Virtualization Security Issues and Mitigations in Cloud Computing

S. Rama Krishna and B. Padmaja Rani

Abstract This paper presents various security issues related to hypervisor in cloud. This paper also brings issues possible with a malicious virtual machine running over hypervisor such as exploiting more resources than allocated by VM, stealing sensitive data by bypassing isolation of VM through side channel attacks, allowing attacks to compromise hypervisor. In this paper, we also bring security measures or requirements to be taken and architectures that are needed by hypervisor to handle various security concerns.

Keywords Security · Hypervisors · Cloud computing

1 Introduction

Several enterprises believed cloud to be a platform to fulfil their requirements such as increased scalability, availability, and upfront setup cost, etc. Though web2.0, Internet, distributed computing are technologies that enable cloud computing, in reality virtualization is the key technology to extract the exact sense of utilization maximization of resources. Sharing the resources is possible with virtualization is called to be multi-tenancy. Where the physical resources are virtualized and provided for multiple users to share them. Hypervisor looks at resolving issues of provisioning, de provisioning of virtual machines, their migration and isolated use to share a common physical space by multiple tenants (Fig. 1).

Security is the major concern in the cloud because several users share their data in cloud without noticing their co tenants in the same physical space. In this case

S. Rama Krishna (✉)
Department of Computer Science & Engineering, VRS & YRN College of Engineering & Technology, Chirala 523155, India
e-mail: ccvy.ram@gmail.com

B. Padmaja Rani
Department of Computer Science & Engineering, JNTUCEH, Hyderabad, India
e-mail: padmaja_jntuh@yahoo.co

S.C. Satapathy et al. (eds.), *Proceedings of the First International Conference on Computational Intelligence and Informatics*, Advances in Intelligent Systems and Computing 507, DOI 10.1007/978-981-10-2471-9_12

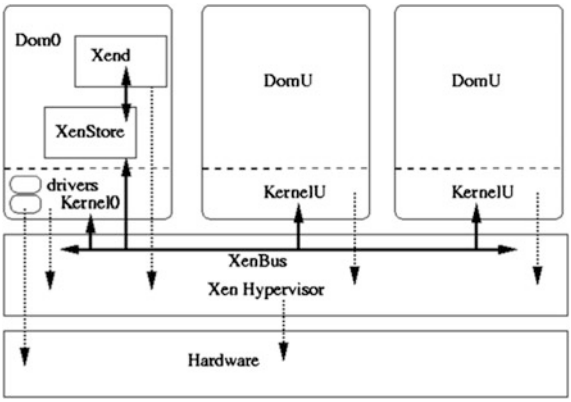

Fig. 1 To show the functionality of virtual machine

hypervisor should ensure a strong VM isolation mechanism because if a VM is vulnerable with its compromised security cause danger to the remaining who share the common spaces. Even denial of service attack is also possible with the compromised VM in hypervisor where the vulnerable VM gets hold of shared resource and cause data leakage also. If the shared resources were hijacked, then co-tenant VMS may be slowed down also.

Remaining of this paper is organized as: Sect. 2 brings Side Channel attacks in Hypervisor. Section 3 brings performance-based attacks that cause target machine slow down. In Sect. 4 security measures needed to protect hypervisor are mentioned. Section 5 presents VM Isolation security issues and counter measures. Section 6 contains conclusions of the derived paper.

2 Side Channel Attacks and Defences

Side channel attack creates opportunity for a co-resident VM to gain access data of other VM without their intervention. It creates a bypassing method to access data. CPU cache, memory, power consumption and network used in extraction of data in side channel attack. Software happenings will be traced by observing behaviour in hardware [1]. Yu et al. [2] takes CPU cache response time to check whether target VM co-resident or not. Cache behaviour is analysed using linear regression of the values collected by load pre-process with cubic spline and load predictor. A malicious VM occupies a major part of CPU cache then targets co-resident by simple data request to it. Then it executes load measuring program over malicious VM for measuring access time of cache. Literature [2] observes and proves that higher cache access time implies more activities by co-resident. The experiment proposal also verified with three VMS sharing resource. Vulnerable VM not only analyses CPU cache access time, but also can get data of the target machine. Literature [3] describe data hijacking of co-resident VM by infecting malware into the software.

