CryptVMI: Encrypted Virtual Machine Introspection in the Cloud

Fangzhou Yao, Roy H. Campbell
Department of Computer Science
University of Illinois at Urbana-Champaign
{yao6, rhc}@illinois.edu

***Abstract*—Virtualization techniques are the key in both public and private cloud computing environments. In such environments, multiple virtual instances are running on the same physical machine. The logical isolation between systems makes security assurance weaker than physically isolated systems. Thus, Virtual Machine Introspection techniques become essential to prevent the virtual system from being vulnerable to attacks. However, this technique breaks down the borders of the segregation between multiple tenants, which should be avoided in a public cloud computing environment. In this paper, we focus on building an encrypted Virtual Machine Introspection system, CryptVMI, to address the above concern, especially in a public cloud system. Our approach maintains a query handler on the management node to handle encrypted queries from user clients. We pass the query to the corresponding compute node that holds the virtual instance queried. The introspection application deployed on the compute node processes the query and acquires the encrypted results from the virtual instance for the user. This work shows our design and preliminary implementation of this system.**

I. Introduction

As cloud computing grows in popularity, many companies are moving to the cloud. With cloud computing, companies do not have to spend a significant portion of their time to build and maintain their computation and storage infrastructure. Virtualization has become inevitable in both public and private cloud computing solutions, such as Amazon EC2 [1] and OpenStack [2], because it provides better utilization of resources and reduces the cost by allowing multiple operating systems (OS) owned by multiple tenants to run concurrently on the same physical machine. Though multiple OSes are running in their own virtual machines (VM) and sharing the same hardware resources, the Virtual Machine Manager (VMM) can still ensure high availability for users [3].

However, even though a VM is designed to be only able to access resources in its own space, the traffic between VMs makes the isolation between systems no longer physical but logical, and hence the security guarantees are weaker [5]. A compromised VM can spread malware quickly and make the entire environment vulnerable to more attacks. This fact has led to the appearance of Virtual Machine Introspection (VMI) techniques. VMI tools inspect a VM from a trustworthy outside environment, and they are able to access the entire system state of a VM [6]. Thus, security systems like Intrusion Detection Systems (IDS) can be built with this technique over the cloud for greater attack resistance, while providing an excellent view of what is happening in the virtual instances [7].

There are many private cloud systems having already used VMI to enhance their security, but concerns are raised for public cloud systems, since this technique might break down the borders of the segregation between multiple tenants [4]. For instance, if Amazon started using VMI, then the administrators in Amazon would be able to access the entire state of every customer's VM running in their public cloud. This is definitely not what Amazon's customers want to happen with respect to their privacy.

To address this concern, we propose CryptVMI, an encrypted Virtual Machine Introspection system, to provide users complete status of their virtual instances, while keeping the confidentiality from administrators of the cloud. This system can be extended to build an IDS on the user end to provide better security for their system, especially when their cloud computing framework is running in the public cloud.

The rest of this paper is organized as follows. We first show our design in Section II for this encrypted VMI system. Next, we reveal our preliminary implementation details in Section III. In Section IV, we conclude our work and provide future approaches.

II. Design

Users can access their VMs through a Secure Shell (SSH) connection, but this approach only allows them to obtain data from the inside of VMs, while VMI is done on the host system, which is the outside. Furthermore, we do not want to expose the structure of our cloud system, such as on which compute node a VM is located, or the IP address of the compute node, namely the host system, since it might cause co-location attacks [8]. Our objective is to prevent the cloud administrator from knowing the entire state of a user's VM while keeping the transparency to users. We assume that the public cloud system has been configured, and hence administrators do not have `root` on compute nodes.

The design of CryptVMI has two major components. Figure 1 shows the overview of this system. The first component handles queries sent from users on the remote clients. Once the query is processed, it transfers the encrypted data back to the client, and the client decrypts the data for the user. The second component involves strategies to acquire the desired result with encryption. Each of these components is described in following subsections.

A. Query Handler

The VMI client on the remote end initiates the query request to the management node in the public cloud environment through a Secure Sockets Layer (SSL) connection.

978-1-4799-5063-8/14 $31.00 © 2014 IEEE
DOI 10.1109/CLOUD.2014.149

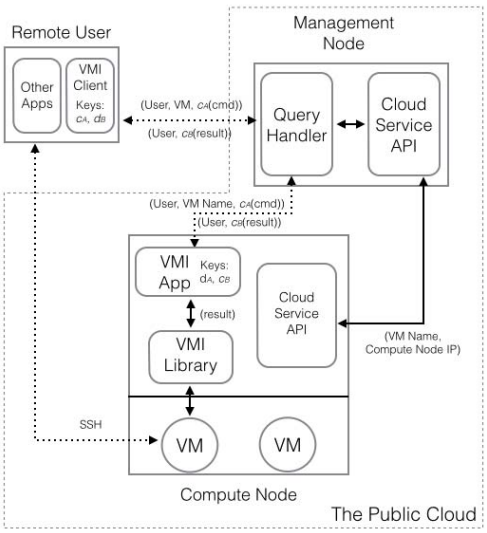

Fig. 1. This is the overall design of CryptVMI. The SSH connection is not part of the VMI system, but it is a general approach that users communicate with their VMs. Dotted lines represent secure connections or connections with encrypted components. There is only one remote user and one compute node shown, but multiple clients and compute nodes are supported in our design.

