demonstrated [Pék et al. 2011] that even hardware-assisted virtualization can be detected in practice using timing discrepancies and CPU errata. Indeed, we maximally accept the key conclusion of Garfinkel et al. [2007] that a completely transparent VMM resistant to local detection methods is fundamentally infeasible to build. Thus, miscreants are capable of determining if the environment

Table II. Virtual Machine Detection and Identification

	Virtual Machine Detection and Identification		
VM	*Reference*	*Access*	*Based on*
VMware WS VMware Server	[Franklin et al. 2008]	Remote	Fuzzy benchmarking
	[Klein 2010]	Remote	High Precision Event Timer (HPET) and Time-Stamp Counter (TSC)
	[Quist and Smith 2008]	Local	Local Descriptor Table (LDT)
	[Quist and Smith 2006]	Local	CR0 Machine Status Word field
	[Rutkowska 2004]	Local	Interrupt Descriptor Table (IDT)
	[Klein 2008]	Local	Interrupt Descriptor Table (IDT) Local Descriptor Table (LDT) Global Descriptor Table (GDT) VMware "get version" command VMware "get memory size" command
	[Omella 2006]	Local	Store Task Register (STR) inst.
VMware ESX(i)	[Klein 2008]	Local	VMware "get version" command VMware "get memory size" command
	[bert 2010]	Local	Get BIOS version via DMI
	[Klein 2010]	Remote	High Precision Event Timer (HPET) and Time-Stamp Counter (TSC)
Xen	[Chen et al. 2008]	Remote	Skew in the clock frequency of TCP timestamp generation
	[Franklin et al. 2008]	Remote	Fuzzy benchmarking
MS Virtual PC	[Quist and Smith 2008]	Local	Local Descriptor Table (LDT)
	[Quist and Smith 2006]	Local	CR0 Machine Status Word field
	[Klein 2010]	Remote	High Precision Event Timer (HPET) and Time-Stamp Counter (TSC)
	[Rutkowska 2004]	Local	Interrupt Descriptor Table (IDT)
VirtualBox	[Ferrie 2006]	Local	Invalid opcode
QEMU	[Raffetseder et al. 2007]	Local	CPU errata

Note: The table summarizes the references of virtual machine detection and identification of various virtual platforms discussed in the literature. Furthermore, both the access rights (local/remote) to the virtual machine and the base of detection are also remarked.

is virtualized. The interested reader is referred to Garfinkel et al. [2007] and Ferrie [2006, 2007] for more information on the topic.

5. COMPROMISING THE GUEST

Up until this point, the adversary could detect and identify the virtual system he is about to compromise. From now on, we distinguish two types of attacks that can affect the sanity of a guest VM according to its state. First, dormant VMs are inadvertently neglected in most of the cases, which is a serious problem, as they can still contain encryption keys, authentication data, or other sensitive information that could be stolen by a rogue user. Moreover, inactive images are typically left out of security measures, such as security patches, access policies, or sensitive configurations. In addition, when a VM is inactivated, its memory content is often stored on the hard drive to be able to resume its state upon reactivation. This virtual image can be easily exposed to a data breach if it is not protected appropriately.

Active guest images can be compromised in various ways as well. One of the most threatening issues is cross-VM information flow, which allows an attacker to steal information from other guest VMs residing on the same physical host [Ristenpart et al. 2009]. Moreover, sensitive keys can be extracted from public virtual machine images (e.g., API keys from Amazon Machine Images (AMI) [Bugiel et al. 2011]) that allow a

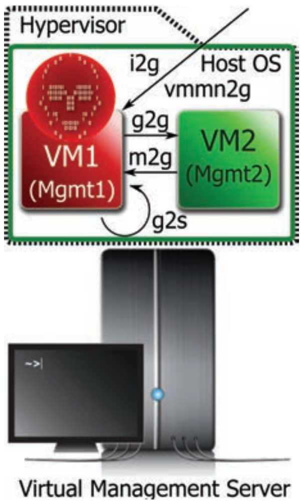


Fig. 4. An adversary can compromise a guest OS in various ways, depending on the access privileges he has. Thus, he can launch an Internet-to-guest (i2g), a guest-to-guest (g2g), a virtual machine migration network-to-guest (vmmn2g), a guest-to-self (g2s) or a management interface-to-guest (m2g) attack.

remote adversary to destroy virtual infrastructures and build one for his own nefarious purposes at the expense of the compromised customers. Another problem arises when a guest OS adversary gains extra privileges by exploiting VM-specific vulnerabilities [Ormandy and Tinnes 2009]. This section classifies these security weaknesses with the corresponding attack vectors depicted in Figure 4.

5.1. Internet to Guest (i2g)

A guest VM is exposed to external network attacks just like conventional server hosts. As a consequence, a network adversary with some black-box technique can compromise guests and gain privileges if he can successfully exploit existing vulnerabilities in the exposed services (e.g., Web (CVE-2012-0392), SSH (CVE-2010-4478), DNS (CVE-2008-0122), etc.). Note that there exist critical OS-level vulnerabilities, for example, the TrueType Font Parsing Vulnerability (CVE-2011-3402) used by the Duqu malware [Bencsáth et al. 2011], that can be exploited remotely by simple social engineering techniques, such as convincing the user to open an infected Word Doc file or visit a website with malicious content. It is reasonable to assume that such remote exploitations will remain successful in the future; therefore, internet-to-guest attacks represent a real risk. Moreover, the intruder can take advantage of the virtualized nature of the system and exploit the virtualization specific flaws in it. For example, CVE-2010-1141 discusses that VMware Tools (in specified VMware products) do not access libraries appropriately, which enables a remote adversary to run arbitrary code by deceiving guest OS users to click on the specified files being shared. However, even an attacker with this knowledge base should first fingerprint the remote system to detect if it is virtualized and attack accordingly if so. Section 4 enumerates a few references for remote virtual machine detection and identification.

5.2. Guest to Guest (g2g)

Guest-to-guest attacks suppose that the adversary has already gained access to a guest VM either by compromising it or hiring one in a cloud infrastructure. These attacks are typically indirect: first a rogue user should escape from a guest environment and then compromise other guests through privileged access to the host. In the following, passive attacks are discussed, thus other guest VMs are not compromised directly. There can be active ones as well, where a guest user has the ability to manipulate other VMs directly, but until the writing of this survey, we could not identify any such attacks in practice. Note that accomplishing a direct guest-to-guest attack could mean that a basic principle of hardware virtualization, for example, the perfect isolation of guest VMs, is violated. This could be due to an unresolved issue in the memory management module (MMU) of the hypervisor that allows a malicious user to access memory pages of other guest VMs. In that case, an adversary could manipulate those pages in accordance with his access rights (read/write/execute). Naturally, other examples can be imagined as well, but till now, we lack this type of attack.

Considering passive attacks, Ristenpart et al. [2009] examined the threat of cross-VM information leakage in Amazon EC2 clouds. This is a real threat, as many providers allow *multitenancy* where the VMs of disjoint customers can reside at the same physical hardware. The risk of memory deduplication attacks in virtualized environments is highlighted in Suzaki et al. [2011]. Memory deduplication is a well-known optimization technique applied by, for example, VMware ESX [Waldspurger 2002], Xen [Gupta et al. 2010], and KSM (Kernel Samepage Merging) for the Linux kernel [Arcangeli et al. 2009], where the VMM shares same-, or similar-content memory pages of various guests. In the case of content-based page sharing, the VMM scans the memory periodically (20 msec by default for KSM) in order to create fingerprints from pages and to check if they are identical. If so, pages are merged and shared by the guests until one of them issues a write access. In that case, the merged page is duplicated by the VMM, and the write access is given to the copied new page. This feature of the VMM, known as copy on write (COW), gives rise to a certain level of latency, thus the access time of the new page is slower than normal. Consequently, an attacker could exploit this behaviour by co-residing at the same host with the victim in a different VM, due to multitenancy or a compromised guest, so as to measure the access time of pages and reveal the presence of applications started by other VMs. By doing so, the adversary prepares its guest by installing and launching applications that are supposedly started by another VM. He waits until the periodic memory scan of the VMM realizes identical pages with other guests and merges them. The adversary issues a write access to one of the pages of the supposedly started application and measures the access time. High latency confirms the presence of the supposed application in the other VM.

5.3. Virtual Machine Migration Network to Guest (vmmn2g)

Virtual machine migration makes system deployment fast and easy by copying runtime memory images of VMs between virtual server hosts through the virtual machine migration network. Albeit, all the utility is in vain if this network traffic is not protected by secure channels, such as TLS or IPsec. A malicious user with access to the migration network could passively snoop the packets transmitted over the cable, thus stealing complete virtual images. An active network adversary could furthermore compromise the system by manipulating the guest images (e.g., embedding rootkits, keyloggers) that are going to be installed into an allegedly sanitized virtual server. A tool that demonstrates this is called *Xensploit* [Oberheide et al. 2008], which initiates a man-in-the-middle (MITM) attack to get access to the transferred VMware or Xen guest images. The authors emphasize that vendors and system administrators should pay more attention to this threat.

