Virtual Machine Isolation

A Survey on the Security of Virtual Machines

Jithin R and Priya Chandran

National Institute of Technology, Calicut, Kerala, India

Abstract. The popularity and widespread adoption of cloud computing has resulted in extensified and intensive use of virtualization technology. Virtualization technology allows the sharing of the same physical resources among several users. This enables the consolidation of servers and a multitude of user machines into a very small set of physical servers, by replacing the physical machines with virtual machines, running on the same physical servers. Consequently, several users work on and store their data in the same physical platform. A software layer is used to enable the sharing of hardware between the different users. Understandably, this leads to apprehensions about the security of their data and working environment for the users, as these are situated only one software layer apart from those belonging to the other users. Centralized storage and centralized computing thus naturally raise the question of security of user's data, and motivate studies on how data security could possibly be compromised. This article surveys the security concerns in virtualization technology. It includes a study of different attacks in the context of virtualization, and logically organizes them in different categories. Where available, the patches to the attacks are also included in the survey. A special focus of the survey is on hardware limitations to support virtualization, and the conclusion drawn is that hardware limitations of different types are the root cause of most of the security issues.

Keywords: Virtualization technology, Virtual Machines, Virtualization Security

1 Introduction

In the computing scenario, virtualization is the creation of virtual versions of some real objects such as hardware and software. Logical partitions of real objects are made, to create instances of virtual objects. A well-known example is a hard disk drive. Each partition of the hard disk in an operating system is the logical copy of original hard disk.

Two main types of virtualization are *hardware virtualization* and *software virtualization* [1]. *Virtualization software* runs on the real object (i.e., the hardware or software) to be shared. The virtualization software makes multiple virtual objects that look exactly the same real object.

This article focusses on hardware virtualization technology, aimed at partitioning physical machines (computers) into several logical machines. The virtualization software used to create logical machines is popularly termed as *Hypervisor* and each logical machine is referred to as *Virtual Machine (VM)*, in this area. An operating system installed on the virtual machine is known as *Guest Operating System*.

Several practical situations, like surges in the demand for services offered by the virtual machine, or the maintenance of physical servers hosting the virtual machines, may warrant the transfer of a virtual machine from one physical platform to another. This is made possible through the introduction of *Virtual Machine Migration* [2].

Virtual Machine (VM) migration is the process of transferring a virtual machine from one physical machine to another. VM Migration can be done either in active or in passive state of the virtual machine [2]. Migration in *passive mode* is defined as moving a VM from one physical machine to another when the VM is turned off. Migration in *active mode* is defined as moving a VM from some physical hardware to another while the VM is running and without interrupting the services running in VM [2]. Active mode migration is called *Live Migration*. Fig.1 serves to illustrate the process of migration of VMs.

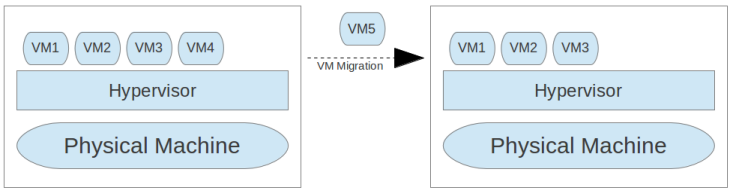

Fig. 1. Virtual Machine Migration Process

Though virtualization technology and technologies realizing the concept of virtual machine migration help to achieve the optimum utilization of physical resources, they spawn several security issues, which are yet to be studied fully and remedied, thus making the system vulnerable to attacks of various kinds. In this paper, various vulnerabilities and attacks in virtualization technology are studied. Generic and specific solutions to these issues, and recent advancements in hardware to overcome these problems are also reported.

The rest of this paper is organized as follows. Section 2 explains the different types of attacks that could exist in virtual machines, as reported in literature. The architectural limitations and the recent advancements in hardware to support efficient virtualization are explained in Section 3. The last section concludes the article with an analysis on the survey.

