SAND2016-11787C

Hypervisor Assisted Forensics and Incident Response in the Cloud

Vincent E. Urias, William M.S. Stout, and Caleb Loverro
Sandia National Laboratories
Albuquerque NM USA
{veuria,wmstout,cloverr}@sandia.gov

John W. Young
Marymount Unversity
Arlington VA USA
jyoung@marymount.edu

***Abstract*—Cloud computing has been integrated into many areas of production and received a great amount of attention from both government and private industry. Despite its popularity, cloud technologies are still not well understood and numerous areas are open for research and development. One critical area that has not received significant attention in the security domain of cloud computing is digital forensics and incident response. In this paper we address the challenges in those domains by introducing a novel approach using virtual machine introspection to provide intelligence, introspection, and modification of VM state in cloud systems. The focus of this research paper provides: (1) the context for security in cloud and the challenges introduced by conducting digital forensics in the cloud; (2) new opportunities that exist to meet these challenges in the Cloud through virtual machine introspection, to include correlation with network data and active state modification, and (3) use-cases and experiment results for our techniques under the guise of Cloud forensics.**

***Index Terms*—cloud computing, virtual machine, virtual machine introspection, digital forensics, incident response**

I. Introduction

The Cloud has been leveraged for many applications by many different industries. Despite its popularity, cloud technologies are still not well understood and are open for research and development [15][17]. The security implications of cloud computing is a critical topic requiring additional research. From a forensic perspective, numerous questions arise on how to analyze the Cloud using traditional digital forensics techniques [16][20]. For instance, during a traditional digital forensic examination, all files on the storage media are examined along with the entire file system structure. However, this is not a practical model for cloud infrastructure, as the elasticity and ephemerality of pooled storage make pinpointing data blocks cumbersome. This difficulty is exacerbated in networked systems by the scale with which computing resources are spread over diverse administrative and geopolitical domains. Cloud is able to combine numerous heterogeneous resources (hardware platforms, storage back ends, file systems) that may be geographically distributed. The idiosyncrasies in cloud have caused a paradigm shift in digital forensics; however, tools and techniques still do not exist to help forensic practitioners cope with these issues. And while many research areas enumerate these challenges, open literature has not made significant headway to address the issues them or provide solutions.

Our approach proposes a new view to the issue through the use of virtual machine introspection. Section II provides a foundational background on cloud computing, digital forensics, and incident response. Section III covers the challenges that arise for forensics for cloud environments. Section IV describes our methodology and approach to addressing those challenges. Section V describes our hypervisor-based introspection tool for cloud forensics. Our experimental analysis for our introspection tool is highlighted in Section VI, while Section VII concludes this paper and suggests future work.

II. Background

A. Cloud Computing

The National Institute for Standards and Technology (NIST) defines cloud computing as "a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [6]. A few businesses have emerged as leaders as cloud computing has become increasingly mature and available. Amazon, Google and Microsoft have demonstrated support through the promotion, encouragement, adoption, and leadership of cloud computing, building a foundation for recent paradigm shifts. The paradigm will continue to evolve as the Cloud becomes more pervasive.

The promise of cloud computing has spurred entrepreneurial development of cloud services. The services provided by these businesses are generally divided into three categories:

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

With both SaaS and PaaS, cloud providers often have tight control of the execution environments, as the applications that users access are limited in the number of configurable options. The focus of the studies in this paper is on the infrastructure component, IaaS. There are several IaaS platforms (e.g., OpenStack, OpenShift, EC2) as well as providers in both private and public settings. IaaS provides users with the most freedom of configuration for their virtual environments, comparable to what they would have in their own enterprises. However, just

as in traditional networks, IaaS is not immune to malicious actors that take advantage of poor security policies, weak credentialing, and multitenancy. While the rapid elasticity of IaaS provides the most bang for the buck', the ephemeral nature of data in the cloud does not lend itself well to forensic investigation following compromises, breaches and attacks. Still, companies and government sectors have made it part of their long-term strategic plans to leverage cloud technologies for their infrastructures [3]. As cloud computing, and specifically IaaS, become more ubiquitous, it becomes more imperative to address the challenges described in the following sections.

B. Digital Forensics and Incident Response

There have been many inquiries into the ability of forensic practitioners to conduct the science of digital forensics in cloud-based infrastructure and the ability for the current tools and techniques of digital forensics to operate in the cloud. The Cloud infrastructure - with its distributed processing, storage, and resources - can be extremely complex because storage capacities can grow geometrically. Before understanding the applicability of current digital forensics practices to the Cloud, we must construct a common understanding of digital forensics.

Informally, digital forensics is defined as "the collection of techniques and tools used to find evidence in a computer." It is often considered a science due to its systematic, technological approach toward inspecting a computer system and its contents. Its aim is to locate and preserve electronic evidence for use in criminal investigations. Digital forensic investigations require a level of expertise and rigorous methodology that exceed standard data collection and preservation routinely performed by system administration personnel.

