A Taxonomy of Hypervisor Forensic Tools

Anand Kumar Mishra, Mahesh Govil, Emmanuel Pilli

Chapter 10

A TAXONOMY OF HYPERVISOR FORENSIC TOOLS

Anand Kumar Mishra, Mahesh Govil and Emmanuel Pilli

Abstract Cloud computing models are deployed on a compute server whose hardware resources are virtualized to enable multiple virtual machines to run on a single physical system. Several types of virtualization such as bare metal and hosted virtualization are available along with virtualization modes such as full, paravirtualized, hardware-assisted and paravirtualized-hardware-assisted virtualization. Virtual machines are inaccessible from each other when the physical server hardware is abstracted in the full virtualization mode. Physical information such as hard disk drives and server memory are made available in a virtualized environment as a virtual hard disk, vCPU and guest operating system state.

Hypervisor operations generate copious amounts of data that are of value in forensic investigations of virtualized cloud environments. This chapter presents a taxonomy of hypervisor forensic tools, which provides a searchable catalog for forensic practitioners to identify specific tools that fulfill their technical requirements. A case study involving a KVM hypervisor demonstrates the evidence that can be found in a virtual machine at the virtual machine manager and host system layers.

Keywords: Cloud computing, hypervisors, forensic tool taxonomy

1. Introduction

In 2003, National Institute of Standards and Technology (NIST) [34] initiated the Computer Forensic Tool Testing (CFTT) Project to support the international digital forensics community. The project has classified computer forensic tools according to their specifications, test procedures, test criteria, test sets and test hardware. A similar taxonomy is required for cloud forensic tools.

Previous research has developed a taxonomy of cloud endpoint forensic tools [32]. This chapter extends the previous research by presenting a taxonomy for hypervisor forensic tools that considers the various layers of a hypervisor system. The chapter also discusses the potential data sources in virtual machines (VMs) and virtual machine managers (VMMs), and discusses the uses of the extracted data in forensic investigations. A case study using a KVM hypervisor demonstrates the valuable evidence that can be found in a virtual machine at the virtual machine manager and host system layers.

2. Hypervisors

A hypervisor is a software system that abstracts the storage, operating system (OS), network and applications. The software layer is implemented on top of hardware to enable multiple virtual machines to be created in isolation. The virtual machines incorporate a processor, memory, secondary storage and networking. The hypervisor also controls the host processor and assets, dispensing resources to virtual machines and ensuring that they are isolated from each other.

A virtualized environment has multiple layers. The hardware layer comprises the processor for computation, network interface card, memory and secondary storage. The host operating system layer is situated between the hardware and hypervisor layers. Virtual machines are created on top of the hypervisor layer. A guest operating system is installed in each virtual machine for user interactions and running applications. The applications execute in a virtualized environment in the guest operating system.

A hypervisor is a software system that virtualizes hardware resources and manages the resources for virtual machines. There are two types of hypervisors. In a Type 1 or bare-metal or native-type hypervisor, the hypervisor software runs directly on the computer system hardware. Example Type 1 hypervisors include VMware ESX and ESXi, Microsoft Hyper-V, Citrix XenServer and Oracle VM (based on open-source Xen). In a Type 2 or hosted or application level hypervisor, the hypervisor software runs on a host operating system that provides virtualization services such as input/output device support and memory management. Example Type 2 hypervisors include VMware Workstation/Fusion/Player, VMware Server, Microsoft Virtual PC, Oracle VM VirtualBox, Red Hat Enterprise Virtualization and KVM.

Depending of their underlying technologies, several types of virtualization techniques have been deployed, including full virtualization, hardware-assisted virtualization and paravirtualization. In full virtual-

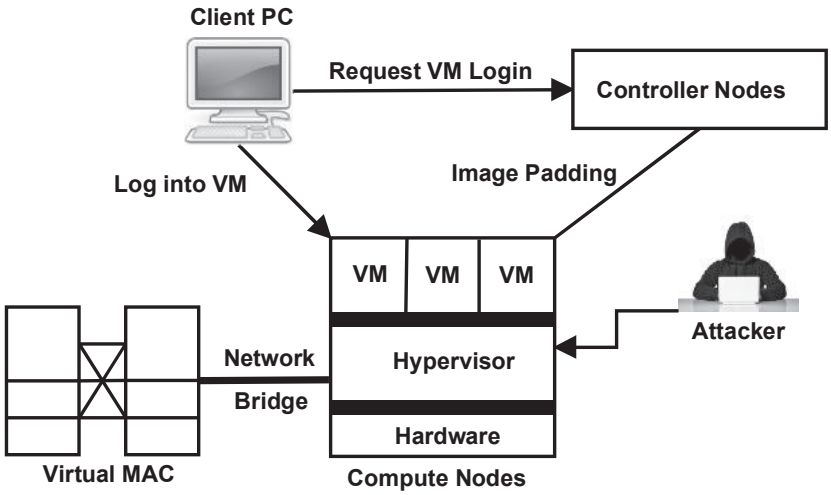

Figure 1. Attack on the hypervisor layer.

ization, a virtual machine runs in isolation; examples include VMware Workstation, VirtualBox (32-bit guests) and VMware Server. Hardware-assisted virtualization is a type of full virtualization that directly interrupts the hardware using virtualization technology, including processors such as Intel-VTx and AMD-V; examples include VMware ESXi/ESX, KVM, Hyper-V and Xen. Paravirtualization (operating system assisted virtualization) is installed on a physical server (host) and a guest operating system is installed in the environment. Unlike full virtualization, virtual guests are aware that they are virtualized; examples include Oracle VM for SPARC (LDOM) and Oracle VM for x86 (OVM).