5.4. Guest to Self (g2s)

In case of guest-to-self attacks, the adversary can compromise the guest OS in a way so as to gain extra privileges in it. Note, here we suppose that the adversary can exploit a weakness in the virtual environment, and thus conventional privilege escalations are out-of-the-scope now. The Google security team discovered a vulnerability in several VMware products that allowed for such an attack (CVE-2009-2267 [Ormandy and Tinnes 2009]). Practically, when the processor operates in protected mode, the two least-significant bits of the code segment (`CS`) register represent the current privilege level (CPL). However, in Virtual 8086-mode, the CPL is always least privileged (3) regardless of the LSB bits of the `CS` register. Furthermore, when a page fault (`#PF`) is generated, the processor pushes the actual value of the `CS` register and the instruction pointer onto the top of the stack, together with an error code referring to the circumstances of the exception. The error code is represented by several flags indicating, for example, if the CPU was operating in user or supervisor mode (U/S bit is set or not). When a VMware guest is in Virtual-8086 mode and the adversary executes a far call or a far jump to an invalid address, a page fault is generated, and the registers are pushed to the stack using supervisory access (U/S bit is cleared) causing an invalid error code report to the guest kernel. Additionally, the Virtual-8086 mode allows a userspace code to set the two LSB bits in the `CS` register to supervisory access, which can trick the page fault handler to jump to a previously inserted shellcode in kernel mode, causing a transition from the lowest privilege level to a higher one ($ring3 \rightarrow ring0$).

5.5. Management interface to Guest (m2g)

An attacker with access to a management interface can gain extra privileges and misconfigure other customers' guest VMs. In that way, he can reroute management or storage traffic to unsecured network segments or open backdoors for external black-box attacks.

6. COMPROMISING THE HOST OS

One of the most appealing targets for an adversary is to access the host operating system. This can have unpredictable consequences, as not only the VMs running on top of it can be manipulated, but the adversary can reach and exploit other hosts in the virtual/physical network. Practically, if a host operating system is compromised, an intruder could stop/start/revert its virtual machines and/or steal sensitive information, such as unencrypted virtual hard disks, or get access to the traffic of network adapters. By doing so, he can snoop and/or manipulate storage data and virtual machine migration data transmitted over the cable and gain privileges to restricted resources, such as storages. More information about storage security is discussed in Section 9.2.4

In most of the cases, however, compromising the host OS is not a straightforward task, as OS vendors guarantee a relatively high level of protection that can be circumvented only by experienced adversaries. Yet, compromising the host OS is not impossible. Typically, the host OS provides multiple interfaces, and any vulnerability on these interfaces may be exploited, so special attention should by paid on their security. Figure 5 depicts the attack vectors of host OS compromises.

6.1. Guest to Host OS (g2h)

Guest-to-host OS attacks are one of the most luring challenges that an adversary could achieve. This threat is even more severe in a cloud computing infrastructure where miscreants can easily hire virtual machines so as to get access to a guest environment without any effort. Additionally, in such an environment, Internet-based services (e.g., Web servers, DNS servers, file servers, etc.) are virtualized; thus any of their exploitable vulnerabilities can grant access to the guest operating system with the privileges of the compromised service. Once the adversary opens a shell, he can control the guest in accordance with his privileges. That is, the privileges he escalated during the compromise enables him to initiate guest-to-host OS attacks by exploiting a vulnerability in the host OS process which shares common resources with one of the components of the

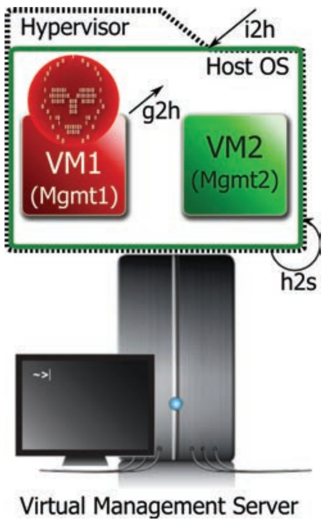


Fig. 5. An adversary can compromise a host OS in various ways, depending on the access privileges he has. Thus, he can launch a guest-to-host (g2h), a host-to-self (h2s),or an Internet-to-host (i2h) attack.

guest OS. In that way, the attacker has a communication/covert channel between the guest and the host OS through which he can escape to the host OS. In the following, a few realistic threats are demonstrated for guest-to-host OS attacks.

6.1.1. Local DoS. Denial of service is a well-known technique for making a service incapable of replying to the requests of clients. Technically speaking, an adversary exposes a targeted system to a large number of requests in order to overload it to such a degree that it cannot serve other clients. Although, there are proposed techniques [Peng et al. 2007; Yau et al. 2005] for mitigating the impact of such threats, we still lack practical implementations. The terminology of local DoS is not a newborn concept, but it poses a real security risk to the host OS executing various VMs in parallel. That is, a malicious user with access to the guest operating system could exploit a vulnerability in the virtualization platform that makes the host operating system crash, practically causing denial-of-service conditions for all other guests. Examples include CVE-2007-4591, which allows a malicious unprivileged guest user to mount a denial-of-service attack against the host OS and possibly gain kernel privileges for arbitrary code execution by reporting a small buffer size (and using a large one) to the VMware virtual image mounter driver (`vstor-ws60.sys`) of VMware Workstation 6.0. A nearly identical security flaw is the CVE-2007-4593, where a guest user can carry out a DoS attack via the `vstor2-ws60.sys` device driver. Note that there are several other known vulnerabilities that can be used for a DoS attack, for example, CVE-2008-4916.

6.1.2. Information Leakage. Information leakage is a general problem where an unauthorized party, also known as an *eavesdropper*, obtains information from a supposedly protected system without any permissions. This definition is meaningful for virtualized systems as well, where an unprivileged attacker in the guest can get access to sensitive information of the host OS. One such example is a VMware I/O backdoor [Kato 2003; Holz and Raynal 2005] residing at the VMware binary, the original intent of which is to configure VMware during runtime. However, a local guest user can command the backdoor by setting corresponding registers appropriately to read data from the host's clipboard. This poses a real security threat that allows for sensitive information flow from the host OS to the guest OS.

```
if (!access(file, W_OK)) {  // On success, zero is returned
    f=fopen(file, "wb+");
    write_to_file(f);
} else {
    fprintf(stderr, "Permission denied, cannot open %s.\n", file);
}
```

Fig. 6. Race condition in a set-user-ID program.

6.1.3. Arbitrary Code Execution. The attacks against the host OS discussed previously were passive in the sense that the attacker could not directly manipulate the operation of the host OS, that is, he could only freeze it or make it unstable. In contrast, arbitrary code execution is an active and realistic threat that can compromise the host OS to a large extent. Namely, it allows an attacker with unprivileged access to the guest operating system to execute arbitrary code on the host OS by exploiting a vulnerability in the virtualized environment. Once he had succeeded, control over the host and other guests is granted. For of example, Kortchinsky [2009] explored a vulnerability in the display function inside the vmware-vmx binary, which allows its users to launch arbitrary code on the host OS. This flaw is referred to as Cloudburst (CVE-2009-1244), and it affects both hosted and native VMware products. The attack can be accomplished with minimal preconditions (guest-grey-box), as the adversary has only to instruct the VMware SVGA II driver inside the guest VM to copy arbitrary code snippets into the memory (framebuffer) of the host's vmware-vmx process. By doing so, he can embed any code there and launch it with the privileges of the vmx process.

6.1.4. Arbitrary File Write. An adversary in the guest might be capable of gaining write access to an arbitrary file on the host OS. By doing so, he can manipulate critical system resources at will, possibly with fatal consequences. CVE-2007-1744 reveals a directory traversal vulnerability in the *Shared Folder* feature of VMware, where a guest user can utilize the I/O Backdoor, discussed previously in Section 6.1.2, to write to an arbitrary file on the host OS when a directory is shared.

6.2. Host OS to Self (h2s)

There are circumstances when an adversary could somehow compromise a host with several VMs atop it and gain restricted access to the resources of the host OS. The attacker could reveal a vulnerability in the virtualization software running on the host by means of which he can escalate privileges. Host OS-to-self attacks include such problems. In the following, two real-life examples are given to underline the existence and severity of this issue.

6.2.1. Race Condition. Race condition is an existing security problem for many software products not following safe code writing practices. Generally speaking, race condition is an alternative term for indeterministic behaviour which stems from parallel task execution. The common problem is that an operation or sequence of operations are considered to be atomic; however, in reality, this atomicity is not enforced in the software. A well-known case of race condition is called a *time-of-check-to-time-of-use* (TOCTTOU) problem, which includes the examination of a predicate (e.g., authenticity of a user) and then operates on the predicate. However, the adversary can change the state between the *time of check* (t_c) and the *time of use* (t_u). Even a narrow time window $[t_c, t_u]$ could allow an attacker, possibly after thousands of probes, to slide into that window at time t_a ($t_c < t_a < t_u$) and get access to resources with the privileges of the process. TOCTTOU problems are typical, for instance, in UNIX file system accesses.

For a better understanding, take a look at the code sample in Figure 6.

