Deep-diving into an Easily-overlooked Threat: Inter-VM Attacks

Su Zhang
Kansas State University
Manhattan, KS
zhangs84@ksu.edu

Abstract

‘Cloud computing’ has been embraced more than ever before due to a number of obvious advantages (*e.g.* elasticity, space-saving, hardware consolidation and cost-saving) over traditional computing infrastructure. Virtualization is the core part of cloud computing as it provides a multi-tenancy model which is the foundation of cloud computing. Typically there are two types of threats inherent with virtualization environments. One wildly noticed threat is vm-escape which means a user on a virtual machine can ‘escape’ from it and take control over the whole hypervisor. The other threat known as inter-vm attacks have often been overlooked by the research community. In this paper we want to propose a framework which can systematically detect/analyze a number of inter-vm attacks. We want to build a system capturing such threats with a low false positive rate while providing valuable recommendations to system administrators.

Keywords: Inter-vm attacks, Cloud Computing, Virtualization, Hypervisor Security

1 Introduction

Generally speaking, cloud services can be categorized into three models: Software as a service (SaaS), Platform as a service (PaaS) and Infrastructure as a service (IaaS). A cloud service becomes more vulnerable as more freedom is given to the users since any one of them may be an attacker. Furthermore, it is more error-prone for normal users to manage a whole virtual machine (IaaS) than just a single piece of software (SaaS). Therefore, security threats under IaaS model deserves more attention. There are a number of security issues that need to be considered under the IaaS model such as multi-jurisdictional issue, redundancy checking, privacy-preserving calculations, and hardening virtualization environment. However, it is the last one that requires the most attention because virtualization technology is the enabler and core component of cloud computing.

There are typically two types of threats brought by virtualization. One is called VM-escape, and the other is inter-vm attack. VM-escape means that attackers launch attacks from a virtual machine to compromise the hypervisor it is running on. This could allow attackers to monitor or compromise other co-residing virtual machines. VM-escape attacks are catastrophic threats to the cloud platform as the hypervisor is a single point of failure. A number of works [10, 9] have introduced a various of VM-escape vulnerabilities, even though no such attack in real world has been reported by any cloud service providers (CSPs). All VM-escape attacks are very sophisticated and ad-hoc while hard to detect. Consequently, hypervisor hardening has become a hot topic in research and several works [21, 15, 20, 16] have proposed techniques to realize this. Inter-VM attacks refer to attacks launched from one virtual machine to another co-residing virtual machine directly, typically by bypassing the virtual machine monitor (VMM). This type of attack can be easier to launch and harder to detect. In this paper we focus on inter-vm attacks for the following reasons. Firstly, a certain number of inter-vm communication mechanisms [19, 7, 8] needs to bypass the VMM in order to achieve high performance. Since system administrators mostly monitor the VMM the VMM bypass attack may not be detected. Secondly, compromising a hypervisor would be more difficult than penetrating into individual virtual machines due to the fact that the hypervisor is typically well monitored and better hardened by people both in industry and in academia. Moreover, the VMM typically has a smaller attack surface than an ordinary operating system (e.g. Xen has only 200 KSLOC while Linux kernel has over 1,000 KSLOC). Thirdly, researcher have focused on hypervisor-security rather than inter-vm threats. Lastly, inter-vm threats are potentially more damaging than previously thought as once a virtual machine within a cluster (a group of machines communicating with each other through memory-sharing channel) is compromised, the rest of the machine may be more vulnerable. The rest of this report is structured as follows. Section 2 provides background about Xen – an open source hypervisor which will probably be used at the experimental section. Section 3 talks about related works including several state-of-arts inter-vm communication mechanisms, existing hardening techniques and several threat models. Section 4 asks research questions and illustrates a speculative framework to effectively detect and response to inter-vm attacks. Section 5 states the directions of future work.