3. Hypervisor Attacks and Vulnerabilities

Previous work [31] has discussed the top threats to cloud computing, including wrapping, malware-injection, flooding and browser attacks, insecure interfaces and APIs, malicious administrators, data theft and data leakage. Pearce et al. [37] have conducted a detailed study of virtualization techniques and the accompanying security threats.

When a cloud environment is attacked, the impacts can occur throughout the environment; this complicates evidence collection. Figure 1 shows an attack on the hypervisor layer of a private cloud. When a compute node is compromised, changes occur not only at the node, but also at the cloud controller node, hypervisor level and storage systems.

Forensic investigations of cloud environments are also challenging due to the movement of data within providers. Attacks on a hypervisor are

serious because they may crash the hypervisor and the guest or virtual machines. A vulnerable hypervisor can render every installed guest machine vulnerable. The increased asset utilization by a virtual machine that causes a denial-of-service attack on a service provider server is exacerbated when multiple virtual servers are involved. An attacker typically targets hypervisor services such as `created()`, `delete()`, `clone()` and `migrate()` to exploit and expand vulnerabilities.

3.1 System Calls and Hypercalls

System calls enable a user application to perform specific instructions that maintain the safety of user mode operations and kernel changes to the execution mode. In a system call. the kernel stack is initialized and the framework call handler is invoked. After the execution of a user request, execution returns to the user mode and the unprivileged register connection is restored. Control then returns to the instruction after the system call. Because of the discriminating extension between user applications and the host operating system, system call disruptions are attractive to malicious entities that have access to cloud services.

Attackers often focus on the hypervisor layer and leverage hypercalls, which are software traps from a kernel of a guest virtual machine to the hypervisor. Milenkoski et al. [30] have listed vulnerabilities in several hypercalls: `memory_op, gnttab_op, set_debugreg, physdev_op and mmuext_op` (Table 1). Because cloud computing architectures are based on virtualization, these hypercall vulnerabilities can affect cloud services. Due to the ubiquity of cloud computing, forensic investigations of hypercall-based attacks are on the increase.

Perez-Botero et al. [38] have analyzed vulnerabilities in the Xen and KVM hypervisors. Their analysis covers hypervisor functionalities, which are mapped to vulnerabilities and attack vectors.

3.2 Virtual Machine Introspection

Virtual machine introspection (VMI) is used to investigate real-time events in a virtual machine and to ensure that the virtual system is running properly. Garfinkel and Rosenblum [11] originally defined virtual machine introspection as examining a virtual machine from the outside with the goal of dissecting the software running inside it.

Virtual machine introspection enables an investigation to be conducted without interrupting the monitored virtual machine. Virtual machine introspection assists in malware collection, malware analysis, intrusion detection, intrusion prevention, stealthy debugging, cloud security and mobile security [47].

Table 1. Hypercall vulnerabilities.

Vulnerability	Hypercall	Description	Post-Attack State
CVE-2012-3496, CVE-2012-5513	`memory_op`	Management of virtual machine memory	Hypervisor crash, overwritten memory
CVE-2012-4539, CVE-2012-5510, CVE-2013-1964	`gnttab_op`	Management of shared memory among virtual machines	Hypervisor crash, virtual machine hanging, operation disruption
CVE-2012-3494	`set_debugreg`	Register value management of CPU allocated to a guest virtual machine	Hypervisor crash
CVE-2012-3495	`physdev_op`	Management of component requests by a guest virtual machine	Hypervisor crash, overwritten memory
CVE-2012-5525	`mmuext_op`	Management of memory pages	Hypervisor crash, invalid page information

Figure 2 shows the virtual machine introspection components. The components are:

- **Virtual Machine Introspection API:** This library module serves as an interface between a virtual machine introspection application and the virtual machine monitor.
- **Virtual Machine Introspection Application:** This application observes the monitored guest virtual machine using the virtual machine introspection API functions, which support memory introspection, data streaming and storage performance evaluation.
- **Guest OS Symbol Table:** This virtual machine introspection component collects low-level information that is acquired externally. The low-level information includes the virtual address, system call table and interrupt descriptor table (IDT).