2 Attacks

From the perspective of the operating system, VMs and physical machines are identical. The perspective of this paper is that the virtual machines should be

as secure as physical machines, i. e., there should not be any security vulnerabilities in virtual machines, arising due to virtualization. This implies that virtual machines as isolated logically from each other as are different physical machines, and the only communication mechanism between VMs should be through networking, as in the case of physical machines. This is termed *complete isolation*. Complete isolation is not achieved with existing virtualization technology, as our survey reveals. There are several attacks possible in virtual machines, arising due to the vulnerability introduced by the lack of virtual machine isolation [3] [4] [5] [6] [7] [8] [9] . This paper categorizes them as *Covert Channel Attacks, Malware Attacks* and *Attacks in Migration*. The rest of this section reports on each of these categories of attacks.

2.1 Covert Channel Attacks

Covert channels [3] are the secret channels that exist between two supposedly isolated environments, such as VMs. The existence of covert channels reveals cracks in the isolation, the basic security paradigm, of virtual machines.

Xiao et al. report the construction of a covert channel between virtual machines by exploiting the memory de-duplication feature in the virtualization software i.e., the hypervisor [3]. Virtual machines communicate with the hypervisor through hypercalls, and the hypervisor manages these calls through an event table. *Event-channel* hypercalls and *grant table* hypercalls have been exploited to create a covert channel in [10]. The hypercalls were used to create a shared memory to communicate between two virtual machines. This is a straight covert channel between two virtual machines. Moreover, in [4], a hidden covert channel named *Shared Memory Covert Timing Channel* (SMCTC) was constructed inside such a shared channel by fixing *read* and *write* time intervals. It can be observed that these two covert channels leveraged memory vulnerabilities.

The cache subsystem also has its share of vulnerabilities, open for exploiters. In [4], a covert channel is demonstrated by using the L1 instruction cache as the channel for covert communication between virtual machines. The disaster potential of this covert channel is further emphasized by using the channel to construe an attack to extract the ElGamal decryption key [11] from a victim virtual machine[4]. L2 caches can also be exploited, as introduced in [12], and studied further in several works. [12] quantified the L2 cache based covert channels and assessed the damage potential of L2 cache based covert channels.

Even the silent work horse, the CPU, can become the carrier of covert information. A CPU based Covert Channel between VMs (CCCV) was created by using the CPU work load as the medium of covert communication [13], to work on a single-core processor. Covert Channels using Core-alternation (CCCA) have been demonstrated on virtual platforms with multicore processors [6].

Network resources were also not spared, and have been leveraged to create covert channels between virtual machine. The FCFS packet scheduling system is used as the covert medium between virtual machines in [14]. [15] shows another covert channel, which uses the IP identification field of the IP datagram header

as the cover medium, thus exploiting network protocol features to provide covert channels.

Strong isolation between virtual machines at various levels of computer architecture is the need of the day. to prevent covert channels. A model was proposed in [16], for improving isolation of virtual machines. Much research is required into the potential covert channels and integrated approaches to complete isolation of VMs.

The next category of attacks is through malware, which are malicious programs carrying out illegal and unwarranted activities in the system. Though malware are a general threat to computer systems, there are special malware designed exclusively for VMs. They exploit features of VMs or the virtualization environments and technology to carry out their damaging activities, and hence affect only the virtual machines and applications running on them. In the next sub section, two types of VM based malware, named as *VM-aware malware*, and *Hypervisor level rootkits*, are described.

2.2 Malware Attacks

VM-aware malware can identify whether they are running on a virtual machine or real machine [7]. The malware detects the presence of a virtual environment using counter based detection[7] and is possible only on processors with two or more cores. [7] introduces a technique to prevent counter-based detection attack using the Inter Processor Interrupt (IPI) signal.

Virtual Machine Based Rootkit (VMBR) [8] [17] is a hypervisor level rootkit which mimics the structure of a hypervisor. The malware gets installed above the hardware as a hypervisor and the existing operating system is moved to a virtual machine in the hypervisor. The malware works from this hypervisor without its presence getting detected by the operating system, as it runs on a virtual machine running on the spurious hypervisor. VMBR is also called hyperjacking.

Subvirt [8] is a rootkit developed jointly by Microsoft and University of Michigan researchers as an academic example of virtual machine based rootkit. Blue pill [17] is malware based on Intel's x86 virtualization and requires Intel VT-x or AMD-v virtualization support [17]. [18] describes the source code for presenting a minimal hypervisor framework for a rootkit.

