AL-SAFE: A Secure Self-Adaptable Application-Level Firewall for IaaS Clouds

Anna Giannakou, Louis Rilling, Christine Morin, Jean-Louis Pazat

AL-SAFE: A Secure Self-Adaptable Application-Level Firewall for IaaS Clouds

Anna Giannakou, Inria, IRISA
Louis Rilling, DGA
Christine Morin, Inria, IRISA
Jean-Louis Pazat, INSA, IRISA

1. Introduction

Some of the key IaaS clouds features are multi-tenancy, elasticity and on demand availability. In a modern virtualised environment dynamic resource management allows tenants to create, destroy or reconfigure virtual resources with unprecedented ease.

However, the same characteristics that make cloud environments agile and dynamic, also affect the ability of a security monitoring framework to successfully detect attacks in outsourced information systems [6] and sometimes introduce new security vulnerabilities.

Large scale security frameworks include various components (firewalls, intrusion detection systems, log collectors etc.) that have different functionalities and are located in different areas or even outside the virtual infrastructure. Consequently, changes in the number of virtual machines (e.g. addition of new instances) or in their placement (e.g. live migration) require the security monitoring system to be adapted. To be able to successfully cope with the high frequency of occurring changes, a cloud-tailored security monitoring infrastructure should be able to adapt its components with little to no human intervention. In opposition to a typical host- or network-level firewall which filters network traffic based on a list of rules that use IP addresses and ports, application-level firewalls operate based on an access policy that is defined from a white list of processes that are allowed to make connections. This fine-grained level of filtering is achievable because application-level firewalls have a complete overview of the host in which they are running. Unfortunately, the conventional design of application-level firewalls has an inherent deficiency: limited isolation between the firewall and vulnerable applications which increases the probability of a successful attack that disables the firewall (i.e removing hooks from packet filtering functions). Hence, an important question is: *Can we retain the same level of increased visibility while limiting the attack surface between any infected application and a trusted application-level firewall?*

To address this question we designed and implemented AL-SAFE a two-level application-level firewall that operates outside of the virtual machine it is monitoring in a completely separate domain. By leveraging virtual machine introspection we retain the same level of ”inside-the-host” visibility while introducing a high-confidence barrier between the firewall and the attacker's malicious code.

Our firewall consists of two different components that operate at distinct infrastructure levels: an edge firewall responsible for filtering network traffic between the outside world and the cloud infrastructure, and a virtual switch-level firewall that filters traffic in the virtual switch of each physical host. Both components, executed outside the untrusted virtual machine, become application-level firewalls by using virtual machine introspection.

The rest of the paper is organised as follows: Section 2 presents related work. Section 3 outlines our frameworks' objectives and architecture. Section 4 focuses on implementation details while Section 5 concludes the paper focusing on evaluation and future work.

2. Related Work

Virtual machine introspection is a mechanism that allows, through memory mapping, indirect inspection of and control over the current state of a virtual machine from software running outside

of the virtual machine. The approach is based on building higher-level semantics that can be accessed by a monitoring application (data structures, files) from the mapped memory pages. In this section we discuss various security applications that leverage the isolation properties offered by virtual machine introspection.

Garfinkel and Rosenblum [1] were the first ones to introduce the concept of virtual machine introspection, and used it to implement a secure host-based IDS. Livewire enables an IDS that is located in the physical host to monitor the state of the operating system in a VM by mapping the guest's physical page frames to its own address space.

Joshi et al [2] created a system called IntroVirt that can assess whether a system has been compromised through a known pre-existing vulnerability based on introspection and replay.

Both Livewire and IntroVirt incorporate virtual machine introspection for designing an IDS hence providing passive network monitoring and not addressing dynamic changes in the virtual infrastructure.

Amazon EC2 provides IaaS cloud tenants with firewalls, named security groups [5], that are located outside the virtual machines and offer protection from attacks originating outside the cloud infrastructure by filtering inbound traffic only. The security groups are oblivious to the type and nature of the tenant applications thus making fine-grained application-based network traffic filtering impossible. Furthermore it is unclear whether the cloud provider adds rules to the security groups.

The authors of xFilter [3] utilize virtual machine introspection for creating a self protection mechanism against stepping-stone-attacks [4] that filters outgoing packets at the level of the hypervisor. xFilter cannot handle cloud oriented dynamic events (such as VM migration) and only filters outgoing traffic.