This attack targets at obtaining information from the target VM without its notice covertly and it will not leave any trace. For doing this the attacker uses memory bus. Vulnerable VM locks memory bus by sending 1 for issuing atomic CPU instruction, memory latency will increase with execution of CPU atomic instruction. If the memory bus released latency will decrease and it transfers bit 0. Similarly, it also build other side channels.

For exploiting cache contention for manipulating latency times [3], this attack also calculates bandwidth by finding the length of overlapped execution time of malicious VM with target VM. This attack can be defended by making some changes to the scheduler in the hypervisor. To successfully defend, the scheduler can try to limit the overlapping execution times of any two VMS on the system while maintaining an acceptable level of performance. The scheduler should still maintain fairness because it does not know which VM is malicious. In order to maintain an acceptable level of performance, the scheduler should limit the frequency of VM switching which reduces performance [3]. System performance may decrease due to limiting of the overlapped execution time. Pumping noise to the side channel proposed to prevent attack due to which error rate increases and bandwidth reduced. Created noise due to random atomic memory access defends the side channel attack that happens over memory bus contention.

Literature [1, 4] proposes other methods for defending side channel attacks such as Xenpump. This method limits effectiveness in timing channels. Bandwidth of the timing channel is limited by adding some random latencies by Xenpump. Hence confusion is created to vulnerable VM that receives channel bandwidth. That unpredictability created in the receiver VM in the generated latency information is because of VM or hypervisor. This proposed model also decreases the system performance. Literature [1] presents another kind of side channel attack called as a Cache-based side channel attack that uses the prime trigger probing method for attack. Like the previous case attacker VM occupies the cache by accessing many lines and records. Then triggering is done while target VM is running a message is encoded by accessing parts of cache. Once target VM finishes its job VM used for probing starts accessing the cache, where each line used to access cache causes cache miss. It has a higher access time when compared with the base line.

A diagram is shown (Fig. 2) to describe Prime trigger probing.

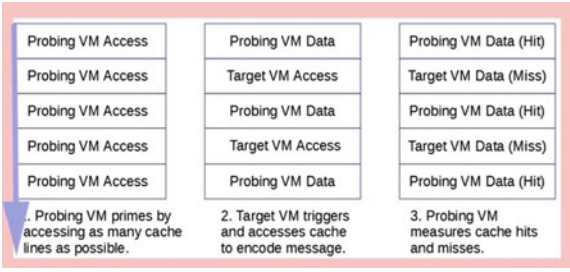

Fig. 2 Prime trigger probe method

Flushing is done between time of switching of target VM and probing VM to defend this attack, but this method creates additional overhead of 15 % [1].

3 Performance-Based Attacks and Defences

An attack is called performance-based attack if the target VM slows down because of resource hijacking by attacker VM. Literature [5] mentioned an attack which cheats scheduler to achieve 98 % of CPU usage over a physical machine. Credit-based scheduling is done in Xen hypervisor which uses token bucket scheduling. Every VM is allowed to get a credit at most 300 to store. VMs which are running will be debited its credits by one on every scheduler tick. An attacker VM blindfolds the scheduler enter before scheduler tics when a co-resident VM is in execution at scheduler which will not reflect in change of credits of attacker VM. The scheduler works in two modes, a one boost mode and a non-boost mode. In boost mode hypervisor could not differentiate among VMS waking by deliberately yielding and scheduler executed VMS. In the other mode attacker will never be debited his CPU credits. This problem can be eliminated in two ways. By using a high-precision clock and randomized scheduler ticks. In the first method the scheduler uses a clock with high precision and measures CPU usage even when VM yields or it is idle. Other method clock ticks at random intervals 10 or 30 ms. The Amazon EC2 instance will be allowed to access from 40 to 85 % of cap. An attacker VM shares resources, and calculates CPU usage if it is 85 % no more co-resident available in location otherwise there exist a co-resident [5] (Fig. 3).