Hypervisor level rootkit attack are rendered feasible because systems with virtualization enabled CPUs, when used in the the absence of hypervisors, have just the normal operating systems running above the virtualization enabled CPU [19], and the virtualization capability is leveraged by the pseudo-hypervisors. Gaurdhype [20] is a proposed solution to this problem. Gaurdhype is a lightweight hypervisor which monitors the virtual machine control structures (VMCS)[19]. VMCS is the central part of hardware-assisted virtualization architecture. VMCS contains the states of each virtual machine and hypervisor. By monitoring the VMCS values, the presence of VMBR is determined.

A new approach named *Ether* to analyze the malware in a virtual environment is given in [21]. The idea is to use Intel Hardware Virtualization technology

extensions [22] [21]. Due to the architectural limitations in Intel hardware virtualization technology, the malware analysis done by Intrusion Detection Software, Intrusion Prevension Software etc can be detected by the malware itself. It is claimed that these limitations do not exist in AMD hardware virtualization technology [21].

Malware can also help the attacker to create unauthorized access to virtual machines, which violate the isolation property of virtual machines. Malware has been a threat to the operating systems running in physical machines and now prove to be a threat to the operating systems running in virtual machines also. Finding generic and efficient solutions to prevent malware attacks is also a very important research issue.

The third category of attacks is based on a exploiting powerful feature provided by virtualization technology, namely virtual machine migration, which allows virtual machines to be moved from one physical machine to another for logistic or similar reasons. The following subsection reviews various security issues in virtual machine migration.

2.3 Attacks in Migration

The starting point for several attacks in the virtualization environment is the detection of the virtual machine migration process. Detection of the virtual machine live migration has been demonstrated in [23] using ICMP packets. [24] provides a comprehensive survey of vulnerabilities leading to attacks in Live Migration. They are shortly listed here as:

Inappropriate access control policies Due to inappropriate access control policies any virtual machine can initiate migration, terminate the migration and become susceptible to man-in-the-middle attacks. The attacker can utilize these loop holes in access control to migrate the malicious VM to a hypervisor. The malicious VM in a hypervisor can obviously harm the hypervisor and other VMs [24]

Unprotected transmission channel Unprotected transmission channel, between the guest and host physical machines involved in the migration process helps the attacker to do passive and active attacks [24]

Loopholes in migration module Loopholes in migration module, like stack-overflow, heap-overflow and integer-overflow make the migration further vulnerable [24]

Oberheide et al. developed a tool named *Xensploit* [9] to carry out man-in-the-middle attacks on virtual machine migration. Xensploit was used to modify the memory segment, specifically the *sshd* memory segment, in such a way that the *sshd* authentication was be bypassed.

Solutions for preventing the attacks in live migration have been suggested in [24] for secure live migration. These are

Virtual Local Area Network Virtual Local Area Networks (VLAN) have been proposed for secure migration traffic. VLAN is an invisible network created

inside a public Network. VLAN is independent of physical location and is created by tagging the packets with the tag-ID of corresponding VLAN[24]

Network Security Engine Network Security Engine is a security module proposed to be included inside the hypervisor, and contains protection mechanisms like firewall, IDS and IPS. The goal of the network security engine is to prevent intrusions to the virtual network from outside [24]

Role Based Migration In a role based migration process, a role based technique using the Trusted Platform Module hardware to find a cryptographically trusted remote hypervisor is used for secure migration. Role based migration process helps to establish policies for deciding who can start migration and to which host and so on[24].

Trusted platform module(TPM) functionality can be leveraged in several other ways as well for secure virtual machine migration. TPM helps to identify the presence of unauthorized access to the system. [25] created a software module named vTPM inside the hypervisor, to share the TPM functionalities with the OS running in each virtual machine. For each virtual machine, an instance of TPM module (vTPM) is created. However, [26] points out that as this implementation is completely inside the software, it cannot protect the cryptographic secrets in every operating system. Due to these limitations Stumpf [26] suggests a different scheme to share the TPM device with the virtual machines.

