# ESI-Cloud: Extending Virtual Machine Introspection for Integrating Multiple Security Services

(+) Jiangchun Ren, (*) Ling Liu, (+) Da Zhang, (+) Huaizhe Zhou, (*) Qi Zhang
(+) School of Computer, National University of Defense Technology, Changsha, China
(*) School of Computer Science, College of Computing, Georgia Institute of Technology, Atlanta, USA
wwwrjc@163.com, lingliu@cc.gatech.edu, dadabop2003@163.com, zhz_alvin@foxmail.com, qzhang90@gatech.edu

***Abstract*—The cloud computing has become the most popular service model with a proven ability to reduce costs and improve resource efficiency. However cloud platform also suffer from potent attacks from internal and external environment. The cooperation with Managed Security Service Providers (MSSPs) has becoming a compelling solution to the security and privacy risks in the cloud platform. But it is difficult to deploy and manage a variety of security service efficiently and effectively. In this paper, we propose a novel framework, ESI-Cloud (Easy Security services Integration Cloud), to facilitate the integration of multiple managed security services into a cloud computing platform. ESI-Cloud utilizes virtual machine introspection to monitor and analyze the guest VMs execution information. It also provides a client library with rich APIs and a management console to the managed security service providers (MSSPs), with which multiple security services can be easily integrated into the basic cloud platform. We have implemented ESI-Cloud in the Xen hypervisor platform and evaluated the system functionality and performance.**

***Keywords—Cloud services; security services; VM introspection***

## I. INTRODUCTION

The cloud computing and services enhance flexibility, scaling and availability, and provide the potential for reducing ownership cost through utility-based computing. Although more and more businesses are moving into the clouds, the fast growth of cloud computing and cloud data centers has also raised increased concerns about the growing potential of security and privacy risks and vulnerabilities. Unfortunately, the security prevention in clouds is still in its infancy. On one hand, Infrastructure-as-a-Service (IaaS) is one of the dominating cloud service provisioning models. The guest VMs are considered the main source of security threats on the IaaS platform because the cloud provider is hosting VMs without being aware of their actual contents, and has no control over the execution of those VMs. Most of IaaS cloud providers offer the rental service for VMs and storage space without offering special security protection for guest VMs. Tenants must purchase and install different types of security software (such as anti-virus, IDS, etc.) by themselves to protect their VM executions and application services. Even those cloud providers who offer recovery service are not equipped with professional security measure, which could lead to inability to find malware prevent advanced attacks by malicious hackers.

Referring to the solutions in traditional servers, which have detached functions of business and security, some cloud providers begin to cooperate with the Managed Security Service Provider (MSSP). In this way could It is helpful to introduce professional security services into the cloud platforms. However, many problems remain open challenges:

1) The guest VMs in cloud platform are managed by tenants. The security solutions provided by MSSPs may become outdated when the VMs are in an offline state, which may create a window of opportunity for malicious intent before its updating. 2) It is difficult to integrate multiple types of security services for legacy design and systems for different architecture, data structure, access control policies, and so forth. The implementation of some new functions, interfaces and protocols may demand a lot of modifications and cause unrealistic performance overhead for the basic cloud platforms.

In this paper, we present ESI-Cloud (Easy Security services Integration Cloud), a novel framework to integrate multiple managed security services by extending VMI techniques [1]. The ESI-Cloud service hosted on a special security VM, which acts as the trusted back-end of VMI service. It can monitor and track execution states of hosted VMs effectively, and renew or modify the security and access control policies on the guest VMs. We also offer a client library as the front-end with rich APIs for the MSSPs; and implement a management console to allow tenants to configure security policies and manage their security services on their guest VMs. With ESI-Cloud service, multiple security services can be easily integrated into the cloud platform. MSSPs can share the current and historical information about executions of VMs, and perform different measurements and detection in parallel.

This paper makes three main contributions: (1) The proposed ESI-cloud service solution can facilitate both cloud service providers and cloud tenants to incorporate multiple security services offered by different MSSPs. (2) The proposed approach enables sharing of VM execution monitoring information among different MSSPs, which can further ease the management of structured metadata and different access rights, as well as the implementation of various security requirements. (3) The proposed approach employs parallel processing for further improving the efficiency of VM security protection in the cloud.

The rest of this paper is organized as follows: Section II describes the architecture and design of the system. Section III presents the implementation detail. Section IV shows the evaluation results. We describe relative work in Section V, and concludes the paper in Section VI.