I/O based attacks were discussed in literature [6]. Specially designed I/O workloads are deployed over shared queues of I/O for reducing the performance of target VM. First scheduling characteristics of hypervisor observed and extracted from targeted VM by attacking. Using this information I/O resources are hijacked and hence resulting in slow down of target I/O performance and access. Using

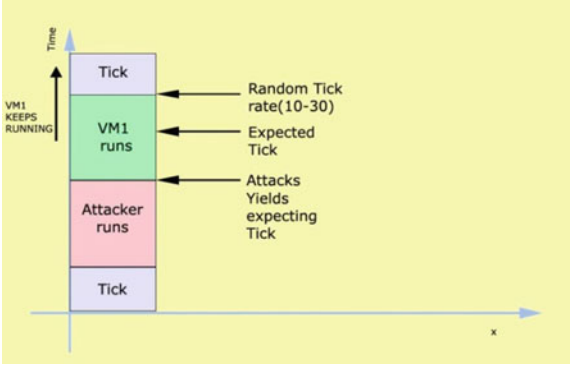

Fig. 3 Randomized schedule tick rate prevents a VM from being preempted

multiple disks to store data and accessing and the random schedule ticker can protect from I/O performance reducing attacks [5].

4 Hypervisor Security Issues and Defences

Along with side channel attacks there is one more attack possible called as virtual machine escape is an attempt of VM compromising hypervisor. Hypervisor is a layer that provides separate VMS for isolated tenants. This section presents security issues associated with VMM (Virtual Machine Manager) and VM (Virtual Machine).

A. **VMM Security**

In order to say that hypervisor is secured it should have a trusted VM. Still there are several security concerns with VMM.

(1) **VMM Vulnerabilities**:

Literature [7] mentioned, six major security vulnerabilities of VMM and emulators using auditing of the source code, fuzzing techniques. Further vulnerabilities were presented in [8] issues including VMware, Xen and other Softwares.

(2) **VM-Based Rootkits**:

As traditional Rootkit kernel is available in OS kernel, VM-based Rootkits are available in VMM. Kernel rootkit detector could not identify these VM-based rootkits. VMware and Virtual PC were used to develop these Proof of concept VM-based rootkits. They were used to observe the stealthiness of environment by considering parameters such as installation time, boot time and memory footprints [9]. Literature [10] mentioned Blue pill a VM-based rootkit. It can be installed dynamically without rebooting the system on the fly. Blue Pill leverages nested virtualization features in AMD svm. While a hypervisor is running Blue pill enable itself VMM layer. Kernel Guard [11] uses VMM policies to monitor kernel memory access for root kit attack protection. It blocks dynamic code and prevents root kit attack.

(3) **VMM Transparency**:

VMM detectability is one more major concern. In order to protect from VM detection threats it is obvious to host potential hostile codes like honepots. Literature [12] states that VMM transparency is not feasible because of discrepancies in between physical machine and VM. Several clues like time sources, hardware abstraction, and overhead were left to make VM detection possible. Four essentials for detecting VMM were mentioned in literature [13]. In same literature they mentioned an experiment which is run for detecting remotely different VMM types. Remote verifier can detect P IV architecture and VMM type (Xen, Linux,

VMware). VMM version and type can be revealed, so this may be an attack possible over VM transparency.

(4) **Platform Integrity**:

Users of the cloud has to blindly believe in trust of VMM because there is possibility of co tenants modifying data. VMM should ensure the trust for each VM that is in execution in each layer of software stack. Literature [14] proposes a model called Terra. Terra is a model built as a prototype for trusted VMM. For every application it assigns a different VM. Integrity of data can be ensured by deploying each application with an optimized OS. Literature [15] suggests Trusted Platform Module (TPM) which is a Trusted Computing Group specified security definition. Trusted Computing Group extends in accommodating virtualization techniques. vTPM embedded in TPM and can run on over external co-processor and VM. TPM 1.2 extends command set for accommodating vTPM, which enables TPM to access every VM. Hypervisor deserves tools to measure integrity of VM that is running. HIMA is proposed in literature [16] which is based on hypervisor agent. Isolating measurement agent and target is desired for tool to measure integrity. HIMA makes sure that VMS that pass integrity check only run on VMM. So it ensures healthy program in execution.