Digital forensics, as a scientific discipline, is concerned with the collection, analysis and interpretation of digital data connected to a computer security incident, as well crimes that involves a digital device that may store electronic information. Practitioners have attempted to provide some formalization to the field by defining a five-phase process:

1) Identification of an incident from its source(s) and determine its type.
2) Acquisition of evidence from various sources.
3) Preservation of the state of evidential data.
4) Analysis of evidential data, reconstructing fragments and drawing conclusions.
5) Reporting of results and conclusions about the evidence.

This standard unifies many of the previous forensic protocols and provides an abstraction to the process that is not focused on a particular tool or technology, nor is bound to a specific class of cyber-crimes. During a forensic examination, all files (e.g., storage, log files), memory, and external media are examined along with the entire file system structure to locate forensic artifacts. However, each of the phases provides unique challenges and opportunities for investigators as cases and artifacts are situated in the Cloud. Numerous papers have discussed the challenges to current hard-disk-based forensics approaches [10][9]. To date, the predominantly focus has been on challenges and solutions in in the network forensics subdiscipline (which focuses on forensics of network traffic, rather than hard disk forensics). There is a gap in the ability to conduct the preservation and analysis of hard-disk forensics in the cloud. Efforts have been made [8], but do not address the area and do not provide a significant improvement or methodology in this area.

We claim that there is some overlap in the goals and approaches in conducting digital forensics and incident response. National Institute and Standard Technology (NIST) [2] defines incident handling as a lifecycle that includes incident response, includes preparation, detection/analysis, containment, eradication and recovery. The authors of [1] describe that incident management includes responding to an incident (cyber), vulnerability and artifact handling, and other related services.

There are many parallels between the steps in incident handling and the forensics phases of the incident response life cycle. With the focus on creating forensic artifacts that are actionable, there are limited tools, methods and approaches that enable the collection and preservation of forensic evidence in the Cloud. Both approaches and disciplines face similar challenges in the need to interact with the system in a transparent, non-intrusive fashion. In order to contain, collect and analyze evidence, both areas are demanding that advancements and tools be written to aid in their approaches.

III. Cloud Forensics Challenges

Some of the most attractive benefits for cloud computing involve a subscriber's ability to receive services from a broker or provider, and expand their requirements at scale; the burden of scaling is placed on the broker or provider and becomes transparent to the user. Coupled with this is the economic perk that subscribers need only pay for what they use (i.e., a pay-as-you-go), forgoing the operations and maintenance costs that would normally accompany an on-premises data center.

However, incorporating cloud infrastructure into a company's network may alter its threat surface and appear contrary to security and privacy controls implemented for boundary protection. Cloud computing presents the risk of shared computing resources among multiple tenants on the same physical hardware; there is a need to have strict software isolation in order to prevent one tenant's software from compromising another tenant. For IaaS, a lack of proper virtual machine (VM) separation severely elevates this risk. For data protection, providers and administrators must ensure only authorized users have access to their data, and that their data is protected at rest, then sufficiently isolated and permanently erased during data sanitization. When security incidents occur in violation of risk-reduction controls, the challenges involved in the cloud incident response and forensics begin to manifest [19].

The notion and risk of acquisition changes within virtualized environments. Without physical queues and devices, it may no longer be feasible to physically protect against contamination of the machine through a "write-blocker." In traditional networks, the analyst physically removes the drive

to create a bit-to-bit image of the device. In the cloud, analysts may be bound to the network that the VM is on. As a consequence, investigations are more dependent on the surrounding infrastructure than physical machines.

The elasticity of cloud relies on the dynamic allocation of physical resources [19]; difficulty arises if the geographic location of an examiner is drastically removed from one or many servers involved in an incident. What if the analyst is attempting acquisition from the device when the physical network interface card goes down? If the analysts must now remove the drives, they must acquire the entire device in order to acquire the specific image that they were looking for, increasing processing time. Additionally, there are legal implications with regard to other data which may be tangentially acquired (intentionally or incidentally).

The notion of forensically sound images is also a challenge, paricularly source images. Will the service provider have to store the image/backup on their system until the case is resolved to ensure image integrity/attribution? Or, will a hash of the VM compared to the other VM be enough to satisfy the requirement of producing the source evidence in the court? Such circumstances will leave the analyst dependent on the backup strategies of the service provider, which may vary from cloud to cloud, and may not always be viable.

The identification of possible roadblocks in conducting cloud forensics is a daunting task. The authors of [7] effectively categorize those challenges into nine major groups:

1) Architecture: diversity, complexity, provenance, multitenancy, data segregation.
2) Data collection: data integrity, data recovery, data location, imaging.
3) Analysis: correlation, reconstruction, time sync, logs, metadata, timelines.
4) Anti-forensics: obfuscation, data hiding, malware.
5) Incident first responders: trustworthiness of cloud providers, response time, reconstruction.
6) Role management: data owners, identity management, users, access control.
7) Legal: jurisdiction, laws, SLA, contracts, subpoenas, international cooperation, privacy, ethics.
8) Standards: operating procedures, interoperability, testing, validation.
9) Training: forensic investigators, cloud providers, qualification, certification.