In this example, the `access()` system call is used to check the user's permission for a file opened by the `fopen()` system call. More precisely, `access()` checks the calling process's real UID and GID instead of the effective IDs used by `fopen()`. This enables set-user-ID programs (i.e., any root-owned program with the setuid file permission bit set, e.g., passwd) to determine if the invoking user owns the required access rights. However, this leads to a race condition, because the attacker may change the file reference returned by the `fopen()` function to a restricted resource, such as the /etc/shadow file, before calling the `write_to_file()` function. As a consequence, race conditions carry real threats which emerge in virtualization softwares as well. CVE-2010-4295 refers to this bug in the mounting process `vmware-mount` of hosted VMware products, which allows a host OS user to gain extra privileges. In our definition, this is a certain type of white-box attack, as the adversary has restricted access to the host. Another flaw in the `vmware-mount` suid binary is reported by Martin Carpenter and described in CVE-2010-4296. It similarly allows a local adversary on the host to escalate privileges.

6.3. Internet to Host OS (i2h)

This section demonstrates a relatively old (2005) black-box attack where a remote intruder can exploit a heap overflow vulnerability in the VMware natd module (vmnat.exe) so as to execute arbitrary userland commands on the host [Shelton 2005]. The significance of this attack is that an adversary can directly access the host without compromising a guest VM. The problem is that vmnat cannot process specially crafted *PORT* and *EPRT* FTP commands. The former defines the local PORT on which the client listens for data communication, while the latter refers to *Extended Data Port*, defined in RFC 2428, and allows for an extended address (network protocol, network/transport address) for data connection. By constructing an evil buffer content, the attacker can gain control over registers `ECX`, `EDI`, and `EBX`, which allows him to overwrite an available heap header FLINK (pointer to the next free memory chunk) and PLINK (pointer to the previous free memory chunk). Finally, this can result in the opening of a reverse shell on which the attacker could connect to the host OS with user privileges.

7. COMPROMISING THE HYPERVISOR

Compromising the hypervisor is a luring goal for miscreants; however, quite strict preconditions have to be met to be successful. In the following, we discuss two possible ways to compromise hypervisors; these attacks vectors are illustrated in Figure 7.

7.1. Guest to Hypervisor (g2hy)

Guest-to-hypervisor escapes are possibly the most frightening security issues related to hardware virtualization. That is, an intruder with access to the guest OS can compromise the hypervisor directly and execute arbitrary code with root privileges. A proof-of-concept example is the Xen FLASK exploit [Wojtczuk 2008b], which was the first public attack against a hypervisor that allowed for a guest-to-hypervisor escape. More precisely, an adversary could load malicious buffer content into the input parameter of FLASK-specific hypercalls. This malicious content caused a heap overflow that enabled the adversary to write zeros to the upper half of hypercall addresses. In this way, he could redirect these hypercalls to usermode functions, which resulted in a guest-to-hypervisor (DomU-to-hypervisor) escape. Note that FLASK is an XSM (Xen Security Modules) implementation allowing fine-grained control over security decisions; however, it is not compiled into Xen by default.

Other guest-to-hypervisor escapes [Wojtczuk and Rutkowska 2011] exploit the rich feature set of Intel's I/O virtualization technology (Intel VT-d) that enables the creation

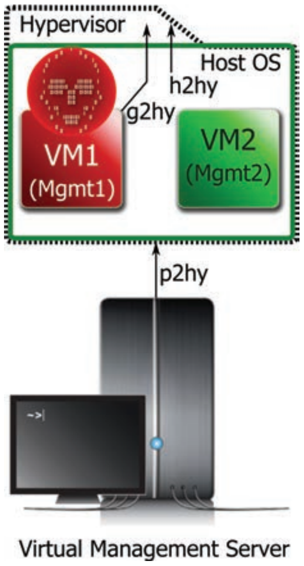


Fig. 7. An adversary could compromise the hypervisor in various ways depending on the access privileges he has. Thus, he can launch a guest-to-hypervisor (g2hy), a host OS-to-hypervisor (h2hy), and a pyhsical/pyhsical management interface-to-hypervisor (p2hy) attack.

of driver domains or driver virtual machines (guests). The only difference between a driver domain and a traditional guest is that the former has direct access to certain physical resources, such as network cards or disk controllers. The introduced (theoretical and practical) attacks send specially crafted message signaled interrupts (MSI) from an untrusted driver domain via a network interface card (NIC) to one of the processors in a multiprocessor (multicore) system. These malicious MSIs allow the adversary to execute code in the context of the Xen hypervisor or access arbitrary physical memory locations. Note that these attacks are either limited to paravirtualized domains (e.g., in Xen) or processors without Interrupt remapping capabilities.

All in all, these exotic attacks perfectly demonstrate that even hypervisors and the x86 architecture can contain serious vulnerabilities, which can be exploited from the guest both theoretically and practically.

7.2. Host OS to Hypervisor (h2hy)

The runtime modification of hypervisors is an appealing problem for many adversaries, as they could insert silent backdoors and rootkits underneath the operating systems. In the Xen 0wning Triology [Wojtczuk 2008b], the author presents several ways to modify the Xen hypervisor via DMA transfers. The attack supposes that the adversary has root privileges on the host OS, which is not impossible to achieve (see Section 8.1 for demonstration). Furthermore, we cannot exclude the existence of white-box attacks. One way to succeed is to use the loopback mode of network cards which allows an adversary to copy arbitrary data between two locations in the RAM. Note that I/O virtualization solutions (AMD IOMMU, Intel VT-d) can limit the address range that a DMA device can access; however, these technologies are not supported in every case. In addition, the author of Wojtczuk [2008b] could present another attack at the same time, where even an Intel VT-d-aware architecture could be tricked.

7.3. Physical/Physical Management Interface to Hypervisor (p2hy)

Physical white-box attacks are not so frequent in reality, yet they present a serious threat due to rogue malicious system administrators and employees. In a complex IT infrastructure, multiple system administrators supervise the system, and each of them may have a specially assigned role. Thus, we can suppose the existence of a malicious admin with role-specific tasks. Such an administrator could shutdown, reboot, etc. a virtual server host with multiple VMs on top of it. Furthermore, guest-to-host escapes (see Sections 6.1.3 and 8.1) could allow a guest user to execute arbitrary privileged instructions on the host. Wojtczuk et al. [2009; Wojtczuk and Rutkowska [2009], supposed such preconditions when constructing evasive attacks for Intel Trusted Execution Technology (TXT). Intel TXT builds upon a trusted platform module (TPM)[8] to give hardware support for trusted operating system/VMM load and execution. However, Intel TXT does not provide runtime protection; thus a vulnerability in a hypervisor could still be exploited after it is launched. The first attack [Wojtczuk and Rutkowska 2009] consists of two stages: hooking the CPU's system management mode (SMM)[9] handler and compromising the securely-loaded hypervisor (here Xen) via that attack vector. Note that the focal point of the attack is the embedded SMM code, which is notoriously known to be executed at a higher privilege level, in ring -2, than the hypervisor itself. The second attack [Wojtczuk et al. 2009] could misconfigure the TXT chipset in such a way that a securely loaded hypervisor could be compromised via traditional DMA attacks.

8. COMPROMISING THE MANAGEMENT INTERFACE

Access to the management interface can be critical, considering the overall security of the virtual environment. As has already been introduced in Section 2.4, management interfaces can reside at different locations in a virtualization software stack depending on the virtualization technology. That is, management interfaces can either be bound to an existing host operating system, or the virtualization technology provides only a small management console to which the administrator can remotely connect by means of a management client. In the following we give examples for these two categories.

—*With host operating system*. In the case of Xen, the management interface is part of the host operating system (Dom0).[10] Dom0 is a paravirtualized VM that has privileged and direct access to all I/O devices (e.g., disks, network devices, PCI devices, etc.). It has two key roles: administering other domains (creating, destroying, saving, and so on), which is realized by the control panel component (management interface), and making an abstraction layer on top of the hardware devices to the hypervisor and hence to virtual machines. Thus, a guest-to-management interface escape can correspond to a guest-to-host OS escape in the case of Xen. However, we do distinguish these two categories, because the adversary has the potential to control guest VMs with such an attack (see Section 8.1.1 for an example).

—*Without host operating system*. In the case of VMware ESXi, the vendor provides a management client called vSphere (formerly VI Client) which can either be used to connect directly to the exposed management console, or to a central management interface called vCenter (formerly Virtual Center). Using vSphere/VI Client, one could configure and control a standalone VMware ESXi server, or by means of vCenter/Virtual Center, one could manage a whole VM infrastructure with multiple

[8]Both the name of the specification of a secure cryptoprocessor that stores cryptographic keys and the implementations of that specification.

[9]An operating mode on the x86 CPU that is responsible for the low-level hardware control.

[10]Also referred to as control domain.

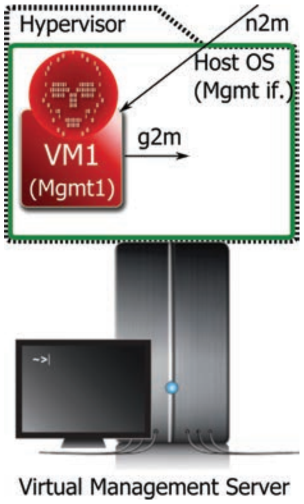


Fig. 8. An adversary can compromise the management interface in various ways depending on the access privileges he has. Thus, he could launch a guest-to-management interface (g2m) or a network-to-management interface (n2m) attack.

virtual servers and network capabilities. Furthermore, VMware provides a Web-interface for these products for an increased availability. An attacker with access to a central management interface could compromise a whole virtual infrastructure. See Sections 8.1.2 and 8.2 to demonstrate the reality of these threats.