2 Related works

A number of inter-vm communication mechanisms [25, 11, 23, 7, 17, 19, 22, 8] have been proposed. They mostly strive for a better performance (e.g. high throughput, low latency, etc). They are mostly based on shared-memory channel (bypassing virtual network interface (VIF)) because there is a need for high efficiency inter-vm communication (services requested by customers may be provided by highly collaborative virtual machines like web service needs independent web server and database server both of which need to communicate with each other efficiently). Packets go through the VIF between different virtual machines on the same physical machine is one factor of magnitude slower than shared memory-based approaches [25]. A number of summaries [6, 18] are in existence but again they stress more on performance than security. However, we want to focus on security evaluation over this new trend of inter-vm communication mechanism (shared memory based channel). Performance penalty will be taken into consideration. Besides, there are tools detecting inter-vm threats, but they merely rely on low-level data (e.g. resource usage) to identify the threat other than utilize anything related to the new type of mechanisms.

2.1 State-of-the-art inter-vm communication mechanisms

2.1.1 Xen Default Virtual Network Interface

The default virtual network interface (VIF) is a full emulation of a network stack as a normal physical machine. VM communicates with each other as if they are on different physical machines. Packets go through the whole network stack of one virtual machine and then cross another one in a reversed order. Even though this mechanism can provide a highest isolation, it performs poorly when bridging co-residing virtual machines. The reason why the virtual network interface has such performance limitation has been partly stated in [12]. The performance penalty couldn't be affordable if tasks count on intensive inter-vm communications. The inter-VM security level under this mechanism is the same as networked environment as all of the hosts are treated as fully-isolated physical machines.

2.1.2 XenStore [5]

XenStore can be regarded as a database storing configuration information of each virtual machine (domain) running on the same hypervisor. It also can be used as an inter-domain communication channel, but the channel is narrow (the maximum size of each message is 1024 bytes). Message here typically refers to a variable-value pair. Therefore XenStore may mostly be used as a place for data collection (for the configuration information (resource (memory, CPU cycles) usage) of each virtual machine). VMs still need to count on other channels (memory-share based channel, IO buffer or VIF) exchanging large amount of data.

2.1.3 XenLoop [19]

Mechanism: XenLoop relies upon a self-contained Linux module (netfilter) in each virtual machine. It can intercept and inspect each outgoing packets. If the destination host is a co-residing vm, then the packet will be delivered through a shared memory channel (two FIFO pipes (one for each direction allocated from receiver's memory space) and one event channel as a notifier to let each authenticated (through grant table references) end-point realize the data/space is ready to read/write in the pipe) bypass the VIF, otherwise the it will go through the default VIF.

Performance: This mechanism is user-transparent because even if the host is migrated to another physical machine, it is still able to use VIF to reach the other VMs on its previous physical machine. The two mechanisms are switchable which makes live-migration easier. However, the throughput and latency may not as good as other fully-mapped approaches (e.g. [4]) because it is limited by the size of the shared buffer.

Security Concern: A hypervisor-bypassing communication reduces the attack surface since the VMM is a single point of failure. However, it increases the chance of inter-vm attack (by letting the VM to manage its own grant tables). Along with the fact that shared-memory based communication doesn't check the content of exchanging memory. Therefore a client-side attack could be easily triggered since the shared-memory channel is a proper place for malicious code injection. Furthermore, DDoS attacks could be potentially launched. If attackers owning a number of VMs, then He can establish as many as number of services from the VM he has. The guests can only tear down the inactive communication (by checking the existence of other communication parties' tokens storing in XenStore) channels. However, there is no defined rules for user to identify if the other party connecting to its machine is a malicious or benign user. Current standard is far from enough from the security's perspective.). The FIFO (shared ring buffer) can be filled up easily by co-residing attackers.