Several tools and utilities have been developed to support hypervisor forensics. These include LibVMI, file carving tools, disk image mounting

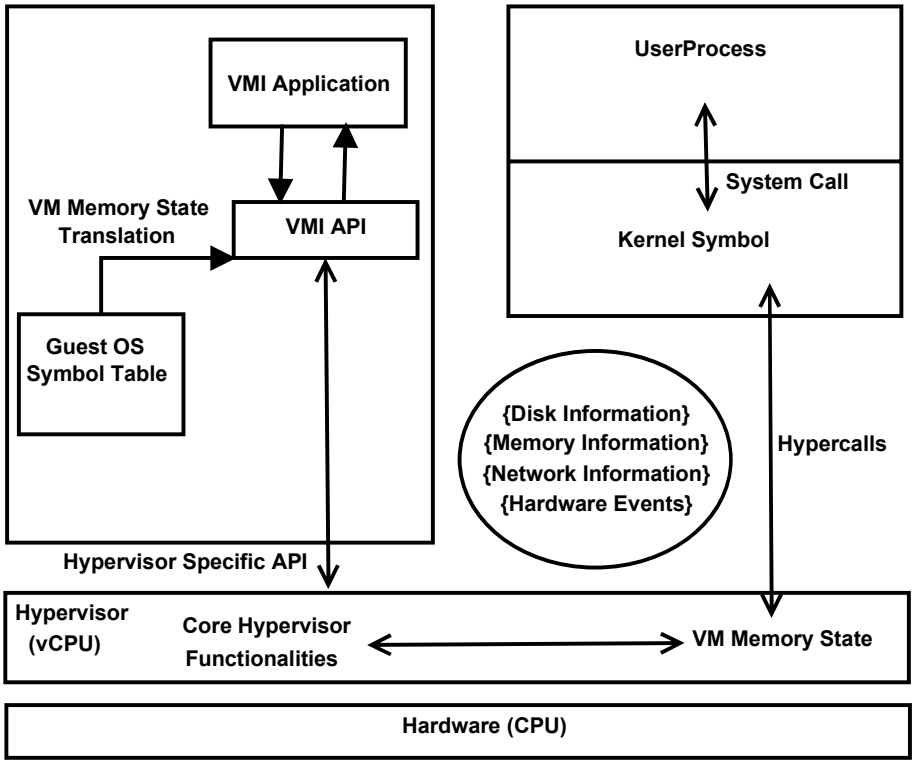

Figure 2. Virtual machine introspection components.

utilities, LiveView, Bitdefender Hypervisor Introspection and Volatility. Interested readers are referred to [16, 33] for details about virtual machine introspection techniques and their applications.

4. Taxonomy of Hypervisor Forensic Tools

Hypervisor forensics is the application of digital forensic techniques and tools to collect and analyze digital evidence for event construction, interpretation and reporting in order to prove hypervisor usability and exploitation. The primary goal of the hypervisor forensic tool taxonomy presented in this section is to provide a searchable catalog of digital forensic tools. Forensic practitioners can use the taxonomy to identify tools that meet the technical requirements of hypervisor investigations.

Figure 3 shows the taxonomy of hypervisor forensic tools. Evidentiary data can be extracted from five distinct layers or levels: (i) virtual machine layer; (ii) virtual machine manager layer; (iii) network layer; (iv) host operating system layer; and (v) hardware layer.