The first four groups are the most technology dependent; what is troublesome regarding the groups is the dependence on media and/or disk-based forensic analysis. The identification, collection and preservation of physical media to capture incident evidence and artifacts in a cloud environment are difficult and sometimes impossible. These shortcomings are promulgated by varying cloud providers, improper identification of cloud user accounts, gaining assistance from cloud staff, system understanding, volume of data and noise, data location, privacy issues (multitenant data), and encryption.

To alleviate some of the issues of media, disk, data temporality, location and ownership in the Cloud infrastructure, providers leverage logging to detail the events that occur in their domains. These logs are normally comprised of: (1) audit logs that may correlate services to operating systems, (2) security logs that may attempt to connect users to broad actions, and (3) application logs that highlight cloud application activity. However, these logs often suffer from the semantic-gap problem due to the lack visibility into the VM, where the events take place. The common denominator in any IaaS-based environment is the hypervisor. Is the our belief that many of the deficiencies with cloud forensics may be addressed by tapping the hypervisor for VM introspection (VMI); that is, uncovering forensic artifacts at the virtual machine manager (VMM) layer. While the "use of logs in hypervisors is not well understood and presents a significant challenge to cloud forensics" [7], it is our conjecture that such logging may be the successful path forward. This is the core of our research, and is outlined in the following sections.

IV. Methodology and Approach

As systems and devices become virtualized and deployed in the cloud, the hypervisor becomes an increasingly appropriate place to collect performance data, system state, system landscape, function calls, transaction traces, and other characteristics. We propose a method by which an introspection application may be coupled with a hypervisor, in order to "reach into" the VM with minimal intrusiveness to collect data critical to the reconstruction of events, files, and operations. Such a capability is required to take advantage of the hypervisor as an instrumentation platform and to integrate that data with more traditional collection mechanisms.

The concept of a VM serviced by a lightweight hypervisor is a relatively new paradigm for forensic practitioners. Traditional forensic techniques, based on assumptions that the file-system was directly interacting with the hardware through an abstraction, afforded the forensic practitioner the assumption that there was nothing controlling the application below the file-system. This is not the case when using virtualized technologies. Hypervisors may have the ability to covertly monitor, introspect and interact with the guest in a transparent fashion. As mentioned in previous sections, the problems of storage and collection of actionable data are exhausting.

The current challenge is most hypervisors do not expose a useful application programming interface (API) at a sufficient level to do transparent, fine-grained and customizable introspection. Scalable VM instrumentation and introspection at an in-depth level requires fast handling of events, as well as direct access to VM state. Furthermore, deep introspection benefits greatly from the ability to gather data from the hardware during the VM's exit to the hypervisor. All of this requires identical access to the system as the hypervisor itself; improper use of this ability could easily cause system instability. It is for this reason we believe that the hypervisor developers have been hesitant to grant this much control through their APIs.

However, our approach leverages other means to collect and monitor the guests in a targeted fashion.

A. Virtual Machine Introspection

Virtual machine introspection (VMI) is a technique used to monitor the runtime state of a system-level virtual machine. The runtime state can be may include processor registers, memory, disk, network, and any other hardware-level events [13]. A review of research literature and current VMI technologies exposed a number of limitations and trade-offs in VMI approaches [11][5][12][8], including: the use of in-guest agents; kernel to user space transitions (dramatically slowing down processing); VMI tool pre-configuration requirements; hypervisor version lock-in or source code patching; reliance on operating system (OS) symbols; limited processor features due to hypervisor (even if the hardware could do more).

To address these constraints, a VMI tool was envisioned to provide the cloud forensic capabilities while having as few of these limitations as possible. The Kernel-based Virtual Machine Introspection (KVMi) tool was developed for the Kernel-based Virtual Machine (KVM) hypervisor on Intel's x86 architecture [4]. To meet performance, scoping and use-case demands, the follow tenets were applied to KVMi:

- Shall not require in-guest agents.
- Shall work with any recent version of KVM.
- Shall work with any version of Windows starting with Windows 7 64-bit, including newer versions such as Windows 10 64-bit while still having the ability to find and track basic Windows artifacts
- Shall not require OS symbol files.
- Shall be able to fully handle VM-exits, bypassing execution of KVM if necessary, to facilitate new features KVM may not support.
- Shall be compilable/loadable on a running system with standard build tools.

KVMi is implemented as a single loadable kernel module for Linux. It begins by locating the kvm and kvm_intel kernel modules in kernel-space memory. Upon finding them, it hooks code in KVM's exit handler, redirecting execution into its own exit handler. When KVMi encounters a new VM, it determines its operating system and adds the system of interest to a set for further introspection. VMs then run until an operation within the guest causes them to VM-exit, which then passes control to KVMi's VM-exit handler routine, providing the foundation to understand the dynamic behavior of actors within the virtual machine, introspect without introducing artifacts into the running system, and allow full control over guest system.