(5) **Hypervisor Control Flow Integrity**:

Literature [17] proposed a method for providing hypervisor control flow integrity called as hypersafe. Literature [18] suggested Trusted Platform Module (TPM) based on hardware to provide secure attestation, crypto graphic hashes, signatures and secure storage. Second method ensures load time integrity where as the first provides run time integrity which is very crucial. For implementing run time integrity checking Hyper safe uses two techniques. Those are restricted pointer indexing method and non-bypassable memory lock down method. Unauthorized page writes are prevented by locking down memory pages. The designed unlocking process ensures no modification is done to code or data of hypervisor. Malicious code injection for flow control in hypervisor can be prevented using memory page locking system. Literature [17] figures out Hypersafe implementation as an extension in Xen hypervisor. A new layer is created for indirecting all operators in restricted pointer indexing method in Hypersafe. This works as previous technique and control flow targets are pre-computed and stored in a table. This approach provides call target and return target to follow control flow graphs. Without any change to existing hypervisor this method can be added as an extension to compiler [17]. Use of protected hooks is suggested for monitoring untrusted VM execution to get control over applications running on it in Lares [19] framework. Applications running over untrusted VM will be monitored by VMI and security policies when control is transferred to security VM by hooks. Customized OS may not support Lares framework because change is needed on the guest OS on the fly. A state-based control flow comprising static and dynamic control flow provides kernel integrity [20]. Static control flow checking uses hashing whereas dynamic

control flow checking uses control flow graphs generated of source code. Wei et al. Addressed risks in managing security of virtual images such as publishers risk, retriever risk and repository administrator risk. The suggested solutions for access control of images and filtering and scanning images proved the better result than treating images independently. But filters may not give 100 % accurate results; virus scanning may not guarantee identifying malware in vmimage [21].

(6) **Hypervisor Integrity Checking**:

Hypersentry is a method suggested in Literature [22] for providing hypervisor security. Hypervisor is added with this new software component Hyper sentry to ensure integrity and stealth protection. For isolation of hypervisor with TPM Hypersentry uses existing hardware and also provides software integrity. Scrubbing is an attack used to remove evidence of attack without detection of higher software layer. Hypersentry acts as stealth and out-of-band channels were used to trigger this. Intelligent platform management interface (IMPI), Baseboard Management controller and System Management Mode (SMM) are used as out-of-band channels. IMPI is implemented in hardware of the hypervisor and functions independently to the CPU and other softwares of the system. BMC is a component installed over mother board for providing interface among hardware management component to remote verifier. SMM is triggered by IMPI call for providing secure environment and prevents manipulations over software which is running on machine. Interaction among these components is given in Fig. 4.

A verifiable, non-interruptible and deterministic measurement is provided in Hypersentry. It saves CPU current state after checking it thoroughly restores it. Hypersentry also provides integrity, authentication and attestation as output.

(7) **Return Oriented Programming Attack on Hypervisors**:

Return oriented programming (ROP) is one more attack mentioned in Literature [23] over Xen hypervisor which is very successful attack. It uses existing code for attack. Turing language was created by sequence chaining which ends in return statement. This is an extension of DEP (Data Execution Prevention) which is security measure implemented in most of systems today. ROP attack modifies hypervisor data which are used for control level of VM privilege level. An attacker can modify their VM level from normal level to privileged. Literature [24] suggests a defence method for ROP problem. In this solution stack is analysed continuously looking for possibilities in occurrence of ROP attack and quarantined for the use of

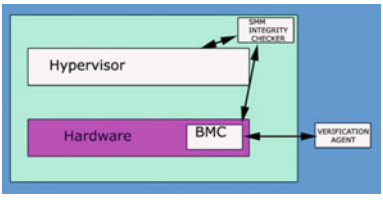

Fig. 4 HyperSentry architecture [22]

further investigations. As ROP requires many address that range in program this key feature is used to search ROP attacks using libraries.

(8) **Modifying Non-control Data**:

Literature [25] attempts to bring forward attacks over non-control data in hypervisor. There is a possibility of vulnerability in three different types of non-control data: privilege-level data, resource utilization and security policy data. An attacker uses this privilege-level data for escalation of VM privilege level. Resource utilization data helps attacker in gaining access to shared physical resources. Using security policy data attacker can attempt for side channel attacks for stealing sensitive data from target VM. Hypervisor version number helps in execution of attack over non-control data attack. Memory offsets are calculated with the help of version number of hypervisor for modifying non-control data. Writing in non-controlled data memory locations have to be limited by hardware can prevent attacks. Hypersafe [25] can be used in preventing these attacks by using non-bypassable memory lockdown.

