Hypervisor Introspection: Exploiting Timing Side-channels against VM Monitoring

Gary Wang, Zachary Estrada, Cuong Pham, Zbigniew Kalbarczyk, Ravishankar Iyer
University of Illinois at Urbana-Champaign
Email: {glwang2, zestrad2, pham9, kalbarcz, rkiyer}@illinois.edu

I. Introduction

Hypervisor activity is designed to be hidden from guest Virtual Machines (VM) as well as external observers. In this paper, we demonstrate that this does not always occur. We present a method by which an external observer can learn sensitive information about hypervisor internals, such as VM scheduling or hypervisor-level monitoring schemes, by observing a VM. We refer to this capability as *Hypervisor Introspection* (HI).

HI can be viewed as the inverse process of the well-known Virtual Machine Introspection (VMI) technique. VMI is a technique to *extract VMs' internal state from the hypervisor*, facilitating the implementation of reliability and security monitors[1]. Conversely, HI is a technique that allows *VMs to autonomously extract hypervisor information*. This capability enables a wide range of attacks, for example, learning a hypervisor's properties (version, configuration, etc.), defeating hypervisor-level monitoring systems, and compromising the confidentiality of co-resident VMs. This paper focuses on the discovery of a channel to implement HI, and then leveraging that channel for a novel attack against traditional VMI.

In order to perform HI, there must be a method of extracting information from the hypervisor. Since this information is intentionally hidden from a VM, we make use of a *side channel*. When the hypervisor checks a VM using VMI, VM execution (e.g. network communication between a VM and a remote system) must pause. Therefore, information regarding the hypervisor's activity can be leaked through this suspension of execution. We call this side channel the *VM suspend side channel*, illustrated in Fig. 1. As a proof of concept, this paper presents how correlating the results of in-VM micro-benchmarking and out-of-VM reference monitoring can be used to determine when hypervisor-level monitoring tools are vulnerable to attacks.

II. Finding a Hypervisor Introspection Channel

Although virtualization aims to isolate VMs, it has been demonstrated that an attacker can determine co-residency of two Amazon EC2 instances through the network [2] and detect hypervisors remotely [3]. These examples relate to the work presented here, however this work goes beyond learning information about a remote VM or the existence of a hypervisor. Our project aims to use the remote VM as a tool to observe and acquire sensitive information about the hypervisor on which the remote VM is running.

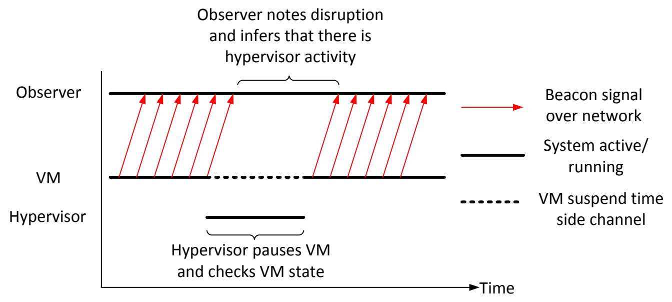
Observer notes disruption and infers that there is hypervisor activity
Observer
VM
Hypervisor
Hypervisor pauses VM and checks VM state
Time
Beacon signal over network
System active/ running
VM suspend time side channel

Fig. 1. Illustration of the VM suspend side channel

We noted that network communication would be disrupted when the VM is suspended by the hypervisor, and these disruptions could be observed by an external observer. This resulted in the discovery of the aforementioned VM suspend side channel.

One challenge with this approach is finding and developing a low latency network utility that can measure the monitoring interval of VMI passive monitoring frameworks. Typical network sockets that operate in user space spend too much time receiving a packet to keep up with how quickly packets can be sent out from user space. Thus, we are looking into ways to reduce the time it takes to receive a packet. We are currently developing a kernel space talker and listener (Linux kernel module) to achieve this goal.

Another challenge we will face further into the project is determining what kind of information we can get out of the network traffic generated by the network side channel.

III. Usage of Hypervisor Introspection

In order to thoroughly understand the limit of HI, we also consider different levels of a user's control over the target VM. For example, the user can have the highest privilege on the target VM, e.g., root access and kernel modification. There is also the case that the HI user has limited access to the target VM. For example, if an attacker wishes to escalate his or her privilege in a VM, he or she can only execute user-level processes. This second scenario poses more challenges in designing the HI.