Each VM of interest is dynamically analyzed to determine offsets of key structures in memory. This is done in multiple ways, including walking exports of portable executable (PE) files, disassembling code, and simple recognition of data in relation to other objects. It also utilizes VM exits for things like control-register access, model specific register (MSR) access, CPUID, and timer related exits. KVMi also keeps track of each virtual CPU separately, and links them to their respective VM. For breakpoints, it uses permissions in Intel's extended-page-tables (EPT) to trap on read, write, or execution on arbitrary sized chunks of memory. It leverages the monitor-trap-flag (MTF) bit for single stepping. KVMi produces log-based output through ring buffered character devices on the host's device file system - and may receive select input through this method as well.

To support forensic analysis of incidents on and from guests, KVMi provides:

- Reconstruction of dynamic linked libraries and drivers (lists as well as full reconstructions out of memory)
- File reconstruction
- Guest process lists (includes parsing the PE files of all modules loaded in each process to find functions of interest)
- Guest system call logging
- Guest operating system function calls and parameters
- Memory access

Since KVMi has the ability to make modifications to the guest system, such as hiding or changing guest files in memory, or redirecting execution, an administrator may run arbitrary code in the kernel or user space processes directly from the hypervisor without any in-guest agent or user logged in. Such a capability may be used to support live, forensic data collection.

B. The Hypervisor Kernel and Inline Introspection

Most hypervisor platforms allow some interaction through an API. They can range from simple things like querying the power status of VMs, to more complex things like viewing or modifying register state inside a guest. In all cases, a considerable amount of overhead is incurred. The APIs for hypervisors like Xen or VMware require ring switches and transfers between the hypervisor and a special VM (Dom0 or secure virtual machine (SVM) respectively). This effectively separates the actual hypervisor kernel from bugs in the VMI code; however, it also causes any introspection data to travel far from the hypervisor kernel before it reaches the VMI code. Additionally, it enables hypervisor authors to decide what data is relevant to the VMI code. This separation is advantageous for the hypervisor, but at best neutral for the VMI code, and only if it is able to get all the data that it needs. The VMI code can request additional information, but this requires even more context switching, and is still limited to what the hypervisor will allow it to request. The VMI code will not be running in VMX root mode and thus does not have the ability to use virtual machine extensions directly. In some cases, APIs to do certain tasks don't exist, and thus the only way to do some types of introspection is to patch or hook the KVM kernel code and obtain VMX root privileges. For example, it is not possible to ask KVM to enable the MTF (monitor trap flag) functionality of Intel hardware virtualization to single-step a guest.

Actions such as extracting large buffers of data from frequently used system calls or other functions require considerable overhead to process, and thus must be handled in the most efficient manner. When a guest VM-exits to the hypervisor, it is in a suspended state on the CPU core which has exited. When the hypervisor takes too long to do its processing during

the exit, a noticeable lag can be seen by users in the guest VM. Since KVMi hooks KVM to gain execution in VMX root mode, it is also running in this window during the VM-exit. KVMi has two interfaces for users to interact with, a set of character devices, and a sysfs tree. In both cases, the data given to the user is stored inside a kernel buffer. When the data is captured by KVMi, it performs a copy from the VM's buffer into the kernel buffer and immediately lets the guest resume. No other communication takes place during the exit. With both character device and sysfs interfaces, the user requests the data at his or her leisure, and it is copied to them outside of VMX root mode. No ring switching or VM transitions occur other than the one required VM-exit and VM-enter (which would have happened even without the hooking). With this method, KVMi is able to move data from the VM to the host with minimal overhead.

V. VM Introspection and Monitoring in the Cloud

A. Needs and Requirements for the Cloud in using KVMi

With the advent of cloud computing and virtualization, special care needs to be taken by the data center, cloud service provider, and the cloud architect to ensure the tenant's (intellectual) property is secure. Cloud computing changes the relationship between the computer hardware and the operating system that manages and controls it. Focusing on the added virtual layer is not enough. With the KVMi we can look at the hypervisor and ways to more tightly secure it.

1) Personnel Security: Today with companies, governments and organizations choose to host services and store information on the cloud (both public and private), the physical access to their digital property will be inevitably lost. Because of this risk, the possibility of data being exposed to attack is higher. The biggest threat to sensitive data will possibly come from individuals or groups inside the data center. Therefore it should be put on the cloud services provider to secure the system, software and data through background checks of data center personnel. Access to the KVMi application should also be restricted and controlled based on detailed roles of the individual.

2) System Security: The cloud service providers should also perform suspicious activities monitoring to eliminate unauthorized or nefarious access to the virtual systems. Although, this type of monitoring is an important security feature of KVMi, it can also be a means of full access to the virtual systems of the cloud. Another key to mitigating mishandling of the KVMi or the cloud is to insure the logical cloud stores are segregated and the data is isolated thoroughly.