## II. SYSTEM ARCHITECTURE

The extended VMI framework has a special security VM (Sec-VM), a manage console in Dom0 and a client library to support integrating with all kinds of security services. The system architecture is shown in Figure 1.

The Sec-VM acts as an effective back-end of VMI. It can crawl and monitor various VM execution and resource consumption information in guest VMs. The design of Sec-

VM adopts the solution of running a "Just enough Operating System (JeOS)", which supports only the functions of the installed software [13]. We use the LibVMI to introspect the guest VMs. LibVMI[3] is an open-source implementation of VMI supporting hypervisors such as Xen and KVM and it has been used in many fields, such as integrity measurement[5], monitor[6], IDS[7], and forensic[2]. In Sec-VM, the Crawl module uses the VM's raw information to create the live VM state; the Pool module has some caches to buffer information of guest VMs from VMI. The Policy module maintains a unified security policy and the server-lib respond request from client-lib and return various data required by services.

The client library act as front-end of VMI with variety of APIs to support analysis and detection of guest VMs. The security service could get information of guest VMs they need to protect VMs. The front-end may be deployed in some special VMs or physical servers.

The management console helps administrators in CSP to manage Sec-VM, configure policy and approve access for third independent security services.

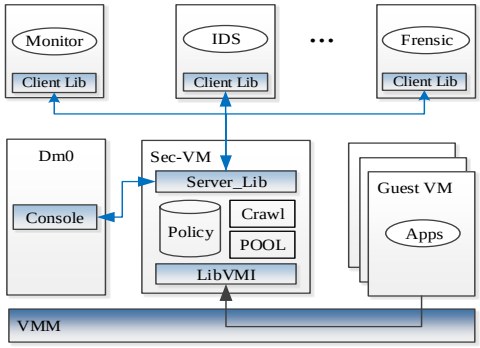



Fig.1. System Architecture of ESI-Cloud Service

## III. IMPLEMENTATION

We implement a prototype of ESI-Cloud based on Xen and LibVMI. The current version is running on Xen 4.2. The Sec-VM is a virtual appliance running Cent OS 6.4 JeOS. Sec-VM is also isolated from other server network in a separate virtual network. Our overall implementation is comprised of four parts: (i) exposing VM runtime view; (ii) developing a client library to support high level analysis and detection; (iii) implementing a console to configure security policy and authorize/revoke access permissions; (iv) optimizing performance by leveraging data caching and parallel processing.

### A. *Exposing VM State*

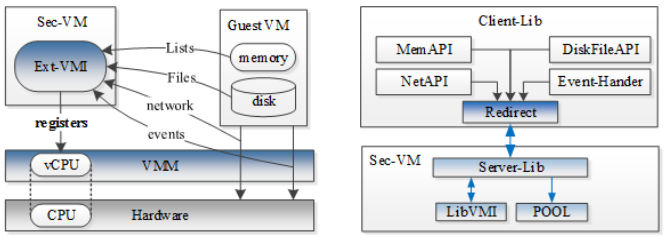
The ESI-Cloud reconstructs an entire view in guest VM, including hardware events, network flow, disk files and memory state. Figure 2(a) illustrates the VMI information collection process.

The memory view is the most important component for ESI-Cloud, because it records many kernel data structures. We install memory access triggers on the pages that contain the kernel data structures of interest and bridge semantic gap just as existing solution of CloudSec [9] and Virtuoso [11]. We focus on how to read disk files, capture network packets and get hardware events.

Ubuntu 12.04 has embedded blktap2 for Xen.Working in conjunction with the kernel blktap driver, all disk I/O requests from VMs are passed to the user space daemon (using a shared memory interface) through a character device. The blkback disks are also mountable in Dom0/Sec-VM without requiring an active VM to be attached. So we can easily read, write files in virtual disks and manage the read/write behavior.

The most common way to achieve network auditing inside a guest VM (DomU) is to assign the physical interfaces from the bare metal to the guest (pci mapping) such that the guest sees the physical hardware. However it will only capture traffic from sources to destinations external to the host. In order to capture all traffic among the guest VMs on the Host, we use Sec-VM to route all traffic between the host and guest VMs and between guest VMs via host.

Event channels are the basic primitive provided by Xen for event notifications. They essentially store one bit of information, the event of interest is signaled by transitioning this bit from 0 to 1. Guest VM receives notifications via an upcall from Xen, indicating when an event arrives (setting the bit). The LibVMI offers useful functions such as 'vmi_register_event', 'vmi_clear_event', 'vmi_get_reg_event', 'vmi_events_listen'.