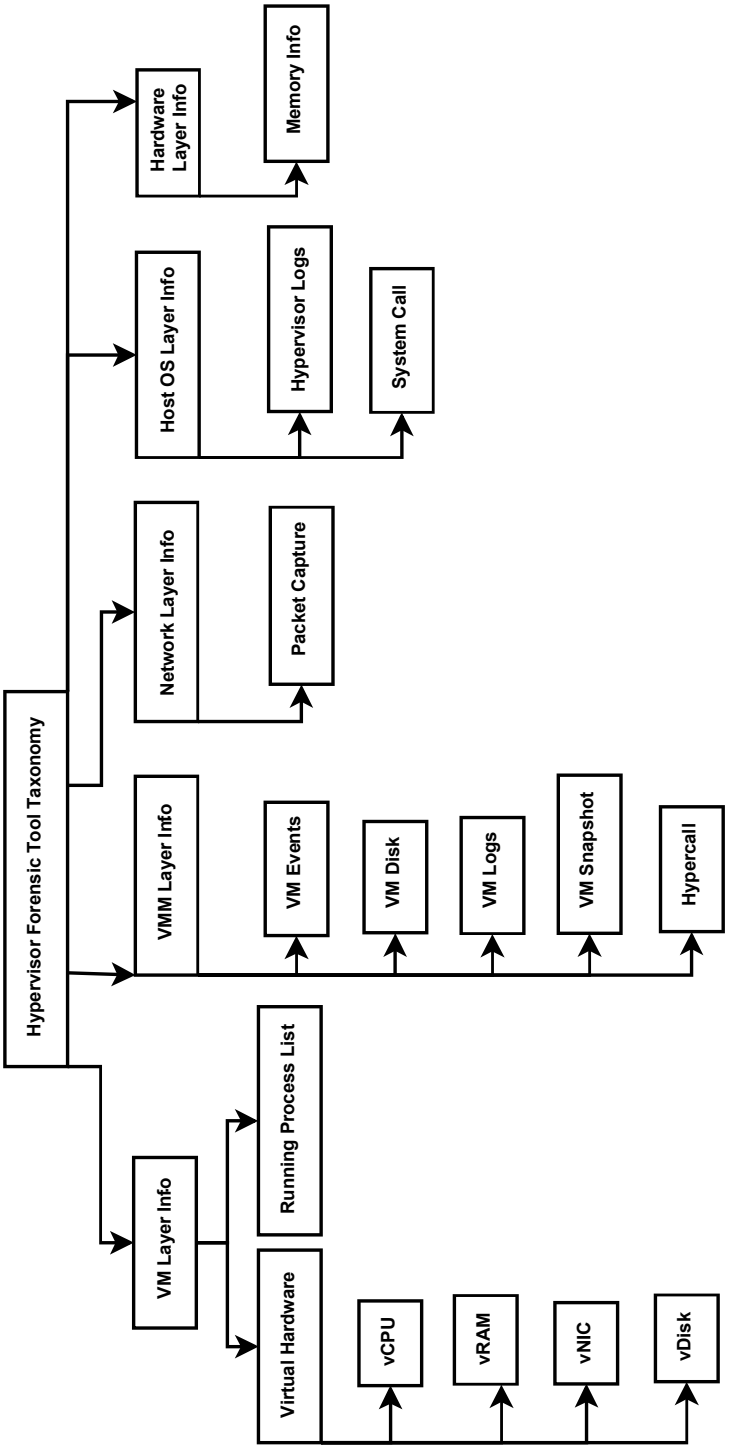

Figure 3. Taxonomy of hypervisor forensic tools.

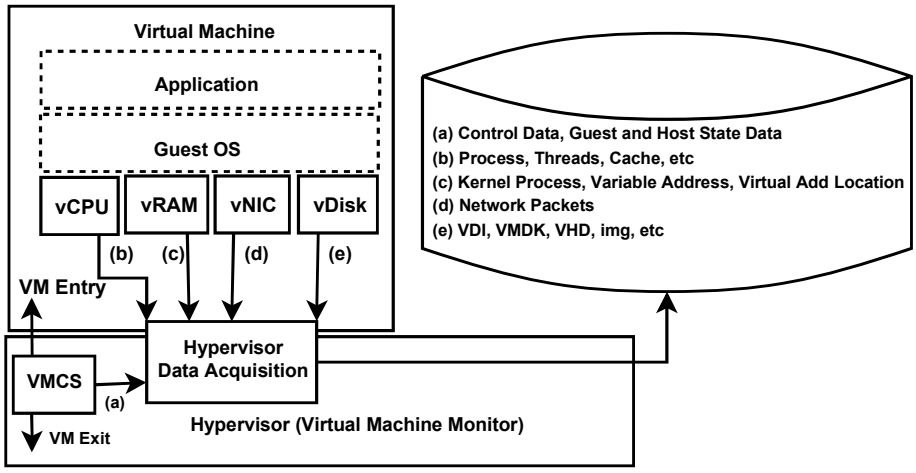

Figure 4. Virtual machine evidence acquisition.

4.1 Virtual Machine Layer Data

Virtual machines are the most important sources of evidence, including data that can support complete event reconstruction. However, virtual machine data is volatile and is an easy target for attackers.

Virtual machine data includes memory content, register contents, input/output device flags, Ethernet/Internet address changes, process list, kernel symbol table, virtual machine physical address space, guest page table, etc. In a virtual machine, vCPU, vRAM, vNIC and vDisk also provide useful data related to processes, threads and control data. Figure 4 shows the virtual machine evidence acquisition process.

4.2 Virtual Machine Manager Layer Data

Information residing in the virtual machine manager layer includes virtual machine logs, disk images, snapshots, configuration files, etc.

Shavers [41] has identified useful types of virtual machine files that reside in a VMWare hypervisor. These include virtual machine activity log files (`.log`), virtual machine disk files (`.vmdk`), paging files of running virtual machines (`.vmem`), virtual machine snapshots (`.vmsn`), metadata snapshots (`.vmsd`), virtual machine BIOS data (`.nvram`), stored virtual machine configurations (`.vmx`) and suspended virtual machine data (`.vmss`). Other important files are lock files created for configuration (`.lck`) and disk files of running virtual machines.

A virtual machine manager supports an interface similar to `/dev/kmem` that provides access to the monitored host's memory in the form of

a flat file. This enables the capture of virtual machine layer events such as virtual machine rebooting or powering down. Hypercalls can also be monitored to enable the analysis of guest machine execution. Additionally, copies of virtual machine images can be preserved.

Forensic tools such as FTK Imager and OSForensics can be used to mount a virtual machine on an external drive. The virtual machine manager needs the virtual machine control structure (VMCS) that holds all the data pertaining to virtual machine configuration and the rules that must be obeyed. This control structure contains a shadow indicator, indicator, guest state area, host state area, virtual machine execution control fields, virtual machine exit control fields, virtual machine entry control fields and virtual machine exit information fields. These fields are very important because the values are stored in registers.

4.3 Network Layer Data

Network layer evidentiary data is crucial when a live acquisition is performed to investigate an active network intrusion. Network managers help analyze and manage overall network traffic and performance.

4.4 Host Machine Layer Data

Hypervisor logs, system calls and hypervisor events are maintained in the host machine layer. An image of a host machine should also be preserved because it contains all the details of the virtual machine and virtual machine manager. Files and folders associated with virtual machines should also collected from the host machine.

