Efficient and Fine-Grained VMM-Level Packet Filtering for Self-Protection

Kenichi Kourai, Kyushu Institute of Technology, Fukuoka, Japan
Takeshi Azumi, Tokyo Institute of Technology, Tokyo, Japan
Shigeru Chiba, The University of Tokyo, Tokyo, Japan

ABSTRACT

In Infrastructure-as-a-Service (IaaS) clouds, stepping-stone attacks via hosted virtual machines (VMs) are critical for the credibility. This type of attack uses compromised VMs as stepping stones for attacking the outside hosts. For self-protection, IaaS clouds should perform active responses against stepping-stone attacks. However, it is difficult to stop only outgoing attacks at edge firewalls, which can only use packet headers. In this paper, we propose a new self-protection mechanism against stepping-stone attacks, which is called xFilter. xFilter is a packet filter running in the virtual machine monitor (VMM) underlying VMs and achieves pinpoint active responses by using VM introspection. VM introspection enables xFilter to directly obtain information on packet senders inside VMs. On attack detection, xFilter automatically generates filtering rules based on packet senders. To make packet filtering with VM introspection efficient, we introduced several optimization techniques. Our experiments showed that the performance degradation due to xFilter was usually less than 16%.

Keywords: Cloud Computing, Packet Filtering, Operating Systems, Outgoing Attacks, Virtual Machines

INTRODUCTION

Cloud computing is rapidly emerging in recent years. Among various types of clouds, infrastructure as a service (IaaS) such as Amazon EC2 (Amazon, Inc., 2006) provides virtual machines (VMs) for the users. The users can use their VMs on demand. Unfortunately, it is not guaranteed that the systems inside VMs are always well-maintained. If the outside attackers compromise such VMs, they can mount attacks to the outside hosts via the VMs, which is known as stepping-stone attacks (Staniford-Chen & Heberlein, 1995). For example, the attackers may perform portscans and denial-of-service (DoS) attacks to the outside hosts.

Therefore, self-protection against such attacks is indispensable for IaaS clouds. If a VM in an IaaS cloud is used as a stepping stone for attacking the outside hosts, not only...
the VM’s user but also the IaaS provider may have a responsibility for the attack. The IaaS provider also becomes an attacker as well as a victim. If an IaaS cloud detects outgoing attacks, it should perform active responses against the attacks. One of the methods for active responses is updating firewall rules. Typically, new rules are added to edge firewalls in the IaaS cloud. The rules block the packets for the attacks from the compromised VM.

Such a self-protection mechanism should stop only outgoing attacks. However, active responses performed at edge firewalls are not pinpoint because edge firewalls can filter packets on the basis of only information contained in the packets. For example, edge firewalls would have to block all the packets from the compromised VM to stop portscans. Even when the system is partially compromised, all the applications and users cannot send any packets to the outside. Pinpoint active responses are beneficial to not only IaaS users but also providers because IaaS providers can easily stop suspicious communication without excessive fears of false positives.

In this paper, we propose a new self-protection mechanism, named xFilter, for IaaS clouds. xFilter is a packet filter running in the virtual machine monitor (VMM), which is underneath VMs. To achieve pinpoint active responses, xFilter obtains information on packet senders by using a technique called VM introspection (Garfinkel & Rosenblum, 2003). VM introspection enables xFilter to inspect the memory of VMs and access data in guest operating systems without interacting with them. Using information on sender processes, xFilter can deny only packets sent from particular processes or users. When xFilter detects an outgoing attack, it automatically identifies the attack source and generates a new filtering rule to stop the stepping-stone attack. In addition, xFilter provides development support of its modules because it is difficult to develop software performing VM introspection in the VMM.

xFilter is performance-critical because it performs VM introspection in the middle of packet transmission. To reduce the overheads of VM introspection, we introduced several optimization techniques. First, we embedded the component for introspecting VMs into the VMM of Xen (Barham et al., 2003) although VM introspection is usually performed in the privileged VM (Jiang, Wang, & Xu, 2007; Payne, Carbone, & Lee, 2007; Payne, Carbone, Sharif, & Lee, 2008). Second, xFilter performs optimized sender traversal to find sender processes so that the number of kernel objects to be introspected is minimized. Third, xFilter performs decision cache to reuse filtering decisions without VM introspection. Fourth, two-phase attack detection minimizes the overheads under no attack symptoms. Thanks to these techniques, performance degradation due to xFilter was less than 16% in usual cases.

This paper is an extended version of our previous paper (Kourai, Azumi, & Chiba, 2012). In this paper, we describe four optimization techniques for xFilter explicitly and in further detail, particularly for optimized sender traversal and two-phase attack detection. In addition, we introduce development support for xFilter and its optimization, and we report the performance in the development phase. Moreover, we add new experiments for optimized sender traversal and raw sockets. We also explain the implementation details of our portscan detector.

BACKGROUND

For IaaS clouds, a fine-grained self-protection mechanism is needed to stop only stepping-stone attacks via VMs. In other words, VMs should be able to provide services continuously as much as possible. Even if the system inside a VM is partially compromised, the rest is usually not compromised. For example, when the Apache web server is compromised, only the privileges of the user www-data are taken over at worst. The other applications are still running legitimately because the user www-data cannot interfere with the other users’ processes. Therefore, such legitimate applications should be able to communicate with the outside hosts.
The World Code: Mathematical Ontology as the Real Road to Reality
www.igi-global.com/chapter/world-code-mathematical-ontology-real/28310?camid=4v1a