In the rest of this section, we suppose a local adversary who has the capability to perform an infrastructure-grey-box attack by compromising a management interface. See Figure 8 for the representation of attack vectors.

8.1. Guest to Management Interface (g2m)

As has already been mentioned previously, a potential attack surface is the management interface that could be compromised from a guest VM, for example. This section discusses two examples of this case.

8.1.1. Arbitrary Code Execution. CVE-2007-4993 is one example where a privileged guest user over Xen 3.0.3 can craft a malicious boot config file (grub.conf) to execute arbitrary commands in the host operating system during system launch. By doing so, he can control other guest VMs or compromise the host OS at will. Note that this is an indirect attack against the OS of the administrator, so in Table I, the threat is classified as a guest OS-to-management interface/host OS attack as well.

8.1.2. Information Leakage. Another security problem is described in CVE-2009-0518, where the VMware VI Client retains the server-side password in its process memory after connecting to the virtual center that might allow a guest user to read it out. In this way, a malicious user, intruding the guest via a black-box attack, could read the management password of the legal guest user. Figure 3 illustrates how an adversary on the virtual management servers could launch infrastructure grey-box attacks via the compromised management interface.

8.2. Network to Management Interface (n2m)

Management interfaces often provide Web surfaces for more convenient system monitoring and controlling. However, similarly to any other Web applications, these sites may not meet the sufficient security requirements. CVE-2009-3731 reports multiple cross-site scripting (XSS) vulnerabilities in various VMware products, which could allow a remote attacker to inject arbitrary script or HTML code into the Web surface. By doing so, when a system administrator opens the Web interface next time, the script embedded there is executed on his local computer, causing a system attack. From that point on, the impact of the attack depends on the intruder. He could open a root shell, for example, which might allow him to compromise other servers in the virtual architecture with the privileges of the administrator.

9. COMPROMISING NETWORKS

Network virtualization comes with a diverse and useful feature set; however, it still cannot provide the same security level as physical networks. First of all, virtual network components can suffer from the same software weaknesses (e.g., buffer overflows, integer overflows, etc.) as traditional applications do. In that way, virtual network traffic can be rerouted to unsecured paths or circumvented somehow to leak out information. Second, network threats can be mitigated by virtual security appliances (VSAs) (e.g., firewalls, anti-viruses, IDSs); however, these carry the same risk as conventional approaches. Moreover, VSAs are still not strong enough to prevent or contain sophisticated attacks and malcodes. Third, ten years ago, L2 switches suffered from many fatal security issues due to VLAN hoppings, ARP spoofing, spanning tree attacks, and so on. Nowadays, these issues are successfully resolved by many products, so most of these attacks are ineffective on a factory default installation. However, virtual networks with virtual switches do not hold such premises. By compromising a network segment via L2 (virtual) switches, an attacker can sniff and manipulate the traffic, snoop information, and so on.

In the rest of this section, we give a broader overview on traditional L2 attacks that an internal adversary could use in virtual networks. Here we suppose that different networks discussed in Section 2.5 are segregated into different VLANs/network segments, as best practices suggest. Then we detail the specific threats of various networks.

9.1. Attack Surfaces of Physical and Virtual LANs

Each network segment, discussed in Section 2.5, has its own proprietary goal of physical or virtual separation, so segregation should be a focal point in guaranteeing the basic level of security. Theoretically, all the various network communications are put into a distinct physical port; however, most of the time this is not feasible due to the lack of physical ports in virtual servers or other restrictions. Practically, different network traffics are bundled into a common physical port that is represented by a virtual switch in the hypervisor by trunking, and different VLANs are created for each of them (see Figure (3)). That is the reason why there can be situations (e.g., vulnerability in a virtual switch, VLAN hopping in switches or virtual switches) when an adversary escapes its VLAN to compromise other communications.

9.1.1. VLAN Hopping. VLAN hopping exploits networks with multiple VLANs by enabling an attacker to escape from his VLAN segment and intercept or modify network traffic transmitted in other VLANs. VLAN hopping attacks are typically conducted within Cisco's proprietary Dynamic Trunking Protocol (DTP) which negotiates trunking between two VLAN-aware switches. Primary targets are the 802.1q or InterSwitch Link (ISL) trunking encapsulation protocols. The adversary creates network traffic tagged with VLAN ID destined outside the VLAN he resides at. A rogue guest VM over a VMware ESX server can also generate such frames if virtual guest tagging (VGT) is allowed for an 802.1q trunking. Note that ESX/ESXi hosts support three types of VLAN tagging modes: external switch tagging (EST), virtual switch tagging

(VST), and virtual guest tagging (VGT) that define whether VLAN tagging of packets is performed by the physical switch (EST mode), the virtual switch (VST mode), or the virtual machine (VGT mode). In the case of the VGT mode, for example, VLAN tags are preserved between the virtual machine networking stack and the physical switch when L2 frames are transmitted to/from virtual switches. Furthermore, physical switch ports must be set to a trunk port that makes the routing feasible between virtual and physical switches (see [VMware 2006] for more information). The attacker can also mimic a switch/virtual switch which negotiates trunking, enabling himself to not just send, but to receive network traffic of other VLANs.

9.1.2. CAM/MAC Flooding. Switches contain content-addressable memory to store VLAN, port, and MAC address mappings in CAM tables so as to forward frames to appropriate egress ports. These CAM tables have size limitations that an adversary could exploit. Each frame with a different MAC source address is mapped to a corresponding port and inserted into a different row in the table. However, when this table is completely populated, the switch acts as a hub and broadcasts the frames to all ports, except for the one the frame came from (loop prevention).

9.1.3. ARP Spoofing. ARP (Address Resolution Protocol) spoofing is a worrisome problem to defend against even in virtual LANs. Here, an adversary residing in a LAN segment can modify the ARP table of the LAN hub/switch so as to redirect the network traffic to himself. He can use Gratuitous ARP[11] or other ARP messages to poison the segment, causing the hub/switch to associate the adversary's MAC address with the IP address of another host. Thus, he can mediate and launch man-in-the-middle attacks between any two parties by forwarding the intercepted and/or modified traffic to the original target (e.g., edge gateway).

9.1.4. Spanning Tree Attacks. Malicious users can exploit the Spanning Tree Protocol (STP) to conduct a DoS attack against a network segment by generating bogus Bridge Protocol Data Units (BPDUs)[12] to become the root bridge or enforce certain ports to be blocked.

9.1.5. DHCP Address Range Starvation. An adversary on the LAN can starve the local DHCP server out of free addresses by continuously requesting new dummy leases in the DHCP range. As a result, requests from other clients are refused. Effectively, this results in a DoS attack, which lasts until the leases expire.

9.1.6. MAC Address Spoofing. By MAC address spoofing, an attacker can impersonate another host. This is a valid problem also for virtual environments, as the attacker may change the MAC address of either the guest VMs or the host operating system in order to access restricted resources.

9.1.7. Promiscuous Mode. An attacker with access to management interfaces or a system administrator who is about to monitor virtual network traffic can enable promiscuous mode for virtual switches in case of VMware ESX/ESXi. This means that either physical or virtual hosts connecting to the same network can sniff its entire traffic, posing a severe security threat. Unfortunately, in certain cases, promiscuous mode is inevitable, as it makes error detection and correction much simpler for administrators.

[11]It assists to update other machines' ARP tables. See Wireshark [2006] for more information.
[12]A special data frame used by bridges to exchange information about the network so as to determine the root bridge.

9.2. Compromising Networks of Virtual Systems

9.2.1. Compromising the Physical Management Network. Accessing the physical management network enables an adversary to completely compromise a deployed system. By doing so, he can shutdown, reboot, configure both physical and virtual machines at will. That is the reason why hosts to be managed are interconnected via dedicated management ports that only highly privileged super users can access.

9.2.2. Compromising the VM Migration Network. VM migration communication is typically segregated into a distinct LAN or VLAN, because highly sensitive plaintext data is transmitted here. Compromising the VM migration network [Oberheide et al. 2008] allows an attacker to steal sensitive data about guests that can even be manipulated. The threat is valid as most of the migration solutions still do not encrypt the traffic. Note that recent versions of VMware's migration product (vMotion) solves this issue by means of SSL; however, only a few deployed virtual environments employ this feature.

9.2.3. Compromising the Virtual Management Network. Virtual management communication should also be put into a separate network segment due to the critical nature of information. By accessing management communication, an adversary can modify the virtual network topology, manipulate port-to-VLAN mappings, set the virtual switches into promiscuous mode (thus allowing the interception of other network communications in the same VLAN), open ports to create backdoors, and so on. Consequently, virtual management communication should always have a dedicated segment that is secured against L2 attacks. Additionally, default virtual machine ports can be enabled when installing a VMware ESX host, although these are created at the same network interface as the one used by the service console. Thus, virtual machines can get access to plaintext and sensitive management data.

Management clients (vCenter, VI Web Access), in case of VMware, access the ESX server hosts via self-signed SSL certificates. This poses another threat: self-signed certificates are not issued by a trusted third party; thus, anyone can generate new key pairs and sign the public key with the private one. By doing so, innocent users simply accept these certificates too, as they usually do.