4.5 Hardware Layer Data

A RAM capture is one of the most important tasks in live acquisition. This is because the capture contains the footprints of running processes that can be analyzed further in the case of malicious events.

5. Related Work

This section summarizes key research in the areas of hypervisor introspection and forensic investigations. Most of the approaches employ virtual machine introspection techniques for malware detection and/or vulnerability detection.

Tables 2 through 4 list research published from 2003 to 2018. The first column presents author-year information, the second column presents the hypervisor (virtual machine manager) used and the third column the forensic method or methods used. The fourth and fifth columns list

Table 2. Hypervisor introspection and forensic investigation research.

Authors	Hypervisor	Forensic Method	Layer	Extracted Data	Objective
Garfinkel et al., 2003 [11]	VMware Workstation	VMI	VMM to VM, VM to VM	Guest OS metadata	Intrusion detection
Joshi et al., 2005 [21]	User mode Linux	VMI	Host OS to VM	Guest OS processes, kernel processes	Vuln. detection
King et al., 2006 [22]	VMware Workstation Windows XP, Linux	VM-based rootkit, VM detection	VMM to VM	Keystrokes, packets, disk state, memory	Malware detection
Kourai et al., 2005 [24]	Persona OS FreeBSD	VM monitoring mechanisms	VMM to VM	Processes, packets, disk state	Intrusion detection
Quynh et al., 2007 [39]	Xen	Filesystem integrity tools	VMM to VM, VM to VM	System calls, log files	Intrusion detection, system monitoring
Jones et al., 2008 [20]	Xen v3.0.3	Cross-view validation	VMM to VM	Guest OS processes	Hidden
Gu et al., 2011 [13]	KVM Ubuntu 10.04	VMI	VMM to VM, inside VM	Process list	Malware tracing in VM
Dolan-Gavitt et al., 2011 [6]	QEMU	Trace logging, preproc., merging	VMM to VM, VM to VM	Process IDs, process list	Secure VMI
Thorpe et al., 2011 [42]; 2012 [44]	VMware ESXi	Log synch.	Inside VM	VM networks	Event reconstrn.
Harrison et al., 2012 [14]	Xen	Forensic VM and VM	VM to VM via hypervisor	Processes	Malicious behavior
Lim et al., 2012 [28]	VMware Workstation	Direct VM image	Inside VM	VM and vDisk config., VM	VM and VM state
Kourai et al., 2012 [23]	Xen	Packet filtering at VMM recovery	VMM to VM	Attack source, log info. log, etc.	Outbound attack detection, recovery

Table 3. Hypervisor introspection and forensic investigation research (continued).

Authors	**Hypervisor**	**Forensic Method**	**Layer**	**Extracted Data**	**Objective**
Alarifi et al., 2012 [1]	KVM	VM host system call analysis	Inside VM	VMM system calls	Anomaly detection
Deng et al., 2012 [5]	KVM	User level library call tracing	VM	Log files	Dynamic malware analysis
Fu et al., 2012 [8]	QEMU v0.15.50	OS level, binary code reuse	VMM to VM, VM to VM	VM logs, system calls	VMI
Wang et al., 2012 [45]	Xen	VMI	VMM to VM	Process list	Virus detection
Jin et al., 2013 [19]	Xen	Network packet capture	VM to VM	Packets	Anomaly detection, file integrity
Fu et al., 2013 [9]	QEMU v0.15.50	OS kernel code	VMM to VM	VM details	VMI
Fu et al., 2013 [10]	QEMU v1.0	Exterior data acq.	VMM to VM	VM details	VMI
Thorpe et al., 2013 [43]	VMware ESXi	VM log auditing	VMM, VM	VM events, hypervisor logs, kernel logs	VM log auditing
Graziano et al., 2013 [12]	HyperDbg, KVM, Xen, VirtualBox, VMware	VM control structure	VMM	Hypervisor RAM data	Memory acq. and analysis
Lamps et al., 2014 [26]	Xen	WinWizard	VMM to VM, VM to VM	VM data, VM layer info.	VMI
Kumara et al., 2015 [25]	Xen	System call tracing, LibVMI	VMM to VM, VM to VM	VM data	Malicious process detection
Xiao et al., 2016 [46]	QEMU	Hyperlink	VMM to VM, VM to VM	VM data	VMI

Table 4. Hypervisor introspection and forensic investigation research (continued).

Authors	Hypervisor	Forensic Method	Layer	Extracted Data	Objective
Jia et al., 2017 [17]	KVM	Trusted VMI model	VMM to VM, VM to VM	VM data	VMI
Riaz et al., 2018 [40]	VMware	VM data acquisition	VMM to VM	VM log files and snapshot	VMI

the layer or layers containing data and the extracted data, respectively. The sixth column provides the objective or objectives of the research.

Other research focusing on virtual machines and introspection methods include ReVirt [7], network attack detection using Collapsar [18], trustworthy intrusion detection using Psyco-Virt [4], virtual machine monitoring with XenAccess [35, 36], direct kernel structure manipulation attack analysis using virtual machine introspection [3], live digital forensic analysis using the Xen VIX tool [15], rootkit detection in Xen using Patagonix [29], peer-to-peer network monitoring using virtual machine introspection [2], and virtual machine privacy and integrity protection using CloudVisor [49] and CryptVMI [48].