2.1.4 XenSocket [25]

Mechanism: XenSocket is another memory-sharing-based communication mechanism like XenLoop. Similar to XenLoop, it doesn't include per packet page-flipping in order to have a better performance (reduce the number of hypercalls which could generate lots of performance overhead). However, it only has one circular shared buffer rather than two (unlike XenLoop). So it is a unidirectional (rather than bidirectional) communication mechanism. It adapted a Unix socket-like channel to establish the connection. The server (receiver) first create a socket, allocate a share memory (including a descriptor page and a circular shared buffer), obtain the grant table reference to the descriptor page, create an event channel for event notifications between the two end-points. At the client (sender) side, it connect to the server side through the function connect (), the function could also help it obtain the address of the physical pages of the shared circular buffer (the address was placed in shared address memory when bind () is called at the server's side, so that bind could return the grant table reference address to the server). Meanwhile, the sender also hooks up with the event channel created by server and the very beginning for further events notification (e.g. arrival of new data or more space available due to the deletion of newly consumed data). In order to avoid DoS attack, resources (memory) must be shared by a lower-trustworthy party and mapped by a higher-trustworthy party. Therefore the server side must be lower than client side in terms of trustworthy level (However, this wouldn't guarantee a secure communication as the shared memory from the lower trustworthy party main contain malicious code which could put the higher trustworthy host in trouble once the connection established). Also in order to keep the synchronization feature between the two parties, the connection must be first terminated from the client side as the server side owned all of the resources (descriptor page, event channel and shared circular buffer). So if server side terminate first, parts of data in the air may have been lost, making the synchronization impossible. There is a functionality drawback of XenSocket since it only allows lower trusted host as server (receiver) while higher trusted host as client (sender). It is not convenient if two parties want to have a bidirectional communication. XenLoop resolved this issue gracefully.

2.1.5 Fido [4]

Fido is especially designed for highly-collaborative VMs over the same hypervisor. It assumes that each virtual machine on the hypervisor is benign. Therefore all of the communication Virtual Machine's memory are readable to each other. They utilize the advantage of 64 bit addressing strategy to resolve the throughput limitation from circular shared producer and customer buffer. Because a host typically only uses a minor part of the spaces mappable by all of the 64-bit addresses. The initial memory mapping information is exchanged through bootrap step via I/O ring (The circular producer and consumer buffer). Besides memory mapping, they have another two components included in their framework: connection module and signaling module. The connection module is managed through xenStore (a centralized key-value configuration database in dom0). XenStore makes the connection dynamically pluggable since communicating virtual machines synchronize the connection information through a connection token storing on XenStore. Once there is a change on the token, the communication VMs will be notified. The connection management is similar to XenLoop. The signaling module used here is the same as event channel in Xen, which is used as a notifier of an arrival of an event. The security concern of this work is about the assumption that each of the virtual machine on the hypervisor is trustworthy. This assumption may be true under certain circumstance. However, one physical server may hold several clusters of VMs. VMs within the same cluster are trustworthy to each other. But if a VM compromised by attackers outside of its cluster, then the rest of VMs in the same cluster will be completely exposed to the attackers. The data transfer instead of copying everything, they just share the pointers and size of each chunk of shared data (in source VM) to the destination VM. The author introduces a global virtual address by mapping all communication virtual machines into 64-bit addresses, so each virtual machine will have a global page table including all of the co-residing virtual machine's address.

2.1.6 Inter-OS Communication on Highly Parallel Multi-Core Architectures [23]

The authors break the multi-trusted level rules among all of the virtual machines, instead, they treat all of the virtual machine equally by extending the functionality of grant table. Grant table is used only for memory shared based communication between privileged domain (Dom0) and normal user domain (DomU) for the data used by device driver. They allow arbitrary virtual machine within the same physical host to share resources with each other based on the grant table. However, this new settings would result in potential DoS attack if there is any user in a virtual machine is malicious. Even though this somehow increase the performance, it could only work for internal enterprise network while all of the users are trustworthy.