(9) **VM Rollback Attack**:

Literature [26] mentions VM rollback attack. It assumes hypervisor is compromised already. This compromised hypervisor tries to execute VM from its older snapshot without owner's awareness. This attack damages target VM's execution history and undo security patches and updates make it vulnerable target VM. This lets attacker to bypass security system (Fig. 5).

By roll back VM state that attacks an attacker gets a chance to execute brute force password attack. Actually this will occur as when there is brute force attack occurred target VM raises security alert but compromised hypervisor brings its previous snapshot by roll back and allowing brute force attack to be possible. Using suspend/resume function we can prevent this roll back attack. But it makes developing solution more complex because it cannot distinguish between normal suspend/resume and an attack of roll back. One more requirement is this solution should not create burden for users. By securely logging all rollback actions and

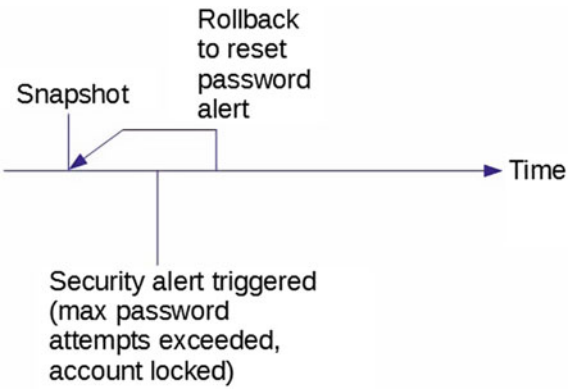

Fig. 5 Demonstrates this attack

auditing them can prevent roll back attack. Even TPM can be used in protection of log integrity. VM boot, VM suspend, VM resume and VM resume are four hyper calls used in logging information. Isolating and encrypting the VM'S memory in hypervisor helps in protecting memory hence creates solution to the rollback attack. This solution also prevents hypervisor to modify or read memory pages [26].

B. **VM Security**
Virtual Machine gives opportunity to interact with hardware in a multi-tenant and shared mode over VMM. VMS running like this should be secured and they should make sure proper security measures need to be taken. Introspection and Secure resource allocation are issues related to VM security.

(1) **Introspection**: VM Introspection is a process to track data flow inside guest VMS; it has many challenges. Moonsols livecloudkd is one such implementation presented in Literature [27]. It debugs a guest OS running over Hypervisor by allowing KD and windbg to inspect Windows Virtual Machine delivered from Microsoft Hyper-V R2 hypervisor. VIX is a tool suite presented in Literature [28] used for introspection of Xen. It tracks guest VMS Process by mapping domU's virtual memory to Dom0's memory address in VMM.
(2) **Secure resource allocation**: complete isolation of VMS in hypervisor will reduce performance and efficiency in resource utilization hence it may not be a deserved solution. Efficiency in utilizing resources with security is needed. So literature [29] suggested resource sharing in Hypervisor through shype [30]. It uses MAC-based policy for security to share resources without compromise in security with minimized overhead.

5 VM Isolation Techniques

This section brings four different approaches used in isolation of VM. Literature [31] suggests security Turtles an architecture based on nested virtualization for protection of guest VM.

Even though attacks possible in Level 1 Hypervisor for VMS running Level 2, Level 0 Hypervisor is the highest privileged which protects.

(1) Lifetime kernel code integrity of Level 1 hypervisor.
(2) Code-data integrity in QEMU-KVM daemons.
(3) Data integrity in Level 2 execution of Guest VM.
(4) Guest VMS running in Level 2 need to be aware that weather there is any violations in above 3 requirements.

Secure Turtles believe that outside attacks are not possible over hypervisor of Level 0. Literature [32] eliminates hypervisor almost and proposed a new approach. Minimizing code base area will reduce vulnerabilities in virtualization software also

reduces its functionality. Multi-tenancy is a needed requirement to be provided in cloud for provisioning resources to customers on demand leveraging economies. In this literature they propose a solution temporary Hypervisor; a temporary hypervisor that runs at initialization. It is consists of pre-allocated hardware resources (process cores, memory, etc.). It sets virtualized I/O to avoid indirection for bringing the VM a more direct interaction with hardware.