9.2.4. Compromising the Storage Network. Storage virtualization also has strong security requirements, because, first of all, stored data may contain all sorts of sensitive application data, and second, it may contain system data that allow the adversary to compromise the virtual environment. Note that local storage attacks, for example, accessing .vmdk files on a hosted virtualization platform, can result from known compromises, such as guest-to-host escapes (discussed earlier); therefore, we do not address them in detail here. Instead, we focus on the attacks against virtualized storage services based on network storage solutions. In the sequel, we assume that the adversary has already obtained all the required privileges, for example, by escalation techniques previously discussed, to perform the following attacks against storage networks.

General Problems. The most threatening issues considering storage network security are snooping attacks where a malicious user can passively sniff the data being transmitted. The problem stems from the fact that neither the hardware nor the software layer offers real solutions for addressing this issue. For example, Fibre Channel (FC) transports all data in clear text, and thus appropriate access control and isolation from unprivileged resources should be ensured by other means. Moreover, software vendors such as VMware provide a storage migration solution (Storage vMotion) which can move active virtual machine disks from one storage to another without downtime. However, storage migration transfers virtual disks in plain format, thus a man-in-the-middle attacker could steal or modify entire disks or infrastructures. In addition, virtual disks on shared storages can also be accessed by anyone with the necessary privileges [VMware 2011b]. In the case of Xen, too, the storage traffic is unencrypted and goes through the control domain (Dom0) if one uses IP-based storages (NFS, iSCSI). Both for security and availability reasons, it is necessary to be segregated from the management communication. That is, the IP addresses of dedicated storage network interfaces should belong to different subnets than those of the management networks. File-based storage repositories mount a dedicated NFS repository into Dom0, where each storage repository is a directory on the NFS server. This server furthermore stores virtual disks in .vhd file formats that are not encrypted either, so only highly privileged administrators should access them.

Another problematic issue is the DoS. Similarly to any other networks, DoS attacks represent a valid threat against storage networks, too, with no effective solution for addressing them.

Compromising the Fibre Channel SAN. FC was always considered as a general storage mechanism; however, its performance-to-security trade-off always raised debates. Furthermore, Fibre Channel SANs face many other serious security problems that could be valid in virtual environments [VMware 2011b] as well. Attacks against FC SANs can be classified into three main groups: compromising the FC HBA, the FC Switch, or the FC Frame. Due to page limitations, we introduce only one example for each class; however, interested readers can find more information in [Dwivedi 2003].

—*WWN Spoofing (FC HBA Attack)*. Each node in the FC SAN has a 24-bit address administered by the fabric, and a 64-bit World Wide Name (WWN)[13] determined by the host bus adapter (network interface card.). An adversary with access to the HBA can spoof its WWN, which allows him to gain unauthorized access to data that should have been accessed only by client nodes with the spoofed WWN.

—*Zone Hopping (FC Switch Attack)*. Zoning is a segmentation tool that allows system administrators to separate data stemming from various sources. Zoning is often considered to be an effective solution for storage security problems; however, we note that it lacks enforcement capabilities, which actually limits its effectiveness. Basically, there are two types of zoning defined: hard (enforcement based) and soft (information based), each of which supports both WWN-(zones are based on WWNs) and port-based zoning (zones are based on the physical ports of FC switches). Depending on the zoning technique being used, an attacker can hop between zones to access data residing in a zone he should not have access to. More precisely, by spoofing the WWN, an adversary can subvert the zoning table and evade both hard and soft zones. In the case of port-based zoning, WWN spoofing is ineffective; however, if an attacker knows the route to another WWN in a different zone, soft zoning grants the access. As a consequence, only hard zoning based on port numbers can eliminate both types of hopping attacks.

—*Session Hijacking (FC Frame Attack)*. Each layer 2 FC frame transmitted over the cable belongs to an FC sequence, which is defined as the series of related frames passed unidirectionally from one port to another. Each frame within the same sequence contains an identical SEQ_ID value and is identified by a counter (SEQ_CNT) being incremented by 1 after every frame transmission. This type of operation allows the recipient to arrange frames into the right order and to determine if all the required frames have arrived. These conditions allow an adversary to accomplish a session hijacking attack as follows. He can capture the SEQ_ID value of a sequence being transmitted in plaintext and also predict the SEQ_CNT value. In addition, there is no integrity check on frames, so the adversary can fabricate

[13]WWN represents the same concept as the MAC address in Ethernet.

frames with modified SEQ_CNT, and by doing so, hijack existing sessions and get access to sensitive resources.

Compromising the iSCSI SAN. iSCSI (Internet Small Computer System Interface) is another SAN protocol that, similarly to FC, allows block-level access to storages; thus entire LUNs (logical unit numbers)[14] can be accessed by iSCSI clients. In contrast to FC, iSCSI is used over IP networks by carrying SCSI commands (CDBs) between an initiator (iSCSI client—NIC with an installed iSCSI driver) and a target (iSCSI storage device). The main advantage of iSCSI is that it does not require any special hardware and works over existing LANs; however, it is also an unencrypted protocol. In the following we introduce some practical attacks on iSCSI; more examples are discussed in [Dwivedi 2005].

—*CHAP Attacks*. In 2009, VMware introduced new iSCSI software initiators with more enhanced security [VMware 2009d]. By means of bidirectional authentication, both the initiator and the target can authenticate each other via the Challenge-Handshake Authentication Protocol (CHAP). Therefore, an adversary should know the preshared secret to impersonate either the initiator or the target of the communication. However, by configuring VMware to not authenticate both the initiator and the target, an adversary could potentially impersonate either side of the connection and accomplish a man-in-the-middle attack. Furthermore, by capturing the credential hash, the adversary is not required to know the current credentials in order to hijack the CHAP session. This is also referred to as the "Pass the Hash" attack [Butler and Vandenbrink 2009].
—*iSNS Man-in-the-Middle*. iSNS (iSCSI Simple Name Services) is an optional iSCSI component with similar roles to DNS. It maintains a table in order to group initiators and targets into separate domain sets for logical segmentation. iSNS servers are used to register initiators and targets and inform initiators about specific events, for example, the availability of targets on the network. In an iSNS man-in-the-middle attack, the adversary identifies the iSNS server listening on port 3205 in the first place. By using ARP poisoning, he can impersonate the valid iSNS server with a fake one which is used as a mediator. That is, the fake iSNS server captures and relays all the traffic sent to the valid iSNS server, so the adversary has full control over the plaintext packets being transmitted.

Compromising the NAS. NAS (network attached storage) is a remotely attached storage that supports local file systems being accessed over IP networks via CIFS (for Windows), NFS (for *NIX), HTTP or FTP. Similarly to the aforementioned storage solutions, NAS also comes with weak security guarantees. A few corresponding attacks are discussed next. A more detailed discussion can be found in [Dwivedi 2004].

—*Export/Share Enumeration*. First of all, NAS exports (e.g., /dev/dsk/server), NAS shares (e.g., C$), and NAS usernames (e.g., administrator) can be enumerated by an adversary easily and can be used later to launch even more specific attacks.
—*Authentication Attacks*. After enumeration, an attacker might access CIFS shares and NFS exports by anonymous access rights. If that is the case, an attacker could easily access sensitive resources.
—*Bypassing Permissions*. This attack supposes that a network share supports both CIFS (Windows) and NFS (*NIX) access. The attacker X wishes to get access to a share he has no file permission to, but he knows that user Y has full control (rwx)

[14]Logical unit numbers are used to identify a logical array of storage units that can be any device addressable by the SCSI protocol. Note that the term is a bit misleading, as an LUN usually refers to the logical unit itself.

over that. X can access that folder neither from Windows nor from *NIX, but he can check the UID and GID of Y from a *NIX console (ls -al). As X has local root access, he can edit the local password file (/etc/passwd) to change the UID and GID of his account (X) to the UID and GID of Y. By doing so, the CIFS permissions are subverted by means of NFS, and access is granted to the restricted resource.

As just demonstrated, storage virtualization carries potential risks into virtual infrastructures; thus dedicated attention should be paid to their appropriate configuration.

9.2.5. Compromising the Production Network. The production network transmits customer-specific traffic, including Web access, VPN, and so on. Best practices suggest isolating production networks as well, which are typically connected to the internal network via the data management zone (DMZ). That is, DMZ servers should be secured in order to contain malicious users (either network or local). In addition, as nowadays DMZ services are preferred to be virtualized as well, more enhanced countermeasures have to be applied to secure them [Cisco and VMware 2009]. More information about networking issues can be found in VMware [2008b, 2006, 2009b].

10. MISCELLANEOUS THREATS

Up until now, a considerably wide spectrum of attacks on hardware virtualization has been introduced with detailed description of flaws or design vulnerabilities. Albeit, there are other threats called *virtual-machine based rootkits*, which install a VMM underneath an existing operating system and execute the original OS in a guest system. By doing so, the rootkit can mediate between the hardware and the OS in order to perform naughty activities, such as key logging, opening a backdoor, and monitoring the OS. In 2006, three such projects emerged—Bluepill [Rutkowska 2006], Subvirt [King et al. 2006], and Vitriol [Zovi 2006]—each of which targeted different platforms and operating systems. Bluepill and Vitriol are capable of installing themselves underneath a running operating system on the fly, without reboot, while SubVirt modifies the boot process in order to activate itself.