6. KVM Hypervisor Forensics

In the KVM hypervisor forensics case study, LibVMI [27] was installed in a QEMU-KVM hypervisor v2.0.0 on a Linux Ubuntu 14.04.1 operating system (64-bit, v3.13.0-32 generic kernel). The KVM made it possible to spin up multiple virtual machines running unmodified Linux or Windows operating systems with private virtualized hardware, a network card, disk, graphics adapter, etc.

Figure 5 shows virtual machine data acquisition using hypervisor introspection. In the scenario, an attacker targets the virtualized environment containing the compute nodes C1, C2 and C3.

The LibVMI C language library provides low-level information about running virtual machines such as memory, process lists and process IDs. Using LibVMI, it is possible to examine process records, kernel module records, system call observations and memory page information. The core function of virtual machine introspection, specifically `vmi_read()`, makes it possible to read the virtual machine memory that supports translation, caching and hypervisor access. The translation of the kernel virtual address to the physical address is performed by `vmi_translate_kv2p`.

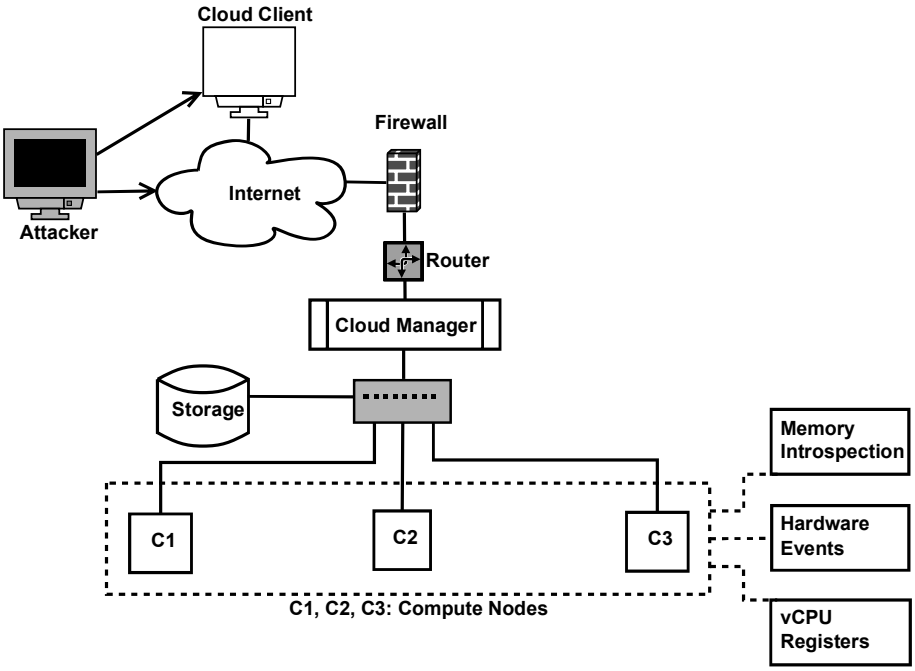

Figure 5. Virtual machine data acquisition.

```
def main(argv):
vmi = pyvmi.init(argv[1], "complete")
print vmi
print vmi.get_name()
print vmi.get_vmid()
print vmi.get_ostype()
print vmi.get_memsize()
for pid,x in list_processes(vmi):
print pid
print x
try:
print vmi.pid_to_dtb(pid)
```

Figure 6. Process ID to directory table base code snippet.

The code snippet in Figure 6 shows that, if the process ID (PID) is known, then the `vmi_pid_to_dtb` function returns the virtual address of the directory table base (DTB) for the process address space. This address is effectively in the CR3 control register while the process is exe-

cuting. The CR3 register, which indicates the page directory base, holds the physical address of the initial structure used for address translation.

The following data related to the KVM hypervisor and its virtual machines was obtained:

- **Process List:** The list of running processes was extracted via LibVMI using the command: `$sudo ./process-list VIRTUAL_MACHINE_NAME`.
- **Disk Images and Formats:** Virtual machine disk images with format `qcow2` at `/var/lib/libvirt/images` were obtained. The image format can be converted using the command: `$ sudo qemu-img convert -O qcow2 vm1.img vm1.qcow2`.
- **Virtual Machine Logs:** The log file for each running virtual machine at `/var/log/libvirt/qemu/VM.log` was obtained.
- **SSH Login:** SSH login information at `/var/log/auth.log` along with the IP addresses and login times were obtained.
- **Audit Logs:** Hypervisor audit logs at `/var/log/audit/audit.log` were obtained.

7. Conclusions

Hypervisor operations generate considerable data that is of evidentiary value in forensic investigations of virtualized environments. The evidence may be extracted from multiple layers – virtual machine layer, virtual machine manager layer, host operating system layer, network layer and hardware layer. As such, there is a need for forensic tools that can extract hypervisor-based native artifacts from virtualized environments with minimum effort and time. The taxonomy of hypervisor forensic tools provides a searchable catalog that assists forensic practitioners in identifying specific tools that fulfill their technical requirements. Additionally, the taxonomy could play a vital role in steering the development of standard forensic tools for virtualized environments.