The observation from our survey on live migration security is that there are several authentication issues, as well as passive and active attacks, that exploit virtual machine live migration. It happens mainly due to the lack of a secure live migration protocol. A secure live migration protocol should provide the following essential security features to a VM Migration process.

- Protected transmission channels
- Integrity of migration data
- Entity authentication

Existing literature reveals only a few secure live migration protocol proposals [26] [27] for the virtual machine migration service. Hence it is clear that researchers on virtualization security should to give more attention to live migration security, as it is a crucial issue.

The above survey points out that the closeness of the hypervisors to the hardware, and the placement of the virtual machine operating systems at a higher software layer is a primary reason for the failure of conventional security mechanisms and existence of hypervisor based malware. Hardware oriented or hardware level solutions seem intuitively possible for securing virtual machines. Hence our survey zooms over to literature on vulnerabilities in virtual machines at various hardware levels. The following section reviews some limitations in different hardware to support virtualization.

3 Architectural limitations and consequences

It was seen in the above sections that covert channel attacks exploit the vulnerabilities in CPU, memory and network. Malware uses the vulnerabilities in

memory to attack the system. Network vulnerabilities are leveraged by the attacks in VM migration. Literature on hardware devices and features like the CPU, memory subsystem and networks was studied to find the reason for vulnerabilities leading to the mentioned security issues. Architectural limitations in different hardware to support hardware virtualization are the topic of focus in the succeeding subsections.

3.1 Limitations in the CPU Ring level [28]

In an Intel or AMD processor, hardware virtualization technology facilitates the sharing of the x86 processor among multiple virtual machines. Virtualization technology in Intel or AMD is generally known as x86 virtualization. [1] reports the existence of several vulnerabilities and hardware limitations in x86 virtualization .

IA-32 architecture provides hardware level protection using a 2-bit privilege level called *CPU Ring levels* [28] or *Privilege levels* [1]. Ring level zero for the most privileged instructions, and three for the least privileged instructions. In the non-virtualized environment, an operating system runs its instruction in privilege level 0 and applications run in privilege level 3. In a virtualized environment, the hypervisor runs in level 0. So the guest operating system(on a virtual machine) is forced to run in ring levels other than 0. If a guest operating system also runs in privilege level 0, the guest OS could control the resources which are to be controlled by the hypervisor. The ambiguities that arise in ring level assignments create security threats like *Ring De-privileging, Ring Aliasing, Address Space Compression, Ring Compression* [28].

These security challenges in x86 hardware virtualization have been solved using *hardware-assisted* virtualization technology. Intel and AMD added some features to the existing x86 hardware to solve these security issues. The x86 hardware virtualization technology augmented with these features (from Intel or AMD) is known as Hardware-assisted virtualization technology. Hardware-assisted virtualization technology from AMD is known as AMD-V [29], and that from Intel is known as Intel VT [22].

Virtualization of CPU calls for the execution of system-level and user-level instructions from the guest operating systems. This can be achieved by virtualizing the Instruction Set Architecture (ISA). The next subsection reviews the hardware limitations in ISA virtualization.

3.2 Limitations in ISA Virtualization

In a normal environment an operating system accesses the CPU through the interface named *instruction set architecture*. In a virtual environment, the guest operating system accomplish this through a virtual ISA. ISA virtualization is the technique used in virtualization technology to share CPU, through virtual ISAs.

In 1972, Goldberg stated the sufficient conditions for efficient ISA virtualization [1]. Considering the execution style, he arranged the instructions into 3 classes.

Non privileged instructions These instructions run in the user mode and can be executed directly in any mode.

Privileged instructions These instructions trap when the machine is in the user mode and do not trap when machine is in the system mode. The privileged instructions should always trap from user mode.

Sensitive instructions These instructions can change the system configuration. They are of two types, namely, *control sensitive* and *behavior sensitive* instructions. Control sensitive instructions can change the configurations of system resources, and behavior sensitive instructions produce the result depending on the current configuration of resources.

Goldberg showed formally that for efficient virtualization of ISA, a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions, which implies that, for efficient virtualization, all sensitive instructions should trap in user mode [1].