11. COUNTERMEASURES

Following the detailed discussion on various vulnerabilities and attacks in hardware virtualized environments, in this section, we give a brief overview on possible countermeasures. We note, however, that the main objective of this article is to survey the vulnerabilities rather then giving a detailed introduction of the countermeasures. Therefore, we do not aim at completeness in this section, and we keep the discussion at a rather high abstraction level.

11.1. Secure Programming

In most of the cases (Sections 5.4, 6.1.1, 6.1.2, 6.1.3, 6.2.1, 6.3, 8.2), malicious users exploit a software bug in virtualization products that allows them to compromise a system to a certain extent. To avoid this, vendors need to generally mitigate such bugs by applying well-accepted secure programming methodologies and/or formally verifying the compile-, and runtime behaviour of their products.

11.2. Hardening the Hypervisor and VMs

Hardening of the hypervisor and the hosted VMs should also be a focal point of system-wide security. That is, the latest security patches for traditional applications should be also installed both on the hypervisor and the hosted VMs. Appropriate configuration of exposed services and security controls (e.g., Web servers, firewalls) is required, as black-box attackers often exploit their weaknesses. The main contributions in this area are summarized elegantly in Szefer et al. [2011] as follows.

First, by minimizing the size of the code in the hypervisor, one can reduce the attack surface being exposed; however, this is not a practical workaround in current virtual systems. One example of this approach is SecVisor [Seshadri et al. 2007] that supports and secures one single guest VM, while TrustVisor [McCune et al. 2010] allows for integrity protection of code and data in designated application portions. Second, other ideas [Suh et al. 2005; Lie et al. 2000; Lee et al. 2005] offer new processor architectures, although these cannot be deployed into current virtual systems either. Third, to harden the hypervisor, researchers came up with various solutions [Li et al. 2010; Sailer et al. 2005; Steinberg and Kauer 2010]. For instance, Hypersafe [Wang and Jiang 2010] protects the hypervisor from control-flow hijacking attacks, while HyperSentry [Azab et al. 2010] operates in ring -2 (system management mode) that allows for stealthy and in-context measurement of the runtime integrity of the hypervisor. Finally, exokernels, such as ExOS [Engler et al. 1995] and Nemesis [Leslie et al. 1996], implement reduced operating systems that allow more direct access to the hardware.

Note that all the preceding issues are addressed by the NoHype system [Szefer et al. 2011] which eliminates the attack surface of hypervisors; thus, VMs cannot exploit its vulnerabilities. It is achieved by (1) the preallocation of processor cores and memory resources to separate VMs, (2) the exclusive use of I/O virtualized devices, (3) the support of the system recovery process with a temporary hypervisor and a modified guest OS, and (4) the elimination of any interaction between the hypervisor and guest VMs. A similar solution is the SICE [Azab et al. 2011] framework that does not rely on any host system (hypervisor or host OS) so as to provide secure and isolated environments. This is achieved by means of a TCB (trusted computing base) that includes only the hardware, the BIOS, and the SMM (system management mode).

11.3. Restriction of Physical Access

The physical or physical management network access of virtual server hosts should be restricted as much as possible (see Section 7.3), as it can simultaneously mean access to VMs, storages, networks, hypervisors, virtual appliances, and so on. Unused physical interfaces should be disabled in order to tailor the attack surface and limit the caused damage.

11.4. Policy and Isolation

Control policies of virtual environments should be implemented and maintained to provide the same level of security that physical environments have. Furthermore, the isolation of security functions is another core point of system-wide hardening. For example, firewall functionalities should never be combined with private key storing on the same virtual host, as it can be a single point of failure that affects the integrity of the whole virtual environment.

11.5. Separation of Roles

Administrative roles should be separated, as best practices suggest. As large virtual environments are supervised by multiple administrators with different roles and duties, their access rights should be restricted to dedicated roles according to VM function, hypervisor, virtual network, hardware, application, storage, and management interfaces. For example, a storage administrator should not get access to firewalls and monitoring services, and vice versa. Furthermore, multifactor authentication and dual- or split-control administrative passwords for various administrators should be used in order to reduce the risk of a malicious supervisor. Role-based access control (RBAC) should be maintained for virtual components in order to avoid unwanted access to resources.

11.6. Separation of VMs

VMs with different security levels should be hosted on different virtual servers. A VM with lower security requirements typically has weaker security controls which can be ideal for an attacker to compromise other VMs with higher security requirements being hosted on the same physical hardware.

11.7. Considering the State of VMs

Section 5 highlighted a few security problems for unprotected inactive VMs. More precisely, migration path of inactive VMs should point to secured routes that cannot be compromised by man-in-the-middle attackers. Inactive VMs containing highly sensitive information should be stored and maintained at the same security level as any other active VMs containing similar pieces of information. Backup and restoration of active/inactive VMs should also be treated securely. Furthermore, data that are not required any more should be deleted simultaneously in each copy of a VM.

11.8. Mitigating Design Discrepancies

There are design-level issues where software vendors must take the security-to-performance trade-off into account. For instance, in case of guest-to-guest attacks (Section 5.2), virtual environments should be critically safe, especially when multi-tenancy is supported. Direct guest-to-guest attacks include the passive memory deduplication attack which can be mitigated by setting the pages of target applications read-only in the page table entries (PTE) of the VM so that the adversary cannot measure write access latency and cannot predict the launch of an application. However, the VMM should know the number of same-page read-only pages in order to prepare such a countermeasure. Another possible workaround is the obfuscation of the code image loaded into the memory by the victim's operating system. This concept prevents the attacker from creating pages with the same content as those of the original application.

Section 7.2 demonstrated a rootkit that initiates a host-to-hypervisor attack to manipulate the runtime behaviour of the hypervisor. Unfortunately, hypervisors do not provide runtime protection against such malcodes. Technically speaking, timing analysis is one option for detecting the presence of a certain variant of this malware; however, there are other enhanced implementations that resist timing analysis. As the sanitization of an infected hypervisor is not trivial, preventive countermeasures have to be employed, including the installation of most-recent security patches, although it still does not solve the problem of white-box attacks.

11.9. Securing the Network

Certain network discrepancies, such as sniffing and snooping of migration and storage traffic (as discussed in Sections 5.3, 9.2.2, 9.2.4) can be mitigated by appropriate secure communication channels, such as SSL or IPsec. Traditional L2 networks and stations are implementing defence methods against a wide spectrum of attacks; however, virtual networks still carry the threat of such attacks, as there can be either design- or implementation-level flaws in their software components. VLAN hopping (Section 9.1.1) can be mitigated by disallowing virtual guest tagging, configuring ports forwarding tagged traffic to be a trunk, and transmitting only certain tagged frames. Furthermore, unused ports inside VLANs transmitting valuable traffic should be put to another unused VLAN, disallowing the attacker to plug or enable it and get access to its communication. CAM/MAC flooding (Section 9.1.2) can be contained by restricting broadcast domains or restricting the maximum number of MAC addresses that can be learned on ports (port security). To avoid ARP spoofing (Section 9.1.3), vendors apply some kind of cross-checking or authentication of ARP responses. If a virtual switch does not conform consciously to any defence mechanisms, it is highly exposed to such attacks. Spanning Tree attacks (Section 9.1.4) can be prevented traditionally by proprietary tools (e.g., bpdu-guard, root-guard, of the vendor), so virtual switches have to implement such methods by default. To alleviate the risk of DHCP address range starvation (Section 9.1.5), switches and virtual switches must be configured to allow only certain IP/MAC addresses to access the network. MAC address spoofing (Section 9.1.6) can also be prevented. In the case of VMware ESX, the administrator can disable the guest OS to change the virtual MAC address. More precisely, the guest OS believes that the MAC address has changed; however, the virtual network adapter does not get any packets if its MAC address is not equal to the one originally assigned to it.

To alleviate the risk of most of the attacks targeting storage networks, system administrators should separate this traffic and apply secure channels (IPSec, SSL) to protect the data transmitted here. This can prevent snooping and session hijacking attacks (Section 9.2.4); however, other active adversaries can still be successful. To thwart WWN spoofing and zone hopping (Section 9.2.4), port-based hard zoning should be deployed in FC SANs. CHAP attacks in iSCSI networks can be evaded by bidirectional authentication, while an iSNS man-in-the-middle adversary (Section 9.2.4) can be contained by protecting against ARP spoofing, as previously discussed. Finally, anonymous access requests should be rejected in NAS networks, and administrators should take care of applying CIFS and NFS sharing simultaneously (Section 9.2.4).

11.10. Adequate Logging and Monitoring

The logging and monitoring of all activities in a deployed virtual environment is highly recommended. Logs should be directed into separate physical storages that are secured with appropriate functions. This helps to identify and analyze data breaches or any compromises that affect the integrity of communication channels, security controls, or segmentation. Out-of-VM analysis platforms enable both OS-level [Dinaburg et al. 2008] and process-level [Srinivasan et al. 2011] monitoring of events via system or library call traces. This information could also serve as a useful input for future forensics analyses. An exciting workaround to mitigate the damage for virtual machine attacks is represented by *ReVirt* [Dunlap et al. 2002] that makes use of virtual machine logs to replay a system's execution before, during, and after an attacker compromises it. Further suggestions for countermeasures can be found in PCI Security Standards Council [2011].