In this fast abstract, we focus on a malicious use of the side channel as an attack vector on remote systems, but we note that the side channel can also be used for legitimate reasons such as remote monitoring and learning more about a specific hypervisor. Because we are using this side channel as

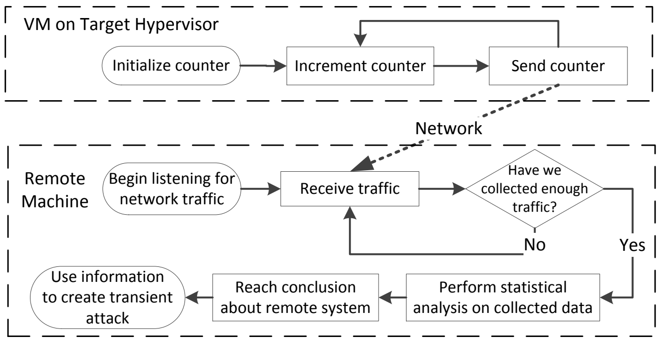
VM on Target Hypervisor
Initialize counter
Increment counter
Send counter
Network
Remote Machine
Begin listening for network traffic
Receive traffic
Have we collected enough traffic?
No
Yes
Perform statistical analysis on collected data
Reach conclusion about remote system
Use information to create transient attack

Fig. 2. An example attack. The attacker utilizes a VM on the target hypervisor and a separate remote machine to measure disruptions in network traffic and the counter values transmitted by the VM. The attacker analyzes this data offline to determine information about the target hypervisor, and then uses this information to create transient attack against similarly configured hypervisors.

an attack vector, we must make reasonable assumptions about the capabilities of the attacker. The attacker can be thought to have either partial or full control of the victim VM. If the attacker has full control of the VM, he can install whatever software he wants, including high privilege code running at the kernel level. Conversely, if the attacker only has partial control of the VM, he would likely only be able to run user space code with limited functionality.

IV. Sample Attack Against VMI

A. Attack Model

Our attack model assumes that an attacker has full control (i.e., root access and the ability to modify the guest OS) of a VM. The attacker wishes to use that VM to extract information from the underlying hypervisor, such as the monitoring interval and duration of any hypervisor-level monitoring. In order to extract this information, he or she needs to establish a network connection between the VM and a remote machine, which uses a high-resolution clock to record accurate timestamps.

In a cloud environment, one can expect these VMs to run on hypervisors with the same configuration as the hypervisor hosting the attacker's VM. The information obtained by HI enables the attacker to perform more sophisticated attacks on high-value neighboring VMs. One such attack is a transient attack, where malicious activity occurs between monitoring checks.

B. Initial Design

To determine the monitoring interval, the attacker must measure the VM suspend time. A naïve method is to record the arrival rate of signals being sent from the victim VM to the attacker. A break in the signals being sent could be indicative of a VM suspend initiated by the hypervisor. Observing these beacons directly gives insight into the victim VM, but these measurements are highly sensitive to environmental effects, e.g., network switch latency or routing policies.

Another method for capturing VM suspends is to have the victim VM perform some computation and determine how often it was interrupted by the hypervisor. The victim VM can continuously increment a counter and then transmit the counter value periodically. The attacker receives these counter values and records the highest value received. A lower value (i.e. fewer increments) corresponds to more VM suspends while a higher value (i.e. more increments) corresponds to fewer VM suspends. This approach relies on quick transmission of the counter value to the attacker.

Although this counter value does not tell us the exact monitoring interval, it can reveal the overhead introduced by the hypervisor. It is also possible to increase the time it takes for the monitoring to check the VM by performing a spamming attack on the VM [1]. We believe we can determine the monitoring interval by analyzing the values and arrival rates of the counter while performing a spamming attack. Fig. 2 illustrates the attack at a high level.

We obtained a baseline counter transmission rate by implementing a naïve user space TCP client/server. This naïve approach showed that the TCP could send the counter value every 268 μs. This transmission rate is too low to detect VM suspends, which occur on the order of tens of microseconds [4]. In order to increase the transmission rate, we are implementing a kernel level client/server to remove context switch overhead when transmitting the counter value.

V. Future Work

This fast abstract details our first steps towards using HI to exploit hypervisor-level monitoring frameworks. Once our kernel space client/server is implemented, we will obtain baseline measurements on the time intervals we can accurately detect and measure. This baseline will provide an understanding that VMI passive monitoring tools will need to perform their checks quickly, so as they cannot be detected through a VM suspend side channel.

We also plan to explore legitimate uses for HI, such as independent assessments of the QoS of rented cloud infrastructure or detection of insider attacks launched by super users of the cloud infrastructure.

VI. Acknowledgments

This material is supported by the NSF under Grant No. CNS 10-18503 CISE, by the NSA under Award No. H98230-14-C-0141, and by the AFRL/AFOSR under agreement No. FA8750-11-2-0084.

References

[1] C. Pham, Z. Estrada, P. Cao, Z. Kalbarczyk, and R. Iyer, "Reliability and security monitoring of virtual machines using hardware architectural invariants," *to appear in* Annual IEEE/IFIP International Conference on Dependable Systems and Networks (2014). IEEE/IFIP, 2014.

[2] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 199–212.

[3] J. Franklin, M. Luk, J. M. McCune, A. Seshadri, A. Perrig, and L. van Doorn, "Remote detection of virtual machine monitors with fuzzy benchmarking," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 3, pp. 83–92, Apr. 2008. [Online]. Available: http://doi.acm.org/10.1145/1368506.1368518

[4] B. D. Payne, M. Carbone, M. Sharif, and W. Lee, "Lares: An architecture for secure active monitoring using virtualization," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 233–247.