(a) (b)

Fig.2. Information collected and client library

### B. *Developing Client Library*

In our current implementation, the interface of client library is developed using the Linux RPC. Figure 2(b) gives an overview. The client library makes RPC calls to the Sec-VM for the target VM to be monitored and it returns results of the LibVMI. We implement four types of APIs, including (i) MemAPI to read/write memory buffer, symbols, system call tables, process/module/dll list; (ii) DiskFileAPI to read or wirte files in guest VMs, (iii) NetAPI, which get basic information and capture packets of network; and (iv) Event-Hander, to report or handle events from kernel of VM. These APIs utilize a redirect module to translate the above calls from client-lib to server-lib by secure network communication. Then, the server-lib continues to call LibVMI or to get cache data in Pool. Finally, the results are return to the respective security services.

### C. *Implementing Console*

The console is provided for the administrators of cloud provider to manage Sec-VM. It can start or stop the Sec-VM by a special control interface, configure access rights for different security services. We adapt the Access Control Lists (ACLs) to define the list of access rights that each user has

with respect to a particular system object, such as a process list or individual directory. Figure 5 shows an example ACL node and its data structure. Each object has a security attribute that identifies its access control list. The list has an entry for each Sec-VM user with access privileges. The most common privileges include the ability to read or write a list (register, table, directory/file, buffer). We compact all access rights in bits of one byte, and use a mask to retrieval these access control policies.

### *D. Optimizing Performance*

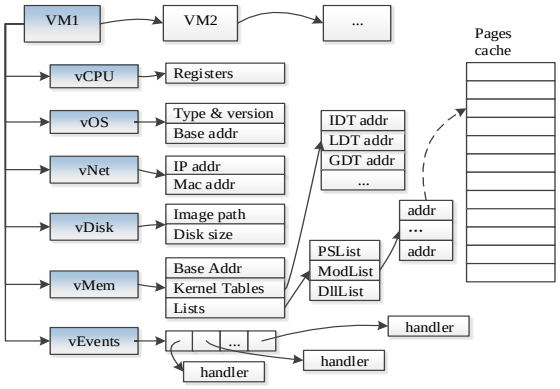



Fig.3. Data recorded for VMs

In ESI-Cloud implementation, we expand the caches for system calls table, process list, modules/drivers list, dynamic link library list, and so forth. Figure 6 shows some data structures that we use to record all the objects in VMs. Each object has sub links to physical address or buffer pages. Each VM assigns a crawl thread to renew data periodically. Some events are put into a FIFO, and evoke relative handler process. At the same time, the system can trap in some critical events and send them to the client-libs. This enables the security services to make detection in time.

## IV. EVALUATION

We evaluate our prototype through extensive experiments by measuring the compliance with functions and the performance overhead of enabling ESI-cloud services.

### *A. Experimental Platform*

We construct an experimental platform on Xen with LibVMI. The host server is a DELL server, with Intel Xeon E3 CPU with VT-x support. Xen 4.2 and Ubuntu 12.04 are installed and the Sec-VM is installed with a kernel cutting CentOS 6.4. We deployed another two guest VMs to run real applications. We use a laptop as attack terminal, which uses Nmap to launch some port scanning attacks. We modify the open source tools of snort [12] and Volatility [4], to make them act as the security services. We evaluate the performance of our system with several micro-bench tools.

### *B. Testing Functionality*

**(1) Guest VM Monitoring.** We monitor process running in the guest VMs in ESI-Cloud and we can get basic information of the process. We selected csrss.exe as example and we compare the external view of mapping the introspected VM's physical memory to Windows OS kernel data structures using ESI-Cloud, with the internal view of the VM using Windbg and the results are shown in Figure 4(a) and Figure 4(b) respectively. By this way could we validate that ESICloud service can bridge the semantic gap successfully.