2.1.7 A High-efficient Inter-Domain Data Transferring System for Virtual Machines [11]

The author combines several advantages of previous work and overcome a number of difficulties of them. It adapted bidirectional communication mechanism like XenLoop (but unlike XenSocket). It is also a VIF-bypass mechanism. While this feature somehow increased the performance (if all of the traffics are belong to the virtual machine within the same physical server), it compromise the migration compatibility. Also it seems that dom0 is a via-point for all of the traffics which would potentially result in DoS attacks if there are malicious virtual host continuously establish connections to dom0. All of the rest components (grant table, event channel and shared buffer) utilized are the same as other popular communication mechanisms.

2.1.8 Virtual machine aware communication libraries for high performance computing [7]

This work is based on a high performance computing library developed by the authors previously. Under high performance computing environment, processes need to cooperate while maintain a certain level of isolation. Therefore it would be good if the communication between processes on two different virtual machines could be comparable to the IPC within the same machine. The communication mechanism is a combination of message passing and shared-memory based inter-vm communication. The process will invoke a function within IVMC lib which will allocate memory pages as shared buffer pages. Meanwhile, it will utilize a socket-liked communication channel to notify the other endpoint about the address of the shared memory, grant table reference and event channel port number. The rest of the inter-vm communication will be the same as the others. Their work could also avoid data lose during vm's migrating by copying the unread data into a cushion buffer before the terminate the connection. Another advantage of this work is VMM-bypass which dramatically reduced the performance overhead.

2.1.9 Inter-domain Socket Communications Supporting High Performance and Full Binary Compatibility on Xen [8]

The advantages of this work including binary compatibility, bypassing VIF, avoiding page flipping, offering a direct, accelerated communication path between domains. Bypassing VIF could minimize the latency while save the CPU cycles. Avoid page-flipping could reduce the number of hypercalls which could take much time. Also, page table and TLB need to be flushed as the mapping between virtual and physical addresses changed. A virtual device – XWAY has been created for inter-domain data exchange. It manages XWAY channels, transfer data, and inform upper layers of any changes in the status of queues. XWAY channel is created while both endpoints agreed on tokens of shared memory, port number of event channel through another control channel across two arbitrary domains. The XWAY packets are TCP format compilable (with a XWAY specific extension appended to each TCP packet).

2.1.10 VMware VMCI [17]

VMware adapted hardware-assisted virtual machine (HVM) mechanism other than the para-virtualization used by Xen. A device called VMCI needs to be installed. The application needs to be re-compiled against with existing communication libraries. Another disadvantage of it is it prohibits network connection therefore live-migration may be affected if a virtual machine is migrated to another physical host (As it may want to change the network mode to standard TCP/IP communication mechanism).

2.2 Threat modeling

There are a number of possible inter-vm attacks. Since not all of the co-residents are trustworthy to each other, we need to dig into the details of inter-vm attacks in order to know how could these attacks be detected or prevented.

2.2.1 Denial of Service

Denial of service attack (DoS) is a catastrophic threat from the server's perspective. As we can see from the previous analysis, both privilege host (Dom0) and normal guest hosts (DomUs) are under the threat of such type of attacks due to the weak authentication with current communication mechanisms. Sometimes in order to pursue a higher degree of performance, system designer would rather sacrifice certain level of security. This maybe okay if the whole system is isolated and used by trustworthy users (e.g. for internal use of a corporation, or by a small community with none of them has conflict of interest.) Nowadays, virtualization is the core technology of cloud computing, and the tenants on the same physical server may have conflict of interest, or maybe there are malicious users as there are no strict authentication before the tenants are allowed to use the service. Therefore strong isolation, good performance and effective tools (can detect/analysis aforementioned threats) are of vital importance.

2.2.2 Side-Channel Threat [24, 13]

There are several works related to side-channel attacks. Since more than one tenants could share a same set of hardware, therefore certain sensitive information (like the pattern of accessing data) may be leaked to malicious users. For example, encryption keys could be derived based on side (timing)-channels.

2.2.3 Threat to Dormant VM

And even if VMs are dormant, attackers may still be able to access them. Because the data center is always on, an offline VM is not equivalent to an off-powered computer at home. Also pre-build images need to be carefully scanned in order to avoid some legacy vulnerable applications or trapdoors (e.g. AWS pre-build images store builder's SSH keys in it which means all of the hosts using such type of images are accessible from their publisher.).