12. SUMMARY

In this survey, we provided a thorough overview of security issues in hardware virtualization. We focused on potential vulnerabilities and existing attacks on hardware virtualization platforms, but we also briefly sketched some possible countermeasures. We structured the presentation of vulnerabilities and attacks based on their target, which could be the guest, the host operating system, the hypervisor layer, the management interfaces, and the different networks within a virtual infrastructure. In addition, we presented an adversary model that we believe to be general enough to be used to classify not only existing but also future vulnerabilities and attacks, and we gave a short overview on virtualization detection and identification techniques, because those can be used as a first step of any attack.

One of the lessons that one can learn from this survey is that the number of reported vulnerabilities and attacks on different virtualization platforms is already quite large, and it is expected to further increase in the future due to the increasing complexity of and additional services in those platforms. Given the increasing popularity of using virtualization technologies, and in particular, the proliferation of cloud computing services, it is important to be aware of these security issues and to address them in an appropriate way.

This requires better understanding and continuous research in this domain. In particular, we identify the need for developing security tools tailored for discovering vulnerabilities, as well as for detecting or preventing attacks in virtualized environments. As we previously said, some inspiration may be provided by how security issues are addressed in traditional computing systems, but they must be adapted to the specific properties of virtualized systems; in addition, some issues are very specific to virtualized environments and hence need new solutions. For instance, the impact of an attack on a hypervisor could be very serious because it affects multiple guest operating systems and the services running on top of those. Therefore, vendors of virtualized platforms should pay special attention to the security of their products, system administrators should pay special attention to careful operation and maintenance of virtual infrastructures, and researchers should develop effective tools for detecting and containing such attacks.

ACKNOWLEDGMENTS

The authors would like to thank KÜRT Co. for providing the virtualization infrastructure to test and verify the feasibility of certain attacks and countermeasures. Special thanks go to Sándor Szunyogh for his precious practical advice. We would also like to thank Márk Félegyházi, Peter Szor, Zoltán Micskei, and the anonymous reviewers for their useful comments. Pictograms and icons used for figures are created by Zsombor Kiss (`http://www.kisszsombor.com`).

REFERENCES

ADAMS, K. AND AGESEN, O. 2006. A comparison of software and hardware techniques for x86 virtualization. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 2–13.

ARCANGELI, A., EIDUS, I., AND WRIGHT, C. 2009. Increasing memory density by using ksm. In *Proceedings of the Linux Symposium (OLS'09)*. 19–28.

AZAB, A. M., NING, P., WANG, Z., JIANG, X., ZHANG, X., AND SKALSKY, N. C. 2010. Hypersentry: Enabling stealthy in-context measurement of hypervisor integrity. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*. 38–49.

AZAB, A. M., NING, P., AND ZHANG, X. 2011. SICE: A hardware-level strongly isolated computing environment for x86 multi-core platforms. In *Proceedings of the 18th ACM Conference on Computer and communications security (CCS'11)*. 375–388.

BELLARD, F. 2005. QEMU, a fast and portable dynamic translator. In *Proceedings of the USENIX Annual Technical Conference, FREENIX Track*. 41–46.

BELLARD, F. 2008. QEMU. `http://wiki.gemu,org/indx.html`.

BENCSÁTH, B., PÉK, G., BUTTYÁN, L., AND FÉLEGYHÁZI, M. 2011. Duqu: A stuxnet-like malware found in the wild. Tech. rep., BME CrySyS Lab.

BERT. 2010. Which ESX version am I running on ? `http://virtwo.blogspot.hu/2010/04/which-esx-version-am-i-running-on.html`.

BUGIEL, S., NÜRNBERGER, S., PÖPPELMANN, T., SADEGHI, A.-R., AND SCHNEIDER, T. 2011. Amazonia: When elasticity snaps back. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*. 389–400.

BUTLER, J. M. AND VANDENBRINK, R. 2009. IT audit for the virtual environment. SANS Whitepaper. `http://www.sans.org/`.

CHEN, X., ANDERSEN, J., MAO, Z. M., BAILEY, M., AND NAZARIO, J. 2008. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *Proceedings of the 38th Annual IEEE International Conference on Dependable Systems and Networks (DSN '08)*. 177–186.

CISCO AND VMWARE. 2009. DMZ virtualization using VMware vSphere 4 and the Cisco Nexus 1000V virtual switch. `http://www.VMWare.com/resources/techresource/10035`.

CITRIX. 2007. XenServer. `http://www.citrix.com/English/ps2/products/product.asp?contentID\=683148`.

DAGON, D., QIN, X., GU, G., LEE, W., GRIZZARD, J., LEVINE, J., AND OWEN, H. 2004. Honeystat: Local worm detection using honeypots. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*. 39–58.

Dinaburg, A., Royal, P., Sharif, M., and Lee, W. 2008. Ether: Malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*. 51–62.

Dolev, D. and Yao, A. C. 1981. On the security of public key protocols. Tech. rep., Stanford University, CA.

Dunlap, G. W., King, S. T., Cinar, S., Basrai, M. A., and Chen, P. M. 2002. Revirt: Enabling intrusion analysis through virtual-machine logging and replay. *SIGOPS Oper. Syst. Rev. 36*, 211–224.

Dwivedi, H. 2003. Storage security. `http://www.blockhat.com/presentations/bh-usa-03/bh-us-03-Dwivedi.pdf`.

Dwivedi, H. 2004. Insecure IP storage networks. `http://blackhat.com/presentations/bh-usa-04/bh-us-04-Dwivedi.pdf`.

Dwivedi, H. 2005. iSCSI Security - Insecure SCSI. `http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-Dwivedi-update.pdf`.

Engler, D. R., Kaashoek, M. F., and O'Toole, Jr., J. 1995. Exokernel: An operating system architecture for application-level resource management. *SIGOPS Oper. Syst. Rev. 29*, 251–266.

Ferrie, P. 2006. Attacks on virtual machine emulators. Symantec Whitepaper. `http://www.symantec.com/avcenter/refrence/virtual_Machine_Threats.pdf`.

Ferrie, P. 2007. Attacks on more virtual machines emulators. In *Proceedings of the 16th USENIX Security Symposium*.

Franklin, J., Luk, M., McCune, J. M., Seshadri, A., Perrig, A., and Doorn, L. V. 2008. Remote detection of virtual machine monitors with fuzzy benchmarking. *ACM SIGOPS Oper. Syst. Rev. Special Edition on Computer Forensics 42,* 3, 83–92.

FreeBSD. 2000. FreeBSD Handbook - Jails. `http://www.freebsd.org/doc/handbook/jails.html`.

Garfinkel, T., Adams, K., Warfield, A., and Franklin, J. 2007. Compatibility is not transparency: VMM detection myths and realities. In *Proceedings of the 11th Workshop on Hot Topics in Operating Systems (HotOS-XI)*.

Gupta, D., Lee, S., Vrable, M., Savage, S., Snoeren, A. C., Varghese, G., Voelker, G. M., and Vahdat, A. 2010. Difference engine: Harnessing memory redundancy in virtual machines. *Commun. ACM 53*, 85–93.

Holz, T. and Raynal, F. 2005. Detecting honeypots and other suspicious environments. In *Proceedings of the 6th IEEE Information Assurance Workshop*.

iSecLab. 2007. Analysing unknown binaries. `http://anubis.iseclab.org/`.

Kato, K. 2003. VMware backdoor commands. `http://chitchat.at.infoseek.co.jp/vmware/backdoor.html`.

Keller, E., Szefer, J., Rexford, J., and Lee, R. B. 2010. Nohype: Virtualized cloud infrastructure without the virtualization. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA'10)*. 350–361.

King, S. T., Chen, P. M., min Wang, Y., Verbowski, C., Wang, H. J., and Lorch, J. R. 2006. Subvirt: Implementing malware with virtual machines. In *Proceedings of the IEEE Symposium on Security and Privacy*. 314–327.

Klein, A. 2010. Detecting virtualization over the Web with IE9 (platform preview) and semi-permanent computer finger printing and user tracking in IE9 (platform preview). `http://www.trusteer.com/sites/default/files/VM_Detection_and_Temporary_User_Tracking_in_IE9_Platform_Preview.pdf`.

Klein, T. 2008. ScoopyDoo - VMware Fingerprint Suite. `http://www.trapkit.de/research/vmm/scoopydoo/index.html`.

Kortchinsky, K. 2009. Cloudburst: A VMware guest to host escape story. `http://blackhat.com`.

KVM. 2007. Kernel-based Virtual Machine. `http://www.linux-kvm.org/page/Main_Page`.

Lee, R. B., Kwan, P. C. S., McGregor, J. P., Dwoskin, J., and Wang, Z. 2005. Architecture for protecting critical secrets in microprocessors. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA'05)*. 2–13.

Leslie, I., Mcauley, D., Black, R., Roscoe, T., Barham, P., Evers, D., Fairbairns, R., and Hyden, E. 1996. The design and implementation of an operating system to support distributed multimedia applications. *IEEE J. Select. Areas Commun. 14,* 17, 1280–1297.

Li, C., Raghunathan, A., and Jha, N. K. 2010. Secure virtual machine execution under an untrusted management os. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing (CLOUD'10)*. 172–179.