Provisioning each VM with individual I/O devices is not much practical so virtualized I/O devices are encouraged to be used. Once virtual I/O devices are allocated, hard will make sure isolation in between virtual machines without hypervisor. Literature [33] describes a mandatory access control policy (MAC) which allows VMS to share resources like disk, networks, etc., this also supports multiple hypervisors to share these resources. Bind time authorization is used to obtain high performance and Chinese wall is also included. Chinese wall assigns every VM a type. It will not allow to run two VMS concurrently that there is conflict with type. Hence it prevents attacker VM not to use covert channels for accessing target VM sensitive data. Type-enforcement is done to specify access of VMS to resources. Literature [34] suggests a multilevel set of security. There are two different types of hypervisors known as pure isolation and shared hypervisor. In hypervisor with pure isolation partitioning is done in machine and it will not allow resource sharing except memory and CPU. Hypervisor with sharing will allow file sharing. There will be two partitions low level and high level. Low level is allowed to read/ write in same level of security where as higher security partition has read only permission to access security data of low level partition. One-way network implementation is done. Secure shared file store is suggested in another implementation. Hypervisor uses cross-ring call for access control to sub system. It also implemented as separate partition which uses message passing mechanism.

6 Conclusion

Virtualization enable cloud computing facilitated several guest VMS to share common physical hardware. Hypervisor is the key component in virtualization. Hence it must resist attack effectively by isolating VMS. But in reality it is vulnerable and exposed to several security flaws such as VM escape. It is said to be the most serious among several attacks said above. An escaped VM will compromise several co-resident VMS. Several architectural and design changes are needed in Hypervisor for potential resistance of VM escape attack. Side channel attacks hijack system resources and steal sensitive data of co-resident VMS. Some solutions were discussed in mitigation of side channel attacks as adding noise, etc. Hypervisor security enables security to the cloud environment which results in trust building and enterprises motivation of migration to cloud.

References

1. M. Godfrey and M. Zulkernine, "A Server-Side Solution to Cache-Based Side-Channel Attacks in the Cloud," Proc. Of 6th IEEE International Conference on Cloud Computing, 2013, pp. 163–170.
2. S. Yu, X. Gui, J. Lin, X. Zhang, and J. Wang, "Detecting vms Co-residency in the Cloud: Using Cache-based Side Channel Attacks," Elektronika Ir Elektrotechnika, 19(5), 2013, pp. 73–78.
3. F. Liu, L. Ren, and H. Bai, "Mitigating Cross-VM Side Channel Attack on Multiple Tenants Cloud Platform," Journal of Computers, 9(4), 2014, pp. 1005–1013.
4. J. Wu, L. Ding, Y. Lin, N. Min-Allah, and Y. Wang, "xenpump: A New Method to Mitigate Timing Channel in Cloud Computing," Proc. Of 5th IEEE International Conference On Cloud Computing, 2012, pp. 678–685.
5. F. Zhou, M. Goel, P. Desnoyers, and R. Sundaram, "Scheduler Vulnerabilities and Coordinated Attacks in Cloud Computing," Journal of Computer Security, 21(4), 2013, pp. 533–559.
6. Z. Yang, H. Fang, Y. Wu, C. Li, B. Zhao, and H. Huang, "Understanding the Effects of Hypervisor I/O Scheduling for Virtual Machine Performance Interference," Proc. Of 4th IEEE International Conference on Cloud Computing Technology and Science (cloudcom 2012), 2012, pp. 34–41.
7. T. Ormandy, "An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments," in cansecwest, 2007.
8. The MITRE Corporation, "Common Vulnerability and Exposures (CVE)," http://cve.mitre.org/, Mar. 2011.
9. S. King and P. Chen, "Subvirt: implementing malware with virtual machines," in IEEE Symposium on Security and Privacy, May 2006.
10. J. Rutkowska, "Subverting Vista kernel for fun and profit," 2006.
11. J. Rhee, R. Riley, D. Xu and X. Jiang "Defeating dynamic data kernel Root-kit attacks via VMM based guest transparent monitoring". In proceedings of ARES 2009, conference 2009, To appear.
12. T. Garfinkel, et al., "Compatibility is not transparency: Vmm detection myths and realities," in hotos, 2007.
13. J. Franklin, et al., "Remote detection of virtual machine monitors with fuzzy benchmarking," SIGOPS Oper. Syst. Rev., April 2008.
14. T. Garfinkel, et al., "Terra: a virtual machine-based platform for trusted computing," in SOSP, 2003.
15. Trusted Computing Group, http://www.trustedcomputinggroup.org/, June 2011.
16. A. Azab, et al., "Hima: A hypervisor-based integrity measurement agent," in ACSAC, dec. 2009.
17. Z. Wang and X. Jiang, "hypersafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity," Proc. Of IEEE Symposium on Security and Privacy, 2010, pp. 380–395.
18. M. Kim, H. Ju, Y. Kim, J. Park, and Y. Park, "Design and Implementation of Mobile Trusted Module for Trusted Mobile Computing," IEEE Transactions on Consumer Electronics, 56(1), 2010, pp. 134–140.`.
19. B.D. Payne, Macaroni, M. Sharif and W. Lee." Lares: an architecture for secure active monitoring using virtualization." Security and privacy IEEE Symposium ON, 0:233–347.
20. N.L. Petroni, Jr and M. Hicks, " automated detection of persistent kernel control flow attacks". In CCS'07: proceedings of the 14th ACM conference on Computer and communications security, pages 103–115, New York NY, USA 2007, ACM.
21. Jinpeg Wei, Xiaolan Zhang, Glenn Ammons, Vasantha Bala, Peng nns, "Managing security of virtual machine images in a cloud environment", in CCW'09 proceedings, Chicago, Illinios, USA, ACM 978-1-60558-78-4/09/11.