2.3 Existing Protecting Works

There are existing hardening or detecting tools to identify virtualization-related threats. However, they are mostly focus on the hypervisor security. While the hypervisor is a single point of failure, it could be easily detected by system admin once it got compromised. Therefore even though inter-vm attack is not as bad as vm-escape, it is hard to be discovered from the system admin's position. As an attack may have already known which vm he want to attack, therefore he only need to target the potential virtual machine rather than the whole hypervisor in order to reduce the chances of being discovered and increase the chance of compromising the target successfully.

2.3.1 Virtual Machine Monitor-Based Lightweight Intrusion [2]

The author build an intrusion detection system over the VMM. Based on the resource consumption information, they define several rules as the trigger of the alerts based on historical data. They trained the data by using data-mining algorithms and continuous applying new data with the model. Alerts will be raised once outlier is caught.

2.3.2 Building a MAC-based security architecture for the Xen open-source hypervisor [14]

This framework is built upon VMM and incorporated with a security policy defined by the system administrator/user. It is a role-based access control for privileged actions, it will first trap each privileged action and check it against with security policy see if it is legal, illegal actions will trigger alerts. System admins can be ensured that each action conforms to the security policy as long as there is no alert raised. This can somehow guarantee each virtual machine can only access the resource it ought to.

2.3.3 HomeAlone: Co-Residency Detection in the Cloud via Side-Channel Analysis [26]

Side-channel is regarded as a catastrophic threat as aforementioned. The author proposed a way of utilizing side-channel as a detector to identify illegal co-residency based on the timing channel (the response time of accessing L2 cache). They set up a threshold (response time in seconds). If the response time is longer than the threshold, then it'll be regarded that the cache (so do the other hardware on the same machine) has been used by other illegal co-resident, otherwise they conclude that the user is using the hardware resource exclusively. They evaluated their tool installed on different VMM environments with all true positive rate about 0.8.

2.3.4 A Security-Aware Scheduler for Virtual Machines on IaaS Clouds [1]

This work tries to solve the conflict of interests among different VMs at a high-level. Users are allowed to provide a list of names including all of their conflict of interest parties. The scheduler (system administrator) will schedule all of these virtual machines based on their priority. The VM with higher priority could be scheduled right away while the VMs with lower priority and VMs with conflict of interests with the current running VMs will be scheduled later or migrate to other physical computers. This work stresses the importance of resolving potential cross-vm attacks at high level.

3 Framework

We here propose a framework to resolve the security issues raised under the memory sharing based inter-VM communication environment.

3.1 Objective

As mentioned earlier, there are a number of reasons why inter-vm threats deserve more attention. The first reason is because this type of attack is harder to be detected than vm-escape threat and system admin or defenders pay most of their attention over the hypervisor. Based on the related works, we can find out that VIF-bypassing memory-sharing based inter-VM communication will become mainstream mechanism as highly interactive VMs require better performances than current settings. However, this type of mechanism needs to be inspected carefully because co-residing VMs may not be trustworthy to each other.

3.2 Generalization over existing inter-vm communication mechanisms

We have seen that most state-of-the-art inter-VM communications include three essential components: shared memory buffer(s), grant table and event channel.