B. Impetus for KVMi in the Cloud

The notion of the virtual machine sitting on top of a lightweight hypervisor is a relatively new paradigm for forensic practitioners. However, it is a near-ubiquitous certainty for most IaaS infrastructures. Traditional forensic techniques, based on assumptions that the file-system was directly interacting with the hardware through an abstraction, afforded the forensic practitioner the assumption that there was nothing controlling the application below the file-system. This is not the case when using virtualized technologies. Hypervisors have the ability to transparently monitor, introspect and interact with the guest in a non-intrusive fashion. There are four key areas that monitoring and collecting data from the hypervisor would assist in alleviating:

First is the need to process entire large storage pools used by the IaaS infrastructure. Storage solutions in the cloud are varied. To support a variety of formats, e.g., Fiber Chanel, Ethernet iSCSI, and a variety of file system types, raw data can be petabytes in size. The VMs often exist in some "shardded" (striped) fashion on the file system. Current forensic tools are unable to collect data from a large data volume in a timely fashion, nor can the companies who host these services afford to take the storage system offline in order to gather forensically sound evidence from the underlying file systems. If one were to move processing to the hypervisor to gather all the file system artifacts - then all the IO is decoded, saved and archived before being written to the distributed file system.

Second, the ephemerality of the guests and of cloud computing is a challenging problem, raising many issues regarding the lifetime of a particular device. The lifetime is no longer years or months, but rather it is weeks at best. Storage issues are one of the greatest challenges of cloud computing; as demand for resources increases, the cloud provider's ability to store all of a particular user's information for weeks, or even months, becomes economically unfeasible. As space is reclaimed, forensic evidence is lost. As an alternative, an emerging enterprise trend is to have users use transient clients and to store user profiles at a separate location. But, as virtual machines are cleaned and reimaged, all potential for evidence residing on the original virtual machine is lost. Some IaaS cloud platforms such as OpenStack and Amazon EC2 have mapping knowledge of where guests are deployed. Given that information, it is possible to do targeted collection by the hypervisor, making it possible to collect artifacts from a guest while it is still running. It is possible to get information regarding file I/O, memory, processes, network connections as well as traceability of the actions on the system.

The third area is the elasticity of the collection methods and processing of the data. By collecting data from individual hosts, the approach scales with the cloud. No longer does forensic analysis or artifact collection focus on a single host; every host that may have information can assist in the collection and processing of the forensic artifacts.

Fourth is forensic collection and time correlation of the guest artifacts make it hard to prove the provenance of the artifacts. By collecting the artifacts from the hypervisor, it is possible to independently verify all the logs, access and interactions from the guest and create a forensic timeline of the event that is grounded with a trusted time source.

C. Fusion with Networking Views of Cloud Space

In the domain of computer networking, virtual switching was devised a means to support networking of virtual machines on a single compute node, or host. A virtual switch is

essentially a kernel process executed on the host, often in collusion with a hypervisor, to provide virtual interfaces (Ethernet segments) to virtual machines and switching/forwarding logic between interfaces. The applications derived from virtual switching can be as simple as bridges, performing layer-2 forwarding operations, or as complex as multilayer forwarding and routing functions and protocols, as well as supporting newer approaches to networking, such as software-defined networking and network function virtualization.

The authors of [14, ANON] describe a method for conducting networking monitoring at the application layer, to provide greater visibility and correlation of network traffic to VMI. Their solution is implemented in C-code for the open-source virtual switch, Open vSwitch. Traffic classification leverages several libraries, provided through a continuously updated catalog of thousands (˜2500) of plugins and signatures covering human-initiated and IoT protocols/applications. Metadata is extracted for HTTP (request, servers, URIs, MIME types), DNS (hosts, queries, servers), SMTP (mailfrom, header), Kerberos (login, server), LDAP (hostname), with goals to include OSPF and BGP data.

Output from classification and metadata extraction are output to agnostically-formatted log files that may then be ingested by security information and event management (SIEM) systems and/or logging/analysis engines. These data are correlated against log data output from the KVMi character devices may be ingested by an analytic engine to provide a rich set of analytics. When coupled with network-based data introspection, a full view of historical and current state of VMs and their interactions with other entities (either in the cloud ecosystem or external) may be captured and inspected. A high level diagram depicting a notional infrastructure for KVMi and network forensic tools in a cloud and analytic environment is shown in Fig. 1. The architecture is fluid and agnostic to be supported on one or many compute host servers of varying architectures.