Satisfying the Goldberg conditions in ISA virtualization will allow the virtual machines to execute non-privileged instructions directly on the hardware, and thereby improve the VM performance [1]. All other privileged and sensitive instructions will move the control of CPU from virtual machine to hypervisor, enforcing VM isolation [1]. Satisfying Goldberg conditions simultaneously achieves isolation and performance for virtual machines at CPU level.

In the year 2000, John Scott Robin et al. [30] analyzed the Intel Pentium processor's ability to support secure virtual machine monitor. His analysis shows that out of 250 instructions in x86 architecture, there are 17 instructions which did not meet the requirement of Goldberg condition of efficient virtualization. It was because these 17 instructions are sensitive instructions but not privileged and they can be executed from the user mode. Smith [1] named these instructions as *Critical Instructions*. Critical instruction are sensitive instructions but not privileged. According to Goldberg, for efficient virtualization there should not be any critical instructions.

Our survey reveals that even a study of existing articles, several books and architecture software developer's manuals related to hardware-assisted virtualization technology [22] [29] [28] [31] does not clarify whether critical instructions exist in the discussed technology. Thus, a detailed practical analysis on the Intel and AMD processor architecture with latest hardware-assisted virtualization technology have to be done to confirm the existence of critical instructions.

In a physical machine, any hardware other than the CPU is considered as an input/output device (I/O device). Virtualization of I/O devices is the next stage of virtualization. Accomplishing I/O virtualization requires overcoming some hardware limitations, as explained in the next sub section.

3.3 Limitations in I/O Virtualization

I/O virtualization is the technique used to share the I/O devices, like storage devices or memory devices among virtual machines. Improper sharing of IO devices (inefficient I/O virtualization) may introduce security vulnerabilities like covert channels [3]. Efficient I/O virtualization is required to improve security of virtual machines. The requirements of I/O virtualization are *Platform independence to operating systems*, *Isolation of I/O devices* and *High Performance that is equivalent to a non-virtualized environment*

Earlier I/O virtualization was implemented through two techniques named *emulation* [1] and *paravirtualization* [20]. The main advantage of emulation was that it supported a wide range of unmodified guest operating systems. But emulation requires additional transactions between hypervisor and guest OS (Virtual Machine), which increases the complexity of hypervisor, resulting in lower performance [20]. Paravirtualization shows an improved performance over emulation due to reduced interaction between the hypervisor and the guest OS [20]. Paravirtualization requires modification to guest OS. This is a major drawback of paravirtualization. The number of modified operating systems is small yet

The security requirement *isolation of I/O devices* implies that I/O devices allocated to a virtual machine should not be allocated to or accessed by any other virtual machines at any cost until unallocated. Emulation and Paravirtualization cannot satisfy isolation.

Intel satisfied all those requirements with the introduction of a new technology named Intel VT-d [20]. Intel VT-d is the hardware-assisted virtualization technology for virtualizing the I/O devices. The main design goal of Intel VT-d was to support a wide range of unmodified guest OS with improved security and performance equivalent to that in a non-virtualized environment. Intel VT-d is the major technology of Intel Hardware Virtualization Technology suite, which eliminates various security challenges in I/O virtualization and provides platform independence to operating systems [22].

Intel VT-d architecture is a generalized IOMMU architecture that provides the system software with multiple direct memory access(DMA) protection domains [20]. Intel VT-d provides an improved performance through the direct assignment of I/O devices to virtual machines. Direct assignment is possible through direct memory access technique and provides isolation by mapping I/O devices to a protection domain [20]. Protection domain is a subset of physical memory allotted to a Virtual Machine. One or more I/O devices are allotted to a protection domain.

Our survey reveals that paravirtualization outperformed the initial release of hardware-assisted full virtualization for workloads that perform input/output operations, creating processes, or switch contexts rapidly [32]. We also learnt that IOMMU does not support the virtual machine live migration [33]. These are the major drawbacks of hardware-assisted virtualization technologies. Hence much improvement in hardware assisted virtualization technology is needed through active research, to make it adaptable in general.

4 Conclusion

In this paper, the security issues arising from hardware virtualization, specifically virtualization technology enabling the co-existence of several virtual machines the same physical platform, have been studied, analyzed and reported.