3.2.1 shared memory

Shared memory is the actual communication channel between VMs. It is typically like a producer and consumer circular buffer shared between communication VMs. Generally it is a bidirectional asynchronous communication. VM can be both writer and reader. Once it finished writing, it will let the

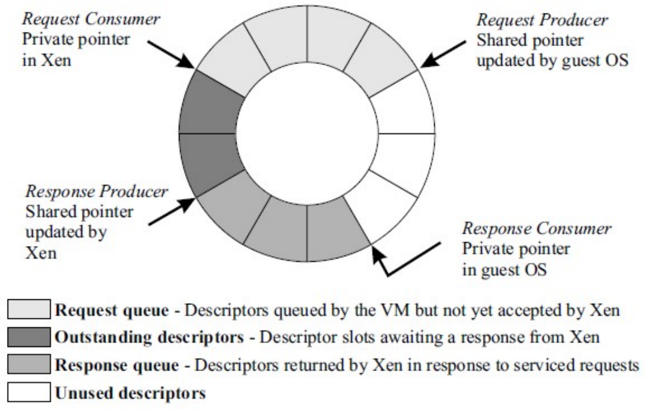
Request Consumer
Private pointer in Xen
Request Producer
Shared pointer updated by guest OS
Response Producer
Shared pointer updated by Xen
Response Consumer
Private pointer in guest OS
Request queue - Descriptors queued by the VM but not yet accepted by Xen
Outstanding descriptors - Descriptor slots awaiting a response from Xen
Response queue - Descriptors returned by Xen in response to serviced requests
Unused descriptors

Figure 1: The structure of asynchronous I/O rings

other party to know that the data is ready. If it finished reading, it will delete the data it just read on the shared buffer and let the other party know that more space is available for new input. Every time before the writer starts to write, it will check if the buffer has enough space available for the next input. If not enough available space, it will then wait for the reader to 'consume' the data on the buffer until the available space is enough. Therefore the size of the shared memory channel needs to be carefully designed in order to reduce latency. Figure 1 is a classical I/O shared buffer design cited from [3].

3.2.2 Event Channel

Event channel is like a signal channel to let communication parties know the arrival of a new event. The writer tell the reader for the data it just wrote. On the other hand, once a reader finished reading it will delete the data and tell the writer that new space is ready for the next input through event channel. Therefore, event channel must be well protected, otherwise it will mess up the whole communication. Delivering wrong information may cause the share-memory channel out-of-sync.

3.2.3 Grant table

Grant table provides two types of grants between different VMs. One is page-flipping and the other is page-sharing. Since per-packet page-flipping will have too much performance overhead (due to the high frequency of hypercalls). Therefore the new communications dropped page-flipping but preserve page-sharing grant. Xen memory sharing mechanism is at a page granularity. Shared pages are identified by an integer, known as a grant reference. The hypervisor keeps the grants information and pass the grant reference to communication VMs and signal it via event channel. The hypervisor will be the authority to authenticate the communication parties. Under certain circumstance [19], the system may delegate communication parties to manage the grant table by themselves for performance reasons.

3.3 Speculative Framework

To systematically build a framework to detect or defend against such threats, we need to understand the whole system completely. Also we need to know what data needs to be collected for analysis and what correlations need to be built based on the possible threat models as we discussed earlier. We draw a draft framework ((see figure 2))to give a high level impression to the audience while make it extensible to more scenarios (threat models) discovered in the future.

3.3.1 Data Collection

Data used for correlations could be obtained from various places like event channel, XenStore, grant table and shared memory channel. The event channel needs to be well monitored because once it couldn't function correctly, the shared memory channel may be out-of-sync, then the connection will be massed up completely. The information stored in grant table must be archived as well because these record can be used for future audit. This type of information is analogous to network log on apache server. The shared-memory channel needs to be monitored as it is like packets transmitted over network. For this new mechanism, there is no IPS or firewall residing between the communication VMs. However, the transmitted content must be scrutinized in order to secure the whole communication. XenStore can be used as a place for low level data collection (e.g. resource usage and configuration parameters of VMs).

3.3.2 Alerts Correlation

Based on the data collected and threat models discussed, correlations can be built to raise alerts when anomalous situation detected. We are going to use a MulVAL like logical-based engine to analyze over the data and aim at raising alerts with low false positive rate. Furthermore, more scenarios (correlations) can be added easily to our engine therefore as more threat models discovered, we can make our tool more completed.