Future research will enhance the tool taxonomy by incorporating features that cover the entire hypervisor forensic process, including acquisition and analysis.

References

[1] S. Alarifi and S. Wolthusen, Detecting anomalies in IaaS environments through virtual machine host system call analysis, *Proceedings of the International Conference on Internet Technology and Secured Transactions*, pp. 211–218, 2012.

[2] R. Ando, Y. Kadobayashi and Y. Shinoda, Blink: Large-scale P2P network monitoring and visualization system using VM introspection, *Proceedings of the Sixth International Conference on Networked Computing and Advanced Information Management*, pp. 351–358, 2010.

[3] S. Bahram, X. Jiang, Z. Wang, M. Grace, J. Li, D. Srinivasan, J. Rhee and D. Xu, DKSM: Subverting virtual machine introspection for fun and profit, *Proceedings of the Twenty-Ninth IEEE Symposium on Reliable Distributed Systems*, pp. 82–91, 2010.

[4] F. Baiardi and D. Sgandurra, Building trustworthy intrusion detection through VM introspection, *Proceedings of the Third International Symposium on Information Assurance and Security*, pp. 209–214, 2007.

[5] Z. Deng, D. Xu, X. Zhang and X. Jiang, IntroLib: Efficient and transparent library call introspection for malware forensics, *Digital Investigation*, vol. 9(S), pp. S13–S23, 2012.

[6] B. Dolan-Gavitt, T. Leek, M. Zhivich, J. Giffin and W. Lee, Virtuoso: Narrowing the semantic gap in virtual machine introspection, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 297–312, 2011.

[7] G. Dunlap, S. King, S. Cinar, M. Basrai and P. Chen, ReVirt: Enabling intrusion analysis through virtual-machine logging and replay, *ACM SIGOPS Operating Systems Review*, vol. 36(SI), pp. 211–224, 2002.

[8] Y. Fu and Z. Lin, Space traveling across VM: Automatically bridging the semantic gap in virtual machine introspection via online kernel data redirection, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 586–600, 2012.

[9] Y. Fu and Z. Lin, Bridging the semantic gap in virtual machine introspection via online kernel data redirection, *ACM Transactions on Information and System Security*, vol. 16(2), article no. 7, 2013.

[10] Y. Fu and Z. Lin, EXTERIOR: Using a dual-VM based external shell for guest OS introspection, configuration and recovery, *ACM SIGPLAN Notices*, vol. 48(7), pp. 97–110, 2013.

[11] T. Garfinkel and M. Rosenblum, A virtual machine introspection based architecture for intrusion detection, *Proceedings of the Network and Distributed Systems Security Symposium*, pp. 191–206, 2003.

[12] M. Graziano, A. Lanzi and D. Balzarotti, Hypervisor memory forensics, *Proceedings of the Sixteenth International Workshop on Recent Advances in Intrusion Detection*, pp. 21–40, 2013.

[13] Z. Gu, Z. Deng, D. Xu and X. Jiang, Process implanting: A new active introspection framework for virtualization, *Proceedings of the Thirtieth IEEE International Symposium on Reliable Distributed Systems*, pp. 147–156, 2011.

[14] K. Harrison, B. Bordbar, S. Ali, C. Dalton and A. Norman, A framework for detecting malware in the cloud by identifying symptoms, *Proceedings of the Sixteenth IEEE International Enterprise Distributed Object Computing Conference*, pp. 164–172, 2012.

[15] B. Hay and K. Nance, Forensic examination of volatile system data using virtual introspection, *ACM SIGOPS Operating Systems Review*, vol. 42(3), pp. 74–82, 2008.

[16] Y. Hebbal, S. Laniepce and J. Menaud, Virtual machine introspection: Techniques and applications, *Proceedings of the Tenth International Conference on Availability, Reliability and Security*, pp. 676–685, 2015.

[17] L. Jia, M. Zhu and B. Tu, T-VMI: Trusted virtual machine introspection in cloud environments, *Proceedings of the Seventeenth IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 478–487, 2017.

[18] X. Jiang and D. Xu, Collapsar: A VM-based architecture for a network attack detention center, *Proceedings of the Thirteenth USENIX Security Symposium*, pp. 15–28, 2004.

[19] H. Jin, G. Xiang, D. Zou, S. Wu, F. Zhao, M. Li and W. Zheng, A VMM-based intrusion prevention system in a cloud computing environment, *Journal of Supercomputing*, vol. 66(3), pp. 1133–1151, 2013.

[20] S. Jones, A. Arpaci-Dusseau and R. Arpaci-Dusseau, VMM-based hidden process detection and identification using Lycosid, *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pp. 91–100, 2008.

[21] A. Joshi, S. King, G. Dunlap and P. Chen, Detecting past and present intrusions through vulnerability-specific predicates, *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, pp. 91–104, 2005.

[22] S. King and P. Chen, SubVirt: Implementing malware with virtual machines, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 314–327, 2006.

[23] K. Kourai, T. Azumi and S. Chiba, A self-protection mechanism against stepping-stone attacks for IaaS clouds, *Proceedings of the Ninth International Conference on Ubiquitous Intelligence and the Ninth International Conference on Autonomic and Trusted Computing*, pp. 539–546, 2012.

