HInjector: Injecting Hypercall Attacks for Evaluating VMI-based Intrusion Detection Systems

Aleksandar Milenkoski[1], Bryan D. Payne[2], Nuno Antunes[3], Marco Vieira[3], and Samuel Kounev[1]

[1] Karlsruhe Institute of Technology, Germany (e-mail: milenkoski, kounev@kit.edu)

[2] Nebula Inc., USA (e-mail: bdpayne@acm.org)

[3] University of Coimbra, Portugal (e-mail: nmsa, mvieira@dei.uc.pt)

1. INTRODUCTION

Virtual machine introspection (VMI) is a mechanism for monitoring the states of guest virtual machines (VMs) from a virtualization host. Its use for intrusion detection is an emerging trend that brings many benefits, such as the possibility to monitor guest VMs in a transparent manner for attackers. However, a major issue is the evaluation of the attack detection accuracy of VMI-based intrusion detection systems (IDSes), e.g., [1], for detecting virtualization-related attacks, such as attacks targeting virtual machine monitors (VMMs). VMMs' attack surfaces are very narrow making them difficult to penetrate. As a result there are only a few publicly available exploits that demonstrate attacks against VMMs. We believe that the automated artificial injection of malicious system activities with respect to representative attack models is a promising approach towards overcoming this issue.

A malicious VM user may execute an attack against the underlying VMM via several attack vectors such as device drivers, VM exit events, or hypercalls. Hypercalls are software traps from a kernel of a paravirtualized guest VM to the VMM. They enable the intrusion in VMMs, initiated from a malicious guest VM kernel, in a procedural manner through VMMs' well-defined hypercall interfaces. This makes the automated injection of hypercall attacks feasible. Further, given the potential high severity of hypercall attacks, the evaluation of the effectiveness of IDSes for detecting and preventing such attacks is crucial. The exploitation of a vulnerability in a VMM's hypercall handler typically leads to altering the VMM's memory enabling, for example, the execution of malicious code with VMM privileges.

In this paper, we present *HInjector*, a customizable framework for injecting hypercall attacks during regular operation of a paravirtualized guest VM in a Xen-based environment. Besides the fact that it is a popular VMM, we chose Xen as a VMM targeted by injected hypercall attacks because most of the documented vulnerabilities of VMMs' hypercall handlers are those of Xen. This enables the construction of representative attack models. Further, Xen is open-source software and thus its codebase can be modified, which, as shown later, is necessary for injecting hypercall attacks.

The attacks injected by HInjector conform to attack models based on existing Xen vulnerabilities (e.g., CVE-2008-3687, CVE-2012-5513). The goal of HInjector is to exercise the sensors of a typical VMI-based IDS. An IDS may use VMI to monitor guest VM's virtual CPU (vCPU) registers in order to identify, for example, anomalous hypercall parameter values. It may also monitor guest VM's memory and inspect, for example, call stacks to identify irregular hypercall call sites. Since VMI-based IDSes do not monitor states of VMMs, HInjector does not inject the effects that hypercall attacks may have on the state of a given attacked VMM (e.g., altered VMM's memory).

To the best of our knowledge, we are the first to consider the injection of (hypercall) attacks targeting VMMs. Pham et al. [4] and Le et al. [2] focus on injecting generic software faults directly into VMMs. The latter is not suitable for evaluating VMI-based IDSes since, as mentioned above, such IDSes do not monitor states of VMMs. Also, generic software faults are not representative of faults caused by exploiting security vulnerabilities. Oyama et al. [3] present a framework for injecting effects of attacks targeting guest VMs, however, they do not take into account hypercall attacks and attacks targeting VMMs in general.

2. HINJECTOR DESIGN

We constructed attack models based on analyzing previously discovered vulnerabilities of Xen's hypercall handlers. We analyzed relevant publicly disclosed vulnerability reports, the earliest dating back to 2008, identifying patterns of VM activities for executing hypercall attacks. We then categorized the identified patterns into attack models. To construct attack models, we also took into consideration hypercall characteristics relevant for intrusion detection. We distinguish the following attack models:

i) Invoking hypercalls from irregular call sites. VMI-based IDSes (e.g., [1]) may consider hypercalls invoked from call sites unknown to them, e.g., an attacker's loadable kernel module (LKM), as malicious. We model the typical scenario where an attacker loads a LKM to execute attacks from kernel mode, which includes hypercall attacks.

ii) Invoking hypercalls with anomalous parameter values a) outside the valid value domains, or b) crafted for exploiting specific vulnerabilities (not necessarily outside the valid value domains). This attack model is based on the Xen vulnerabilities described in CVE-2008-3687, CVE-2012-3516, CVE-2012-5513, and CVE-2012-6035.

iii) Invoking a series of hypercalls in irregular order, including repetitive execution of a single or multiple hypercalls. This attack model is based on the Xen vulnerability described in CVE-2013-1920. The repetitive execution of hypercalls, for example, requesting system resources, is an easily feasible attack which may lead to resource exhaustion

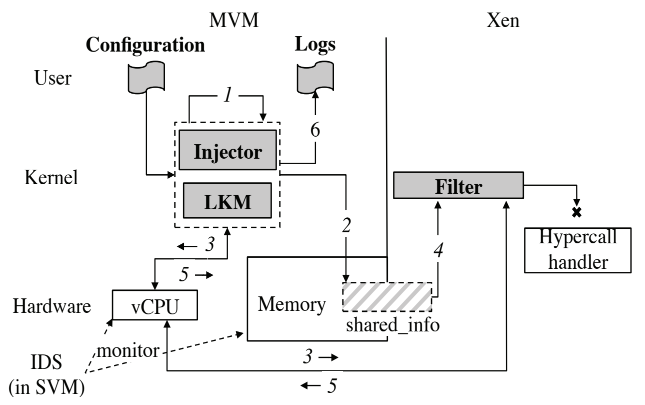

Figure 1: Architecture of HInjector

of collocated VMs.