References

[1] Zaina Afoulki, Zaina Afoulki, and Jonathan Rouzaud-Cornabas. A security-aware scheduler for virtual machines on iaas clouds. Technical Report RR-2011-08, LIFO, ENSI de Bourges, August 2011.

[2] Fatemeh Azmandian, Micha Moffie, Malak Alshawabkeh, Jennifer Dy, Javed Aslam, and David Kaeli. Virtual machine monitor-based lightweight intrusion detection. *ACM SIGOPS Operating Systems Review*, 45, July 2011.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *the nineteenth ACM symposium on Operating systems principles*, New York, NY, December 2003.

[4] A. Burtsev, K. Srinivasan, and P. Radhakrishnan. Fido: Fast inter-virtual-machine communication for enterprise applications. In *2009 USENIX*, San Diego, CA, June 2009.

[5] David Chisnall. *The Definitive Guide to the Xen Hypervisor.* Prentice Hall, 2007.

[6] C. Gebhardt and A. Tomlinson. Challenges for inter virtual machine communication. Tech-

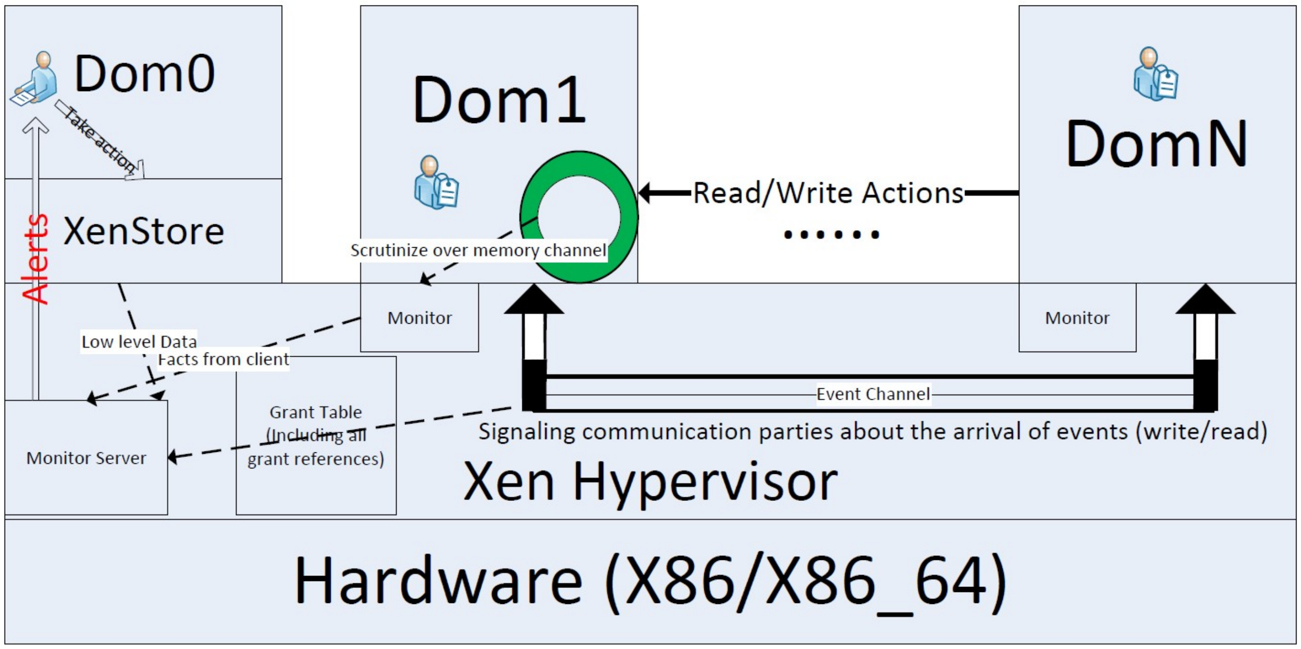

Figure 2: Speculative Framework

nical Report Technical Report RHUL-MA-2010-12, Department of Mathematics Royal Holloway University of London, August 2010.