The query is divided into three parts, the user's credential, VM instance name in the cloud system and an encrypted command. Once the handler receives the query, it first checks the user's credential token and the instance name with the cloud service API to verify if this user has the access to the current tenant in the cloud and if there is such a VM associated with this tenant. Then, the handler uses the cloud service API to locate the IP address of the compute node that holds the designated VM, and obtains this VM's name to the hypervisor, which is different from the instance name provided from the cloud service API to the user. Communications in this process are not encrypted, because they are in the internal cloud system network, and the cloud administrator should have known those information.

Since the command is encrypted by the user's public key c_A, the details of the query will not be disclosed. This command is hence sent to the introspection application on the compute node, along with the user name and the name of the designated VM.

B. Introspection Application

The introspection application on the compute node uses stored private key d_A for this specific user to decrypt the command. Then, it translates the command and invokes corresponding VMI library API to acquire the introspection result. The result is encrypted with the public key c_B, which is also assigned to this specific user, and transferred back to the query handler. Finally, the query handler sends the encrypted result data back to the client, and the client decrypts the result with the private key d_B for the user.

III. Preliminary Implementation

We set up our experiment with OpenStack to simulate the public cloud environment as Amazon EC2. Thus, the virtual instances were running in para-virtualization with Xen hypervisor [9].

We built our VMI application on top of LibVMI, a virtual machine introspection library focused on reading and writing memory from VMs [10]. LibVMI has small performance impact and provides the introspection directly on virtual memory, It also provides functions for accessing CPU registers, pausing a VM and printing binary data. This fact also guarantees that we are able to write introspection functions for different payloads.

We built the communication between the remote client and the query handler in a RESTful fashion for two reasons. First, JSON is the standard data structure present in OpenStack API [2]. It simplifies the process to communicate with OpenStack. Second, it is the preferred method for distributed applications. Since we plan to develop the client application into an IDS, we want the result returned in a standard and extensible criteria.

IV. Conclusion and Future Work

In this paper, we showed our design and preliminary implementation of CryptVMI, an encrypted Virtual Machine Introspection system to keep the confidentiality in the cloud. We believe that this solution is able to address the concern of the multi-tenanacy public cloud.

In our presumption, we did not expect the situation that the administrator in a public cloud would have the direct full access to a compute node and hence exploit the VMI library. In future, we will discuss the above issue, as well as a more secure key distribution and storage framework. We also plan to integrate this encryption feature into LibVMI by modifying this library, instead of staying at the application level. Eventually, we will benchmark the performance and resistance of our system.

References

[1] Amazon Web Services, *Elastic Compute Cloud*, [Online]. Available: http://aws.amazon.com/ec2/

[2] Open Stack, *Open Source Software for Building Private and Public Clouds*, [Online]. Available: https://www.openstack.org/

[3] T. Burger, *The Advantages of Using Virtualization Technology in the Enterprise*, 2012 [Online]. Available: http://goo.gl/2oqZgo

[4] C. Brenton, *Introspection: Boon or Bane of Multitenant Security?*, 2012 [Online]. Available: http://goo.gl/5aIj6W

[5] M. Factor, D. Hadas, A. Hamama, N. Har'el, E. K. Kolodner, A. Kurmus, A. Shulman-Peleg and A. Sorniotti, *Secure Logical Isolation for Multi-tenancy in Cloud Storage*, 30th IEEE International Conference on Massive Storage Systems and Technology, 2013.

[6] K. Nance, B. Hay and M. Bishop, *Virtual Machine Introspection: Observation or Interference?*, IEEE Security and Privacy, 2008.

[7] T. Garfinkel and M. Rosenblum, *A Virtual Machine Introspection Based Architecture for Intrusion Detection*, 10th Annual Network and Distributed System Security Symposium, 2003.

[8] T. Ristenpart, E. Tromer, H. Shacham and S. Savage, *Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds*, 16th ACM Conference on Computer and Communications Security, 2009.

[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, *Xen and the Art of Virtualization*, ACM Special Interest Group on Operating Systems Review. Vol. 37(5), 2003.

[10] B. D. Payne, *Simplifying Virtual Machine Introspection Using LibVMI*, Sandia Report, 2012.