Our goal is to design a self-adaptable introspection-based application level firewall that can filter inbound and outgoing traffic based on a list of legitimate applications that are allowed to make connections. We leverage virtual machine introspection for retaining the high degree of visibility of an application level firewall while making it tamper resistant to any user- or kernel-level malware that may be present in the monitored VM.

3. Overiew of AL-SAFE

We propose a two-level introspection-based application level firewall with a self-adaptable ruleset. The reconfiguration of the enforced ruleset depends upon changes in two different layers: Service layer (addition or removal of services) and Virtual Infrastructure layer (e.g. VM creation, deletion or migration). In this section we outline AL-SAFE objectives together with our threat model and a detailed description of AL-SAFE architecture.

3.1 AL-SAFE Objectives

AL-SAFE should satisfy the following properties:

- Self adaptation: the enforced ruleset should be configured with respect to dynamic events that occur in a cloud environment. These high frequency events include modifications in the virtual infrastructure such as VM creation, deletion and migration.
- Service-based customisation: the ruleset on both firewall levels should be reconfigured in order to allow network traffic that originates from the legitimate tenant-approved services.
- Tamper-resistance: our two-level firewall should continue to function reliably and block all potentially malicious connections even if an attacker gains control of a monitored VM. In particular, the reconfiguration of the enforced ruleset should not rely on information originating from components installed inside the monitored guest as these can be unreliable.
- Cost minimisation: the overall cost in terms of resource consumption must be kept at a minimal level both for the tenants and the provider. AL-SAFE achieves this by enabling sharing of the two-level firewall between tenants.

3.2 Threat model

We consider software attacks only that originate both from malicious VMs as well as from outside the cloud infrastructure. However we con-

sider that the provider infrastructure cannot be corrupted.

3.3 Two-level Firewall Architecture

AL-SAFE is a secure, introspection-based, two-level, application level firewall. It is part of our self-adaptable security monitoring framework for IaaS clouds [12]. AL-SAFE rulesets are automatically reconfigured upon the occurrence of changes in two different levels: the cloud infrastructure (e.g. VM migration, creation or deletion) and the list of services running inside the monitored VM. AL-SAFE consists of five major components as depicted in figure 1: the snapshotting/introspection component (VMI), the Information Extraction Agent (IEA), the Rule Generators (RG) and two separate firewalls. An edge firewall (EF) filters network traffic between the outside world and the cloud infrastructure, and a virtual switch-level firewall (SLF) filters traffic in the virtual switch of each physical host. The components are run by the cloud provider.

The flow of the process which is executed periodically is as follows: first, the VMI takes a snapshot of the monitored guest and introspects the memory image in order to obtain the list of processes attempting to make network connections. Second, the IEA extracts all the necessary information for generating filtering rules and propagates it to two separate Rule Generators. Finally the RGs create the switch-level and edge firewall rules and inject them in the separate firewalls. The IEA takes as a parameter a tenant defined white list of processes that are allowed to send and receive packets.

The *Snapshotting/Introspection* component takes a snapshot of the VM in order to provide a coherent memory view to the introspection process that will follow. The component iterates over the list of running processes and checking whether a file in the list of open files per process corresponds to a network socket. If this is true it extracts the process name, the pid as well as source and destination port, IP and protocol. The time elapsed between two consecutive snapshots/introspections is defined by the tenant and can be dynamically adapted by the Adaptation Manager [12].

The *Information Extraction Agent* compares the list of processes that attempt to send/receive network packets with a white list of processes that are allowed to make connections. The Adaptation Manager [12] is responsible for sharing the white list with the Information Extraction Agent through a secure channel. The IEA allows the connection if it finds a match and blocks the connection otherwise. The IEA propagates the connection information together with an allow or block action to the *Rule Generators*. Each RG creates the corresponding rule utilising all available information such as source port, source IP address, destination port, destination IP address and protocol. The generated rules are then injected in the separate firewalls.

The *Switch-level firewall* filters network packets at the level of virtual switch based on a predefined list of rules. If the rule's action is to block the connection it drops the packet otherwise the packet is injected in the network. A switch-level filtering offers the possibility of blocking malicious traffic originating from inside the cloud infrastructure at an early stage thus significantly reducing the load of monitored traffic in the remaining security probes.