[7] W. Huang, M. J. Koop, Q. Gao, and D. K. Panda. Virtual machine aware communication libraries for high performance computing. In *ACM/IEEE conference on Supercomputing*, New York, NY, November 2007.

[8] K. Kim, C. Kim, S.-I. Jung, H.-S. Shin, , and J.-S. Kim. Inter-domain socket communications supporting high performance and full binary compatibility on xen. In *the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, New York, NY, March 2008.

[9] KostyaKortchinsky. Cloudburst: A vmware guest to host escape, August 2009.

[10] Invisible Things Lab. Vm-escape: Xen0wning trilogy, August 2008.

[11] D. Li, H. Jin, Y. Shao, , and X. Liao. A high-efficient inter-domain data transferring system for virtual machines. In *the 3rd International Conference on Ubiquitous Information Management and Communication*, New York, NY, January 2009.

[12] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel. Diagnosing performance overheads in the xen virtual machine environment. In *First International Conference on Virtual Execution Environments*, Chicago, IL, June 2005.

[13] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud! Exploring information leakage in third-party compute clouds. In Somesh Jha and Angelos Keromytis, editors, *Proceedings of CCS 2009*, pages 199–212. ACM Press, November 2009.

[14] 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. In *Annual Computer Security Applications Conference (ACSAC)*, Washington, DC, December 2005.

[15] Arvind Seshadri, Mark Luk, and Ning Qu Adrian Perrig. Secvisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity oses. In *twenty-first ACM SIGOPS symposium on Operating systems principles (SOSP)*, New York, NY, December 2007.

[16] J. Szefer, E. Keller, R. B. Lee, and J. Rexford. Eliminating the hypervisor attack surface for a more secure cloud. In *the 18th ACM conference on Computer and communications security (CCS)*, Chicago, IL, October 2011.

[17] VMware. Vmci sockets programming. www.vmware.com/pdf/ws65_s2_vmci_sockets.pdf, 2008.

[18] J. Wang. Survey of state-of-the-art in inter-vm communication mechanisms. Technical Report Research Proficiency Report, Binghamton University, September 2009.

[19] J. Wang, K.-L. Wright, and K. Gopalan. Xenloop: a transparent high performance inter-vm network loopback. In *the 17th international symposium on High performance distributed computing*, New York, NY, June 2008.

[20] Jiang Wang, Angelos Stavrou, and Anup K. Ghosh. Hypercheck: A hardware-assisted integrity monitor. In *the 13th International Symposium on Recent Advances in Intrusion Detection*, Ottawa, Canada, September 2010.

[21] Zhi Wang and Xuxian Jiang. A lightweight approach to provide lifetime hypervisor control-flow integrity. In *the 2010 IEEE Symposium on Security and Privacy (Oakland)*, Oakland, CA, May 2010.

[22] M. Wegiel and C. Krintz. Xmem: type-safe, transparent, shared memory for cross-runtime communication and coordination. In *the 2008 ACM SIGPLAN conference on Programming language design and implementation*, New York, NY, June 2008.

[23] Lamia Youseff, Dmitrii Zagorodnov, and Rich Wolski. Inter-os communication on highly parallel multi-core architectures. Technical Report TR 2008-17, UCSB, October 2008.

[24] Kehuan Zhang, Zhou Li, Rui Wang, XiaoFeng Wang, and Shuo Chen. Sidebuster: Automated detection and quantification of side-channel leaks in web application development. In *Proceedings of CCS 2010*. ACM Press, October 2010.

[25] X. Zhang, S. McIntosh, P. Rohatgi, , and J. L. Griffin. Xensocket: A high-throughput interdomain transport for virtual machines. In *the ACM IFIP USENIX 2007 International Conference on Middleware.*, New York, NY, November 2007. Springer-Verlag.

[26] Y. Zhang, A. Juels, A. Oprea, and M. Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *the 2011 IEEE Symposium on Security and Privacy (Oakland)*, Oakland, CA, May 2011.