There is a possibility that a guest VM's kernel may execute hypercalls with parameter values outside the valid value domains or in irregular order due to generic faults that are not of a malicious nature. Currently we assume that the kernels of the guest VMs running in the virtualized environment where HInjector is deployed work correctly. The interference of the execution of anomalous hypercalls, triggered by generic faults, with the operation of HInjector warrants further study.

In Figure 1, we depict the architecture of HInjector, which consists of the components *Injector*, *LKM*, *Filter*, *Configuration*, and *Logs*. We refer to the VM injecting hypercall attacks as malicious VM (MVM). The IDS under test is deployed in a secured VM (SVM) collocated with MVM, which is normally the virtualization host VM or Domain0, as reffered to in Xen terminology. We implemented a preliminary version of HInjector using Xen 4.2.2 on Debian 7.1 as virtualization host and Debian 6.0.7 as MVM.

The *Injector*, deployed in the hypercall interface of MVM's kernel, intercepts hypercalls invoked by the kernel during regular operation and modifies hypercall parameter values on-the-fly making them anomalous. The Injector is used for injecting hypercalls invoked from a regular call site.

The *LKM*, a module of MVM's kernel, invokes regular hypercalls, hypercalls with anomalous parameter values, or hypercalls in irregular order. The LKM is used for injecting hypercalls invoked from an irregular call site.

The *Filter*, deployed in Xen's hypercall interrupt handler (i.e., 0x82 interrupt), identifies hypercalls injected by the Injector or the LKM, blocks their execution, and returns a valid error code. The latter is important for preventing MVM crashes by allowing the control flow of MVM's kernel to handle failed hypercalls that have been invoked by it. The Filter blocks the execution of Xen's hypercall handlers to prevent Xen crashes. The Filter identifies injected hypercalls based on information stored by the Injector/LKM in the *shared_info* structure, a memory region shared between a guest VM and Xen. To this end, we extended *shared_info* with a string field named *hid* (hypercall identification).

The *configuration* is a set of user files containing configuration parameters for managing the operation of the Injector and the LKM. Currently, it allows for specifying duration of an injection campaign, valid parameter value domains and/or specifically crafted parameter values for a given hypercall (relevant to the Injector and the LKM), and valid order of a series of hypercalls (relevant to the LKM). We are implementing support for specifying workload profile parameters such as average injection rate of a specific hypercall, total average injection rate, and temporal distribution of injection actions.

The *logs* are user files containing records about injected hypercalls, i.e., hypercall IDs (hypercall identification numbers assigned by Xen) and parameter values, as well as timestamps. The logged data serves as reference data (i.e., as "ground truth") used for calculating IDS attack detection accuracy metrics. We are currently implementing logging of statistics about injection campaigns such as achieved total average injection rate and temporal distribution of injected hypercalls.

In Figure 1, we depict the steps involved in injecting a single hypercall by the Injector/LKM. An illustrative example of the Injector injecting a hypercall with a parameter value outside of its valid domain is as follows: 1) The Injector intercepts a hypercall invoked by MVM's kernel and replaces the value, for example, of the first parameter, with a generated value outside the parameter's valid value domain specified in the configuration; 2) The Injector stores the ID of the hypercall, the number of the parameter with anomalous value (i.e., 1), and the parameter value itself in *hid*; 3) The Injector passes the hypercall to MVM's vCPU which then issues a 0x82 interrupt and passes control to Xen; 4) The Filter, using the data stored in *hid*, identifies the injected hypercall when it arrives at Xen's 0x82 interrupt handler; 5) The Filter returns a valid error code without invoking the hypercall's handler; 6) After the return code arrives at MVM's kernel, the Injector stores in the log files, the ID and parameter values of the injected hypercall, and a timestamp.

As part of our future work, we plan on identifying and defining representative characteristics of the hypercall attacks that can be injected by HInjector (e.g., parameter values, hypercall order). The identification and definition of representative characteristics of hypercall attacks is a new and challenging research direction. It is challenged primarily by the lack of publicly available and detailed technical information on vulnerabilities of VMMs' hypercall handlers as well as on hypercall attacks performed in practice. As such information becomes available, we plan to refine and extend the features of HInjector, and to provide readily available configuration files for injecting representative attacks. We also plan on extending the customizability of HInjector to enable the injection of IDS evasive hypercall attacks, for example, mimicry attacks.

3. REFERENCES

[1] S. Bharadwaja, S. Weiqing, M. Niamat, and S. Fangyang. Collabra: A Xen Hypervisor Based Collaborative Intrusion Detection System. In *Proc. of ITNG 2011*, pages 695–700, 2011.

[2] M. Le, A. Gallagher, and Y. Tamir. Challenges and Opportunities with Fault Injection in Virtualized Systems. In *VPACT*, 2008.

[3] Y. Oyama and Y. Hoshi. A Hypervisor for Injecting Scenario-Based Attack Effects. In *Proc. of COMPSAC 2011*, pages 682–687, 2011.

[4] C. Pham, D. Chen, Z. Kalbarczyk, and R. Iyer. CloudVal: A Framework for Validation of Virtualization Environment in Cloud Infrastructure. In *Proc. of DSN 2011*, pages 189–196, 2011.