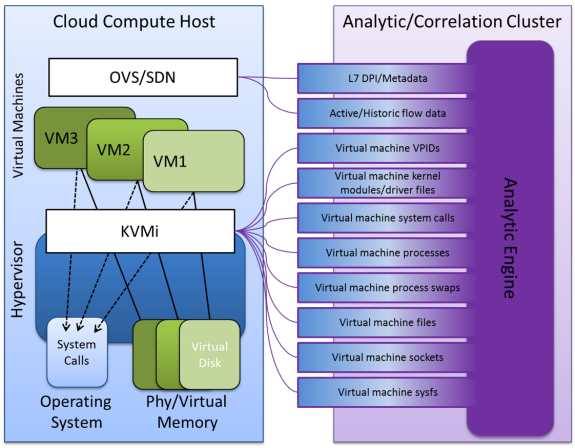


Fig. 1: KVMi/Network Forensics Architecture.

VI. Experiments and Analysis

To evaluate the efficacy and applicability of the KVMi tool for cloud forensics, we identified three areas or use-cases for experimentation and analysis. The first involves the reconstruction of files placed or executed on targeting machines, to be used as forensic evidence. The second involves monitoring and gathering intelligence for attacks in progress, to include network traffic. The final experiment involves the ability to collect general VMI and network data for historical purposes, in a multitenant environment. The results of the use-case testing elucidate the strengths and weaknesses in each situations, and possible means for improvement. The testing environment was comprised of the following hardware and software elements:

- Supermicro servers
- 264GB RAM
- 32 CPUs (Intel(R) Xeon(R) CPU E5-2670 @ 2.60GHz)
- Ubuntu 14.04.02 LTS (kernel 3.13.0-57)
- KVM/QEMU 2.3.50

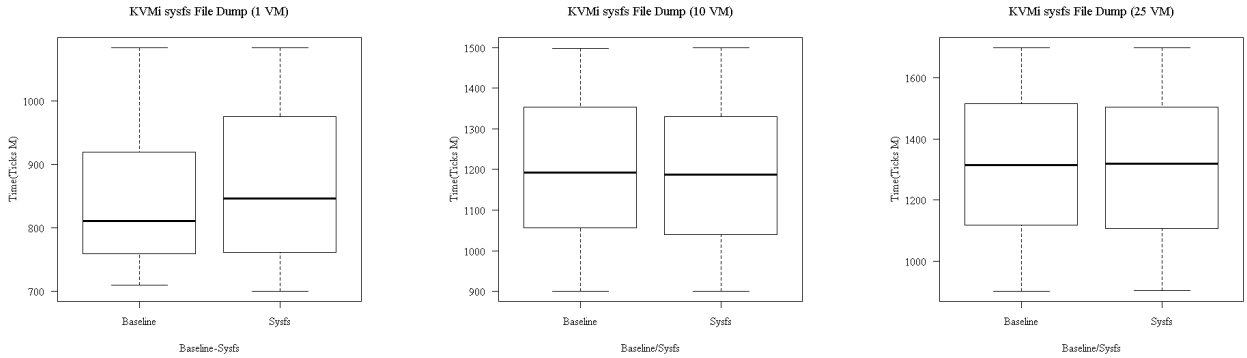
A. Use Case 1: VM as a Platform for Attackers

The purpose of this experiment is to verify the extraction of various forensic artifacts from the system without adversely affecting the guest and without guest detection of the introspection. The underlying concept addresses intellectual property theft, child pornography, etc.

1) Experiment Method: A light weight agent was created that could download files to the guest through a web interface, and then saved them to disk. This would mimic a variety of content being shared (such as child pornographic images, sensitive proprietary information, etc.); typical of what would be transferred and accessed through the Cloud. To conduct the experiment, the same file was used for download in experiments consisting of 1, 10 and 25 virtual machines on a single host. Time ticks were counted during each of the downloads to identify time differences between baseline (that is, without KVMi extracting the file) and with the KVMi sysfs functionality enabled.

2) Results: As mentioned in the description of KVMi, the KVMi kernel module is attached to the KVM hypervisor; hence, its existence is not visible from inside the guest (unlike agent-based solutions). Thus, the only indicator of visibility from inside the guest might be through timing analysis. For the experiment of concurrently downloading a pdf file with 1, 10 and 25 VMs on a host, the time in millions of CPU ticks for the download are show in the box plots below. Each download was run 30 times on each VM instance.

As can be seen in the plots in Fig. 2, there is negligible difference in the time ticks between the downloads (growing more consistent with the greater number of samples). Also, it should be noted that the sysfs process was also able to extract the file before it reaches the disk encryption process for NTFS; md5sums were also taken to show the pdf extracted was the same downloaded.



(a) Case I. 1 VM (b) Case II. 10 VMs (c) Case III. 25 VMs

Fig. 2: sysfs Evaluation Results.

B. Use Case 2: VM used as an Exploited Endpoint

The purpose of this experiment is to help identify anomalous guest processes or to identify "stealthy" malware (process hiding techniques). These are both techniques that could be used by a malicious user's attempt to hide their actions through covert means. The process information would then be correlated to network traffic supporting C2-like operations. The underlying concept here represents a typical drive-by-download attack against a VM to include exploitation and pivoting.