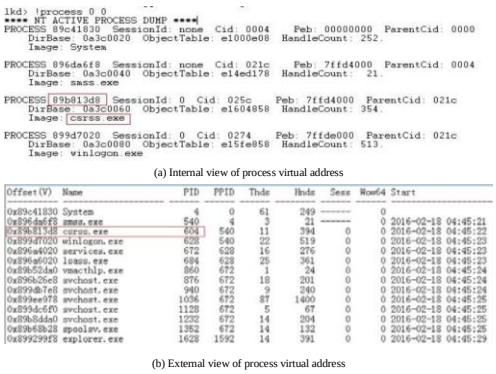



Fig.4. Basic information of csrss.exe

**(2) Intrusion detection.** We perform a test for intrusion detection by using tool of Nmap to see if the modified snort would be able to detect port scanning attack. After the Nmap started to scan for a while, we did see that the modified snort was able to detect and log it as an alert (shown in Figure 5).

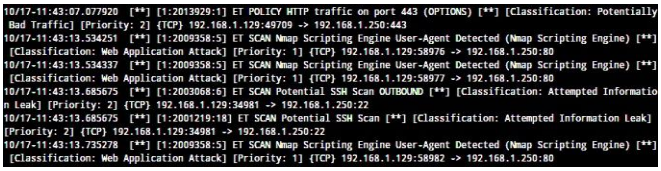



Fig.5. Testing of port scanning detection with nmap

**(3) Malware analysis.** We select KBeast [14], to test function of supporting forensic in our system. KBeast gains its control over a computer by hooking the system call table and by hooking the operations structures used to implement the netstat interface to userland. Volatility detects all of these hooks by enumerating and verifying each entry in the system call table. This is implemented in the linux_check_syscall plugin, which, for every member of the system call table, either prints out the symbol name or, if it is hooked, prints out the hook address. The result show as table 1.

TABLE I. HOOKS ADDRESS IN KBEAST

| Kernel hook | Original addr | Detected addr |
|---|---|---|
| [S] read | 0x0206c90 | Oxf80d49f0 |
| [S] write | 0x0206c20 | 0xf80d51f0 |
| [S] getdents/getdents64 | 0x02153a0 | 0xf80d5940 |
| [S] unlink | 0xc02121c0 | 0xf80d5340 |
| [S] rmdir | 0xc0212310 | 0xf80d5420 |
| [S] unlinkat | 0xc02122d0 | 0df80d5500 |
| [S] rename | 0xc0212010 | 0xf80d55f0 |
| [S] open | 0xc0d044a0 | 0xf80d5780 |
| [S] kill | 0xc015d6a0 | 0xf80d46a0 |
| [S] sys_delete_module | 0xc017e870 | 0xf80d5870 |
| [FP] tcp4_seq_show | 0xc0501980 | 0xf80d47d0 |

### *C. performance Analysis*

To evaluate the overall performance of the ESI-Cloud service, we run two existing LibVMI examples on ESI-Cloud

and on the original (unmodified) LibVMI. We perform experiments in three different scenarios: (i) using the original-LibVMI in Sec-VM, (ii) using ESI-Cloud in Sec-VM, and (iii) using ESI-Cloud in a cross physical machine (Other-PM, e.g. Monitor). We compute the average first cost time and second cost time. For the system can use data in cache since second time, we can see that ESI-Cloud has some overhead in the first time, but is faster in the second time. This is because LibVMI, Sec-VM and Client-Lib all have caches for process list and module list. While in the first time, Sec-VM will do more work than LibVMI, such as write lists into pool.

To check how much our solution can improve performance by parallel execution, we repeat our experimental in the decentralized platform and centralized 100 times in loop and we can see that ESI-Cloud service gets higher performance of 12% by parallel processing.

TABLE II. BENCHMARK TIME

| Time(ms) / Test | LibVMI | | Sec-VM | | Other-PM | |
|---|---|---|---|---|---|---|
| | 1st | 2nd | 1st | 2nd | 1st | 2nd |
| Process-list | 21.23 | 17.37 | 23.58 | 15.54 | 25.12 | 17.43 |
| Module-list | 20.45 | 16.29 | 22.48 | 14.87 | 24.62 | 16.87 |

TABLE III. BENCHMARK TIME

| Test | Centralized | Decentralized |
|---|---|---|
| Process-list | 21.221ms | 19.732ms |
| Module-list | 19.325ms | 18.975ms |

## V. RELATIVE WORK

A number of research projects have developed solutions to employ VMI in a wide range of security protection areas, such as intrusion detection [8], malware detection [7], digital forensics [2] and so forth. Garfinkel et al. [8] proposed the idea of virtual machine introspection, an approach to intrusion detection which co-locates an IDS on the same machine as the host of which it is monitoring, and it leverages a virtual machine monitor to isolate the IDS from the monitored host. Ahmed, et al. CloudSec [9] is a new virtualization-aware monitoring appliance that provides active, transparent and real-time security monitoring for hosted VMs in the IaaS model. The research of CloudSec is similar with our work in terms of extending VMI for security analysis, but ESI-Cloud development is focused on integration of multiple security services instead of only on security analysis. CloudVMI [10] allows virtual machine introspection to be offered as-a-service by cloud providers. It enables cloud centric introspection by allowing VMI actions to be performed across different physical machines.