The *attacks* on virtual machines, which potentially result in the compromise of the security of the virtual machine user's data, have been classified into three types, namely the *covert channel* attacks, *virtual machine migration* attacks and *malware* attacks. The three types of attacks leverage the CPU, memory and network respectively, to attack the virtual machines, and thereby violate the isolation property of virtual machines at different hardware levels. The study also includes hardware limitations in CPU and I/O devices to extend support for isolated virtual machines.

It is concluded from the survey that several security issues and hardware limitations exist at the different architectural levels of virtualization technology that compromise the isolation of virtual machines. Only a few of the security issues and hardware limitations have proposed solutions in literature. Covert channels [3], Live migration attacks [33], and the presence of critical Instructions [1] are some of the unresolved issues.

It is inferred from our analysis that each architectural level problem has a general solution. For example, ring level issues in processor can be solved with efficient virtual ring level creation and the critical instruction issue can be solved with efficient ISA virtualization. The implication is that every processor level issue can be solved with efficient processor virtualization, and similarly every memory level security issue can be solved with protected virtual memory, and the network level security issues can be solved with a secure migration protocol. To summarize, our inference is that every security issue can be solved by providing the required solution at the corresponding architectural level. An overview of the solution requirements for assuring the security of virtualization is summarized in Table 1. We conclude that truly isolated virtual machines can be created

Table 1. Security of Virtualization enabled Architecture

Architecture	**Security Issues**	**Required Solution**
Processor	- Ring level Problems - Critical Instructions - CPU based Covert channels - VM-aware Malware	Efficient CPU Virtualization
Memory	- Malware Attacks - Memory Covert Channels	Protected Virtual Memory
Network	- Live Migration Attacks - Network Covert Channels	Secure Protocols

by completely providing the required solutions, at different architectural levels. This is an area that requires the urgent and devoted attention of researchers in the computing arena, as the proliferation of virtualization technology without sufficient security guarantees can lead to highly vulnerable situations for the users of virtual machines.

References

[1] J. E. Smith and R. Nair, *Virtual Machines: Versatile Platform for Systems and Processes*. Morgan Kaufmann, 2006.

[2] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

[3] J. Xiao, Z. Xu, H. Huang, and H. Wang, "A covert channel construction in a virtualized environment," in *Proceedings of the 2012 ACM conference on Computer and communications security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 1040–1042. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382318

[4] J. Wu, L. Ding, Y. Wang, and W. Han, "Identification and evaluation of sharing memory covert timing channel in xen virtual machines," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 283–291.

[5] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-vm side channels and their use to extract private keys," in *Proceedings of the 2012 ACM conference on Computer and communications security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 305–316. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382230

[6] Y. Li, Q. Shen, C. Zhang, P. Sun, Y. Chen, and S. Qing, "A covert channel using core alternation," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*. IEEE, 2012, pp. 324–328.

[7] H. Li, J. Zhu, T. Zhou, and Q. Wang, "A new mechanism for preventing hvm-aware malware," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*. IEEE, 2011, pp. 163–167.

[8] S. King and P. Chen, "Subvirt: Implementing malware with virtual machines," in *Security and Privacy, 2006 IEEE Symposium on*. IEEE, 2006, pp. 14–pp.

[9] J. Oberheide, E. Cooke, and F. Jahanian, "Empirical exploitation of live virtual machine migration," in *Proc. of BlackHat DC convention*, 2008.

[10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 164–177.

[11] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *Information Theory, IEEE Transactions on*, vol. 31, no. 4, pp. 469–472, 1985.

[12] Y. Xu, M. Bailey, F. Jahanian, K. Joshi, M. Hiltunen, and R. Schlichting, "An exploration of l2 cache covert channels in virtualized environments," in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, ser. CCSW '11. New York, NY, USA: ACM, 2011, pp. 29–40. [Online]. Available: http://doi.acm.org/10.1145/2046660.2046670

[13] K. Okamura and Y. Oyama, "Load-based covert channels between xen virtual machines," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, ser. SAC '10. New York, NY, USA: ACM, 2010, pp. 173–180. [Online]. Available: http://doi.acm.org/10.1145/1774088.1774125

[14] X. Gong, N. Kiyavash, and P. Venkitasubramaniam, "Information theoretic analysis of side channel information leakage in fcfs schedulers," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 1255–1259.