1) Experiment Method: A user on a VM would visit a "malicious" website, that would then exploit a browser vulnerability, providing the attacker privileged control of the virtual machine. At this time, the attack would then pivot to other machines in the network, using metasploit to gain passwords.

2) Results: This particular experiment makes use of KVMi to introspect on guest VMs, and network forensic tools (as described above) to correlate guest data to network data. The results of the experiment largely focus on log data to navigate the attack in realtime and identify the actions done on the target VM. By logging the cloud compute host, virtual machine name/ID, and IP addresses, the VM in multitenancy can be quickly identified. KVMi includes data pulled form Windows APIs, with parameters. The collection of guess and network data address the semantic-gap problem of pulling context from the guest to the host.

To start the experiment, an administrator logs into a VM (.82) and adds a share using domain admin credentials. He then visits a phishing website hosted on the attacker's machine ("attack.com", .2). This is shown in a network forensic log of Fig. 3 as a DNS request to the DNS/domain controller server at .66.

```
1460340185,dpi_log,path=base.ip.udp.dns,srcip=[redacted].66,dstip=[redacted].82,udp_port_src=53,udp_port_dst=54799,bytes=396,packets=5,metadata="query:attacker.com,name:attacker.com,addr:[redacted].2"
host = blue3 | source = /tmp/flow_logs/dpilog.log | sourcetype = dpi_log
```

Fig. 3: URL Observation.

The attacker compromises the VM (.82) using a Silverlight Exploit through a XAP file and runs a bind meterpreter on port 2222 (Fig. 4).

```
1460340189,dpi_log,path=base.ip.tcp.http.silverlight,srcip=[redacted].2,dstip=[redacted].82,tcp_port_src=8080,tcp_port_dst=49211,bytes=149385,packets=193,metadata="mime-type:application/x-silverlight-2"
host = blue3 | source = /tmp/flow_logs/dpilog.log | sourcetype = dpi_log
```

Fig. 4: Silverlight Exploitation.

The attacker then starts a new process, notepad.exe, and migrates to the process so if the user closes iexplore.exe it won't close the meterpreter session (Fig. 5).

```
1460340194.32,kvmi_newproc,host_pid=60f0,vpid=11,eproc_hva=7f4306e68b30,peb_hva=7f42ccc48000,peb64=7efdf000,peb32=7efde000,cr3=13fd7000,pid=840,wow64=1,name="notepad.exe"
host = blue3 | source = /tmp/kvmi_logs/kvmi_newproc | sourcetype = kvmi_newproc
```

Fig. 5: notepad.exe Migration.

The attacker then uploads a binary and executes it. The binary is seen in Fig. 6, and also from the guest to the host for further inspection.

```
1460340204.11,kvmi_apihooks,host_pid=60f0,vpid=11,function="notepad.exe:ntdll.dll:NtCreateFile(PHANDLE:399e7b8, '\??\C:\Users\win7\Desktop\binary.exe')"
host = blue3 | source = /tmp/kvmi_logs/kvmi_apihooks | sourcetype = kvmi_apihooks
```

Fig. 6: binary.exe File Upload.

The attacker then exfils a file from the compromised VM's desktop to the attacker machine, we see this process started by a walk of the directory tree in Fig. 7.

```
1460340205.11,kvmi_apihooks,host_pid=60f0,vpid=11,function="explorer.exe:ntdll.dll:NtCreateFile(PHANDLE:3c4db28, '\??\C:\Users\win7\Desktop')"
host = blue3 | source = /tmp/kvmi_logs/kvmi_apihooks | sourcetype = kvmi_apihooks
```

Fig. 7: Directory Tree Walk.

The attacker then collects the local SAM hashes on the machine and passwords located in memory (kerberos, msv, and ssp passwords). As this information is transferred back to

the attacker machine, high entropy URIs are seen in the DPI log (Fig. 8) over a meterpreter bound port 2222.

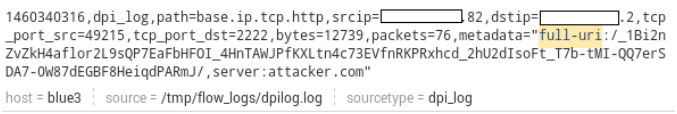


Fig. 8: High Entropy URI.

Using the new found credentials, the attacker logs into the domain controller (DC) (.66). On the DC the attacker again collects passwords and domain password hashes. Hashed URIs are shown traversing port 2222 from the AD to the Attacker server, as well as exfil communication from the AD to the attacker over port 3333 (Fig. 9).

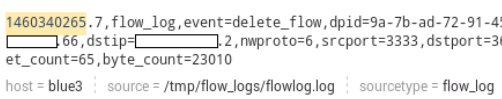


Fig. 9: Pivot Connection.

C. Use case 3: Using Cloud as a Relay

The final use-case examines the situation wherein an internet-connected node might be used a listening-post or a botnet drone waiting C2 commands. The underlying concept involves a targeted VM that is conscripted, running both legitimate and non-legitimate traffic/services.