The *Edge firewall* is located at the edge of the virtual infrastructure in a separate network probe. It filters traffic based on a set of rules that are responsible for mediating the traffic directed to and from all the VMs in the virtual infrastructure. By placing the edge firewall in a stand alone network device we ensure that external malicious traffic will be blocked at an early stage while there will be no CPU penalties in the deployed VMs.

4. Implementation

We implemented a prototype of AL-SAFE using KVM hypervisor and a Linux guest operating system in a private cloud. We used OpenStack [7] as the cloud management system and Open vSwitch [8] as a multilayer virtual switch. A description of the implementation for each component of AL-SAFE follows.

- For the snapshotting/introspection component we utilized the LibVMI [10] introspection framework. For performing the correlation between running processes and open sockets in-

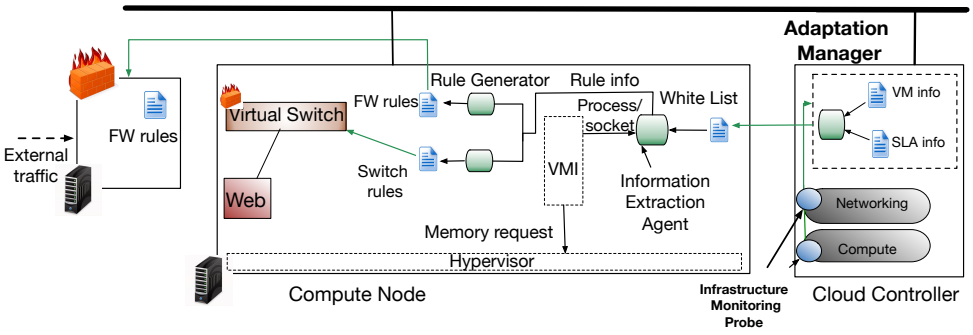

Figure 1. The AL-SAFE architecture

side the monitored VM we deployed Volatility Memory Forensics [11] framework.

- For the switch-level firewall we utilized the filtering capabilities offered by Open vSwitch while for the edge firewall we deployed Nftables [9] packet filtering framework in a stand alone Linux host.
- Both the Information Extraction Agent and the Rule Generators were implemented in python. The switch level rule generator produces OpenFlow filtering rules while the edge firewall rule generator produces Nftables-compatible rules. The created rulesets are injected in parallel in the EF and SLF firewalls.

5. Conclusion and Future Work

We have designed a secure application level firewall that is located outside the VM but is able to retain through virtual machine introspection, the same degree of visibility as an ”inside-the-host” solution. Our firewall filters traffic at two distinct points in the virtual infrastructure and is able to adapt the enforced ruleset based on changes in the virtual infrastructure as well as in the list of services running inside the monitored VMs.

We are currently conducting a thorough evaluation of our approach focusing on both performance and correctness aspects. We plan to address cost minimisation by combining the security monitoring of the tenants and provider infrastructure. We also intend to expand our architecture by including other types of devices such as log collectors and aggregators.

References

[1] T. Garfinkel et al. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *Proc. NDSS*, 2003.

[2] A. Joshi et al. Detecting Past and Present Intrusions Through Vulnerability-specific Predicates. In *Proc. SOSP*, 2005.

[3] K. Kourai et al. A Self-Protection Mechanism against Stepping-Stone Attacks for IaaS Clouds. In *UIC/ATC*, 2012.

[4] S. Staniford-Chen et al. Holding intruders accountable on the Internet. In *Proc. Security and Privacy*, 1995.

[5] Amazon inc. `https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf`

[6] N. Shirazi et al. Assessing the impact of intra-cloud live migration on anomaly detection. In *Proc. CloudNet*, 2014.

[7] OpenStack, version Juno. `http://www.openstack.org/`.

[8] Open vSwitch, version 2.9. `http://openvswitch.org/`.

[9] Nftables version 0.5. `http://netfilter.org/projects/nftables/`.

[10] LibVMI version 0.12. `https://github.com/libvmi/libvmi/releases`.

[11] Volatility Memory Forensics version 2.4. `http://www.volatilityfoundation.org`.

[12] A. Giannakou et al. Towards Self Adaptable Security Monitoring in IaaS Clouds. In *Proc. CCGRID*, 2015.