Lie, D., Thekkath, C., Mitchell, M., Lincoln, P., Boneh, D., Mitchell, J., and Horowitz, M. 2000. Architectural support for copy and tamper resistant software. *SIGPLAN Not. 35*, 168–177.

McAfee. 2011. McAfee DeepSAFE. `http://www.mcafee.com/us/solutions/mcafee-deepsafe.aspx`.

McCune, J. M., Li, Y., Qu, N., Zhou, Z., Datta, A., Gligor, V., and Perrig, A. 2010. TrustVisor: Efficient TCB reduction and attestation. In *Proceedings of the IEEE Symposium on Security and Privacy*.

Microsoft. 2006. Virtual PC. `http://www.microsoft.com/windows/virtual-pc/default.aspx`.

Microsoft. 2008. Hyper-V. `http://technet.microsoft.com/en-us/windowsserver/dd448604`.

Microsoft. 2012. Virtual PC - Configure BIOS for hardware assisted virtualization (HAV) PCs. `https://www.microsoft.com/windows/virtual-pc/support/configure-bios.aspx`.

Oberheide, J., Cooke, E., and Jahanian, F. 2008. Empirical exploitation of live virtual machine migration. `http://www.eecs.mich.edu/fjgroup/pubs/blackhat08-migration.pdf`.

Offensive Computing. 2003. `http://www.offensivecomputing.net/`.

Omella, A. A. 2006. Methods for virtual machine detection. `http://www.s21sec.com/descargas/vmware-eng.pdf`.

OpenVZ. 2005. Open VirtualiZation. `http://wiki.openvz.org/Main_Page`.

Oracle. 2004. Oracle Solaris Containers. `http://www.oracle.com/technetwork/server-storage/solaris/containers-169727.html`.

Oracle. 2007. VirtualBox. `http://www.virtualbox.org/`.

Ormandy, T. and Tinnes, J. 2009. Invalid PF exception code in VMware can result in guest privilege escalation. `http://seclists.org/fulldisclosure/2009/Oct/330`.

PCI Security Standards Council. 2011. Information Supplement: PCI DSS virtualization guidelines. `http://www.pcisecuritystandards.org/`.

Pék, G., Bencsáth, B., and Buttyán, L. 2011. nether: In-guest detection of out-of-the-guest malware analyzers. In *Proceedings of the 4th European Workshop on System Security (EuroSec'11)*. 3:1–3:6.

Peng, T., Leckie, C., and Ramamohanarao, K. 2007. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv. 39*.

Quist, D. and Smith, V. 2006. Further down the VM spiral - detection of full and partial emulation for IA-32 virtual machines. In *Proceedings of the Defcon 14*.

Quist, D. and Smith, V. 2008. Detecting the presence of virtual machines using the local data table. `http://www.offensivecomputing.net/files/active/0/vm.pdf`.

Raffetseder, T., Kruegel, C., and Kirda, E. 2007. Detecting system emulators. In *Proceedings of the Information Security Conference (ISC'07)*.

Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. 2009. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09)*. 199–212.

Rutkowska, J. 2004. Red Pill... or how to detect VMM using (almost) one CPU instruction. `http://www.securiteam.com/securityreviews/6Z00H20BQS.html`.

Rutkowska, J. 2006. Subverting Vista Kernel for fun and profit. `http://blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf`.

Sailer, R., Valdez, E., Jaeger, T., Perez, R., Doorn, L. V., Griffin, J. L., Berger, S., Sailer, R., Valdez, E., Jaeger, T., Perez, R., Doorn, L., Linwood, J., and Berger, G. S. 2005. Shype: Secure hypervisor approach to trusted virtualized systems. In *IBM Res. Rep. RC23511*.

Scarfone, K., Souppaya, M., and Hoffman, P. 2011. Guide to security for full virtualization technologies: Recommendations of the National Institute of Standards and Technology. `http://csrc.nist.gov/publications/nistpubs/800-125/SP800-125-final.pdf`.

Scope, A. 2008. Virtualization: State of the art. `http://scope-alliance.org/technical-documents`.

Seshadri, A., Luk, M., Qu, N., and Perrig, A. 2007. Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity oses. In *Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP'07)*. 335–350.

Shelton, T. 2005. VMware natd heap overflow. `http://lists.grok.og/uk/pipermail/full-disclosure/2005-December/040442.html`.

Sidiroglou, S., Ioannidis, J., Keromytis, A. D., and Stolfo, S. J. 2005. An emailworm vaccine architecture. In *Proceedings of the 1st Information Security Practice and Experience Conference*.

Smith, J. and Nair, R. 2005. The architecture of virtual machines. *Computer 38,* 5, 32–38.

Song, D., Brumley, D., Yin, H., Caballero, J., Jager, I., Kang, M. G., Liang, Z., Newsome, J., Poosankam, P., and Saxena, P. 2008. BitBlaze: A new approach to computer security via binary analysis. In *Proceedings of the 4th International Conference on Information Systems Security.* Keynote invited paper.

Srinivasan, D., Wang, Z., Jiang, X., and Xu, D. 2011. Process out-grafting: An efficient "out-of-vm" approach for fine-grained process execution monitoring. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*. 363–374.

STEINBERG, U. AND KAUER, B. 2010. Nova: A microhypervisor-based secure virtualization architecture. In *Proceedings of the 5th European Conference on Computer Systems (EuroSys'10)*. 209–222.

SUH, G. E., O'DONNELL, C. W., SACHDEV, I., AND DEVADAS, S. 2005. Design and implementation of the aegis single-chip secure processor using physical random functions. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA'05)*. 25–36.

SUZAKI, K., IIJIMA, K., YAGI, T., AND ARTHO, C. 2011. Memory deduplication as a threat to the guest os. In *Proceedings of the 4th European Workshop on System Security (EuroSec'11)*. 1:1–1:6.

SZEFER, J., KELLER, E., LEE, R. B., AND REXFORD, J. 2011. Eliminating the hypervisor attack surface for a more secure cloud. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*. 401–412.

TAKEMURA, C. AND CRAWFORD, L. S. 2007. *The Book of Xen, A Practical Guide for the System Administrator*. No Strach Press, San Francisco, CA.

VIRTUALIZATIONREVIEW. 2009. KVM: Bare-metal hypervisor? http://virtualizationreview.com/Blogs/Mental-Ward/2009/02/KVM-BareMetal-Hypervisor.aspx.

VMWARE. 1999. Workstation. http://www.vmware.com/products/workstation/.

VMWARE. 2006. VMware ESX Server 3 802.1Q VLAN Solutions. http://www.VMWare.com/resources/techresources/412.

VMWARE. 2007. ESXi. http://www.vmware.com/products/vsphere-hypervisor/overview.html.

VMWARE. 2008a. SAN system design and deployment guide. http://www.VMWare.com/resources/techresources/772.

VMWARE. 2008b. Virtual networking concepts. http://www.VMWare.com/resources/techresources/997.

VMWARE. 2009a. ESX monitor modes. http://communities.vmware.com/docs/DOC-9882/.

VMWARE. 2009b. Network segmentation in virtualized environments. http://www.VMWare.com/resources/techresources/1052.

VMWARE. 2009c. VMware ESX and VMware ESXi. http://www.vmware.com/files/pdf/VMware-ESX-and-vmware-ESXi-DS-EN.pdf.

VMWARE. 2009d. What is new in VMware vSphere^{TM}4: Storage. http://www.VMWare.com/files/pdf/storage-with-VMWare-vSphere.pdf.

VMWARE. 2010. VMWare vStorage virtual machine file system - technical overview and best practices. http://www.VMWare.com/resources/techresources/10110.

VMWARE. 2011a. Performance best practices for VMware vSphere 5.0. http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.0.pdf.

VMWARE. 2011b. VMWare vSphere 4.1 security hardening guide. http://www.VMWare.com/resources/techresources/10198.

VMWARE. 2012. Configuring disks to use VMware Paravirtual SCSI (PVSCSI) adapters. http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1010398.

WALDSPURGER, C. A. 2002. Memory resource management in vmware esx server. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI'02)*.

WANG, Z. AND JIANG, X. 2010. Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'10)*. 380–395.

WIRESHARK. 2006. http://wiki.wireshark.org/Gratuitous_ARP.

WOJTCZUK, R. 2008a. Adventures with a certain Xen vulnerability (in the PVFB backend). http://hax.tor.hu/read/xen/xenfb-adventures-10.pdf.

WOJTCZUK, R. 2008b. Xen Owning Trilogy. http://virtualization.com/tag/backhat/.

WOJTCZUK, R. AND RUTKOWSKA, J. 2009. Attacking Intel Trusted Execution Technology. http://invisiblethingslab.com.

WOJTCZUK, R. AND RUTKOWSKA, J. 2011. Following the white rabbit: Software attacks against Intel®VT-d technology. http://invisiblethingslab.com/.

WOJTCZUK, R., RUTKOWSKA, J., AND TERESHKIN, A. 2009. Another way to circumvent Intel Trusted Execution Technology. http://invisiblethingslab.com/.

YAU, D. K. Y., LUI, J. C. S., LIANG, F., AND YAM, Y. 2005. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Trans. Netw. 13*, 29–42.

ZOVI, D. A. D. 2006. Hardware virtualization rootkits. http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf.

Received August 2011; revised March 2012; accepted May 2012