1) Experiment Method: Several connections from the VM are made, combing both normal applications and malicious applications (as denoted by the experimenters).

2) Results: Using the KVMi sockets monitoring feature, the VM making connections and the endpoints (IPs) to which connections are made can be identified. What's novel is the binding of the network connection to the requesting application. As can be seen in Fig. 10, the VM (host process id 0xC27) can be seen making connections to IP .33 over port 80, with the process iexplorer.exe (Internet Explorer).

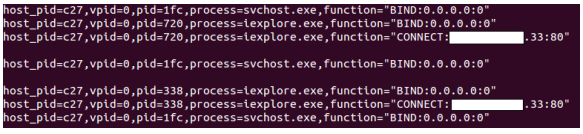


Fig. 10: KVMi Socket Logging.

VII. Conclusions and Future Work

There are several challenges that arise when conducting digital forensics and incident response in the Cloud. In our paper, we have discussed the challenges, current shortcomings, and proposed a unique approach and tools to meet those challenges. While we have made headway in developing our methodology and technology, there are still difficult problems and areas for improvement we're pursuing, such as: (1) extending KVMi for other platforms and various operating systems; (2) furthering KVMi's capability to make on-the-fly modifications to guest execution, such targeted encryption key extraction, or making certain suspicious actions trigger enhanced introspection; (3) further decouple KVMi from KVM, in both its memory accessing ability, and general execution. We are also in discussion with commercial hypervisor companies to extend the KVMi capability to their hypervisors.

Another avenue we're pursing is extending KVMi for general cloud security requirements. Since the hypervisor is the means through which cloud is managed, IT security professionals are concerned it may be leveraged as a vector to present attacks or unauthorized access to the virtual systems. Since KVMi is decoupled from the hypervisor, we are interested in using KVMi to detect, stifle or block attacks.

References

[1] Alberts, D. A, Killcrece G., Ruefle R.,and Zajicek M., "Defining incident management processes for CSIRTs: a work in progress," 2004
[2] Cichonski P., and Scarfone K., Computer security incident handling guide recommendations of the National Institute of Standards and Technology (NIST). Gaithersburg: NIST, 2012.
[3] Defense Information Systems Agency, Strategic Plan 2015-2020, 10 June 2015, http://www.disa.mil/News/Stories/2015/ /media/Files/DISA/About/Strategic-Plan.pdf
[4] [4] KVM, http://www.linux-kvm.org
[5] LibVMI, http://libvmi.com/
[6] Mell, P. and Grace, T., The NIST Definition of Cloud Computing, NIST Special Publication 800-145, Gaithersburg: NIST, 2011
[7] NIST Cloud Computing Forensic Science Challenges, Draft NISTR 8006, NIST Cloud Computing Forensic Science Working Group 3 Information Technology Laboratory, June 2014
[8] Patrascu A. and Patriciu, V., "Beyond Digital Forensics. A cloud computing perspective over incident response and reporting," 8th IEEE International Symposium on Applied Computational Intelligence and Informatics, May 2325, 2013
[9] Pichan, K. R., Lazarescu, M., and Soh, S. T., "Cloud forensics: Technical challenges, solutions and comparative analysis," Digital Investigation, Vol 13, 38-57, 2015
[10] Rahmana, N. H. Ab and Choo, K. R., "A survey of information security incident handling in the cloud," Computers and Security, Vol 49, pp 45-69, 2015
[11] Rekall, http://www.rekall-forensic.com/
[12] Volatility, https://github.com/volatilityfoundation
[13] Payne, Bryan D., Virtual Machine Introspection, Encyclopedia of Cryptography and Security, Springer US, Boston MA, pp 1360-1362, 2011
[14] Anonymized reference
[15] [15] Manoj, Sheik Khadar Ahmad, and D. Lalitha Bhaskari. "Cloud Forensics-A Framework for Investigating Cyber Attacks in Cloud Environment," Procedia Computer Science 85: pp 149-154, 2016
[16] Roussev, Vassil, et al. "Cloud forensicsTool development studies & future outlook." Digital Investigation, 2016
[17] Alqahtany, Saad, et al. "Cloud Forensics: A Review of Challenges, Solutions and Open Problems," Cloud Computing (ICCC), International Conference on. IEEE, 2015
[18] Farina, Jason, et al. "Overview of the forensic investigation of cloud services," Availability, Reliability and Security (ARES), 2015 10th International Conference on. IEEE, 2015
[19] Katilu, Victoria M., Virginia NL Franqueira, and Olga Angelopoulou. "Challenges of data provenance for cloud forensic investigations," Availability, Reliability and Security (ARES), 2015 10th International Conference on. IEEE, 2015
[20] Sndez, Manuel Jess Rivas. "A review of technical problems when conducting an investigation in cloud based environments," arXiv preprint arXiv:1508.01053, 2015