22. A. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. Skalsky, "hypersentry: Enabling Stealthy In-context measurement of Hypervisor Integrity," Proc. Of 17th ACM Conference on Computer and Communications Security, 2010, pp. 38–49.
23. B. Ding, Y. Wu, Y. He, S. Tian, B. Guan, and G. Wu, "Return- Oriented Programming Attack on the Xen Hypervisor," Proc. Of 7th International Conference on Availability, Reliability and Security, 2012, pp. 479–484.
24. X. Jia, R. Wang, J. Jiang, S. Zhang, and P. Liu, "Defending Return-oriented Programming Based on Virtualization Techniques," Security and Communication Networks, 6(10), 2013, pp. 1236–1249.
25. B. Ding, Y. He, Y. Wu, and J. Yu, "Systemic Threats to Hypervisor Non-control Data," Information Security, 7(4), 2013, pp. 349–354.
26. Y. Xia, Y. Liu, H. Chen, and B. Zang, "Defending against VM Rollback Attack," Proc. Of 2nd International Workshop on Dependability of Clouds, Data Centers and Virtual Machine Technology (DCDV 2012), 2012.
27. Moonsols, "livecloudkd," http://www.moonsols.com/2010/08/12/livecloudkd/, Aug. 2011.
28. B. Hay and K. Nance, "Forensics examination of volatile system data using virtual introspection," SIGOPS Oper. Syst. Rev., April 2008.
29. R. Sailer, et al., "Building a mac-based security architecture for the xen open-source hypervisor," in ACSAC, 2005.
30. S. Berger, et al., "vtpm: virtualizing the trusted platform module," in USENIX Security Symposium, 2006.
31. F. Liu, L. Ren, and H. Bai, "Secure-Turtles: Building a Secure Execution Environment for Guest vms on Turtles System," Journal of Computers, 9(3), 2014, pp. 741–749.
32. J. Szefer, E. Keller, R. Lee, and J. Rexford, "Eliminating the Hypervisor Attack Surface for a More Secure Cloud," Proc. Of 18th ACM Conference on Computer and Communications Security, 2011, pp. 401–412.
33. R. Sailer, T. Jaeger, E. Valdez, R. Caceres, R. Perez, S. Berger, J. Griffin, and L. Van Doorn, "Building a MAC-based Security Architecture for the Xen Open-source Hypervisor," Proc. Of 21st Annual Computer Security Applications Conference (ACSAC 2005), 2005, pp. 276–285.
34. P. Karger, "Multi-level Security Requirements for Hypervisors," Proc. Of 21st Annual Computer Security Applications Conference (ACSAC 2005), 2005, pp. 267–275.