[15] J. Jaskolka and R. Khedri, "Exploring covert channels," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on.* IEEE, 2011, pp. 1–10.
[16] S. KALEESWARAN, "Managing covert information leaks in xen virtual machine systems," Master's thesis, NIT Calicut, May 2010.
[17] [Online]. Available: http://en.wikipedia.org/wiki/Blue_Pill_%28software%29
[18] M. Myers and S. Youndt, "An introduction to hardware-assisted virtual machine (hvm) rootkits," *White Paper of Crucial Security*, 2007.
[19] M. Carbone, W. Lee, and D. Zamboni, "Taming virtualization," *Security & Privacy, IEEE*, vol. 6, no. 1, pp. 65–67, 2008.
[20] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I. Schoinas, R. Uhlig, B. Vembu, and J. Wiegert, "Intel virtualization technology for directed i/o," *Intel Technology Journal*, vol. 10, pp. 178–192, August 2006.
[21] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: malware analysis via hardware virtualization extensions," in *Proceedings of the 15th ACM conference on Computer and communications security.* ACM, 2008, pp. 51–62.
[22] "Hardware-assisted virtualization technology," Intel, Web Article, 2012 (Retrieved December 10, 2012). [Online]. Available: http://www.intel.in/content/www/in/en/virtualization/virtualization-technology/hardware-assist-virtualization-embedded-technology.html
[23] A. König and R. Steinmetz, "Detecting migration of virtual machines," in *Proceedings of the 10th Würzburg Workshop on IP: Joint ITG, ITC, and Euro-NF Workshop Visions of Future Generation Networks (EuroView2011), Julius-Maximilians-Universität Würzburg, Lehrstuhl für Informatik III*, 2011.
[24] J. Shetty, M. Anala, and G. Shobha, "A survey on techniques of secure live migration of virtual machine," *International Journal of Computer Applications*, vol. 39, no. 12, 2012.
[25] R. Perez, R. Sailer, and L. van Doorn, "vtpm: virtualizing the trusted platform module," in *Proc. 15th Conf. on USENIX Security Symposium*, 2006, pp. 305–320.
[26] F. Stumpf and C. Eckert, "Enhancing trusted platform modules with hardware-based virtualization techniques," in *Emerging Security Information, Systems and Technologies, 2008. SECURWARE'08. Second International Conference on.* IEEE, 2008, pp. 1–9.
[27] B. Danev, R. Masti, G. Karame, and S. Capkun, "Enabling secure vm-vtpm migration in private clouds," in *Proceedings of the 27th Annual Computer Security Applications Conference.* ACM, 2011, pp. 187–196.
[28] G. Neiger, A. Santoni, F. Leung, and R. U. Dion Rodgers, "Intel virtualization technology: Hardware support for efficient processor virtualization," *Intel Technology Journal*, vol. 10, pp. 166–177, August 2006.
[29] A. M. Devices, *Secure Virtual Machine Architecture Reference Manual.* Advanced Micro Devices, May 2005.
[30] J. S. Robin and C. E. Irvine, "Analysis of the intel pentium's ability to support a secure virtual machine monitor," DTIC Document, Tech. Rep., 2000.
[31] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. Martins, A. V. Anderson, S. M. Bennett, A. Kgi, F. H. Leung, and L. Smith, "Intel virtualization technology." IEEE Computer Society, May 2005, pp. 48–56.
[32] K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization," in *ACM SIGOPS Operating Systems Review*, vol. 40, no. 5. ACM, 2006, pp. 2–13.
[33] M. Ben-Yehuda, J. Mason, J. Xenidis, O. Krieger, L. Van Doorn, J. Nakajima, A. Mallick, and E. Wahlig, "Utilizing iommus for virtualization in linux and xen," in *OLS06: The 2006 Ottawa Linux Symposium.* Citeseer, 2006, pp. 71–86.