## VI. CONCLUSION AND FUTURE WORK

We have presented an extended VMI solution, called the ESI-Cloud (Easy Security services Integration Cloud) service, to facilitate the integration of multiple managed security services into a cloud platform effectively and flexibly. A dedicated security VM monitor and analyze the VM execution of guest VMs and renew or modify security policies effectively. A client library with rich APIs to the managed security service providers (MSSPs) and a management console to configure security policies, approve access for security services and so forth. In ESI-Cloud, multiple security services can be easily integrated into the basic cloud platform by sharing the real-time monitoring data and historical information in VMs, and performing various measurements and detections in parallel. We are working on making the open source of ESI-Cloud framework and library available. In addition, we plan on further improving performance of caches and event hander, to expand the capability of adapting more security services.

## ACKNOWLEDGMENT

This work is carried out when Dr. Jiangchun Ren is a visiting scholar in Georgia Institute of Technology. Authors from NUDT are partially supported by the 863 project of China under Grant Nos. 61433019. Authors from Georgia Tech are partially supported by the National Science Foundation under Grants IIS-0905493, CNS-1115375, NSF-1547102, SaTC 1564097, and Intel ISTC on Cloud Computing.

## REFERENCES

[1] Yulong Wang, Xianqiang Yang.Survey of the Virtual Machine Introspection in Cloud Computing. Advances in information Sciences and Service Sciences(AISS).Volume5(10), 2013

[2] A. L. Shaw, B. Bordbar, J. T. Saxon, K. Harrison and C. I. Dalton , Forensic virtual machines: dynamic defence in the cloud via introspection , IEEE International Conference on Cloud Engineering

[3] Haiquan Xiong, Zhiyong Liu, Weizhi Xu, Shuai Jiao. Libvmi: A Library for Bridging the Semantic Gap between Guest OS and VMM. Proc. of the 2012 IEEE 12th International Conference on Computer and Information Technology.

[4] Volatility Usage. https://github.com/volatilityfoundation/volatility/wiki/Volatility-Usage. [Accessed on 10 Dec. 2015].

[5] Quynh NA, Takefuji Y. A novel approach for a file-system integrity monitor tool of Xen virtual machine. Proceedings of the 2nd ACM Sumposium on Information, Computer and Communications Security (ASIACCS), 2007.

[6] Li N, Li B, Li J, et al. vMON: An Efficient Out-of-VM Process Monitor for Virtual Machines. The IEEE 10th International Conference on High Performance Computing and Communications (HPCC), 2013.

[7] Harrison, C., et al. Constructing a Cloud-based IDS by Merging VMI with FMA. The 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom).

[8] T. Garfinkel and M. Rosenblum, A Virtual Machine Introspection-Based Architecture for Intrusion Detection, Proc. Network and Distributed Systems Security Symp., The Internet Society, 2003, pp. 191-206.

[9] A. S. Ibrahim, J. Hamlyn-Harris, J. Grundy, and M. Almorsy, Cloudsec: A security monitoring appliance for virtual machines in the iaas cloud model, Proceedings of 5th International Conference on Network and System Security (NSS 2011).

[10] WOOK BAEK, H., SRIVASTAVA, A., AND DER MERWE, J. K. V. Cloudvmi: Virtual machine introspection as a cloud service. IEEE International Conference on Cloud Engineering (IC2E) (2014).

[11] T. Leek , M. Zhivich , J. Giffin and W. Lee , Virtuoso: Narrowing the Semantic Gap in Virtual Machine Introspection, Proc. 32nd IEEE Symp. Security and Privacy , 2011..

[12] The Open source Network intrusion Detection System. [online] Available:http://www.snort.org/

[13] VMware, Virtual Appliances: A New Paradigm for Software Delivery, 2009, Available: http://www.vmware.com/files/pdf/vam/ VMware_ Virtual_Appliance_Solutions_White_Paper_08Q3.pdf.

[14] Carbone R. Memory analysis of the KBeast Linux rootkit[J]. 2015.