[24] K. Kourai and S. Chiba, HyperSpector: Virtual distributed monitoring environments for secure intrusion detection, *Proceedings of the First ACM/USENIX International Conference on Virtual Execution Environments*, pp. 197–207, 2005.

[25] M. Kumara and C. Jaidhar, Virtual machine introspection based spurious process detection in virtualized cloud computing environments, *Proceedings of the International Conference on Futuristic Trends in Computational Analysis and Knowledge Management*, pp. 309–315, 2015.

[26] J. Lamps, I. Palmer and R. Sprabery, WinWizard: Expanding Xen with a LibVMI intrusion detection tool, *Proceedings of the Seventh IEEE International Conference on Cloud Computing*, pp. 849–856, 2014.

[27] LibVMI Community, LibVMI: LibVMI Virtual Machine Introspection, LibVMI (`libvmi.com`), 2020.

[28] S. Lim, B. Yoo, J. Park, K. Byun and S. Lee, A research on the investigation method of digital forensics for a VMware Workstation virtual machine, *Mathematical and Computer Modeling*, vol. 55(1-2), pp. 151–160, 2012.

[29] L. Litty, H. Lagar-Cavilla and D. Lie, Hypervisor support for identifying covertly executing binaries, *Proceedings of the Seventeenth USENIX Security Symposium*, pp. 243–258, 2008.

[30] A. Milenkoski, M. Vieira, B. Payne, N. Antunes and S. Kounev, Technical Information on Vulnerabilities of Hypercall Handlers, *arXiv:* 1410.1158v1, 2014.

[31] A. Mishra, P. Matta, E. Pilli and R. Joshi, Cloud forensics: State-of-the-art and research challenges, *Proceedings of the International Symposium on Cloud and Services Computing*, pp. 164–170, 2012.

[32] A. Mishra, E. Pilli and M. Govil, A taxonomy of cloud endpoint forensic tools, in *Advances in Digital Forensics XIV*, G. Peterson and S. Shenoi (Eds.), Springer, Cham, Switzerland, pp. 243–261, 2018.

[33] A. More and S. Tapaswi, Virtual machine introspection: Towards bridging the semantic gap, *Journal of Cloud Computing*, vol. 3, article no. 16, 2014.

[34] National Institute of Standards and Technology, Computer Forensic Tools and Techniques Catalog, Gaithersburg, Maryland (`tool catalog.nist.gov`), 2019.

[35] B. Payne, M. Carbone and W. Lee, Secure and flexible monitoring of virtual machines, *Proceedings of the Twenty-Third Annual Computer Security Applications Conference*, pp. 385–397, 2007.

[36] B. Payne, M. Carbone, M. Sharif and W. Lee, Lares: An architecture for secure active monitoring using virtualization, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 233–247, 2008.

[37] M. Pearce, S. Zeadally and R. Hunt, Virtualization: Issues, security threats and solutions, *ACM Computing Surveys*, vol. 45(2), article no. 17, 2013.

[38] D. Perez-Botero, J. Szefer and R. Lee, Characterizing hypervisor vulnerabilities in cloud computing servers, *Proceedings of the International Workshop on Security in Cloud Computing*, pp. 3–10, 2013.

[39] N. Quynh and Y. Takefuji, A novel approach for a filesystem integrity monitor tool for a Xen virtual machine, *Proceedings of the Second ACM Symposium on Information, Computer and Communications Security*, pp. 194–202, 2007.

[40] H. Riaz and M. Tahir, Analysis of VMware virtual machine in forensics and anti-forensics paradigms, *Proceedings of the Sixth International Symposium on Digital Forensics and Security*, 2018.

[41] B. Shavers, A Discussion of Virtual Machines Related to Forensic Analysis, *Forensic Focus*, November 2008.

[42] S. Thorpe, I. Ray and T. Grandison, A synchronized log cloud forensic framework, presented at the *International Conference on Cybercrime, Security and Digital Forensics*, 2011.

[43] S. Thorpe, I. Ray, T. Grandison, A. Barbir and R. France, Hypervisor event logs as a source of consistent virtual machine evidence for forensic cloud investigations, *Proceedings of the Twenty-Seventh Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*, pp. 97–112, 2013.

[44] S. Thorpe, I. Ray, I. Ray, T. Grandison, A. Barbir and R. France, Formal parameterization of log synchronization events within a distributed forensic compute cloud database environment, *Proceedings of the Third International ICST Conference on Digital Forensics and Cyber Crime*, pp. 156–171, 2012.

[45] L. Wang, Y. Peng, W. Liu and H. Gao, VMSecurexec: Transparent on-access virus detection for virtual machine in the cloud, *Proceedings of the Symposium on ICT and Energy Efficiency and Workshop on Information Theory and Security*, pp. 116–121, 2012.

[46] J. Xiao, L. Lu, H. Wang and X. Zhu, HyperLink: Virtual machine introspection and memory forensic analysis without kernel source code, *Proceedings of the IEEE International Conference on Autonomic Computing*, pp. 127–136, 2016.

[47] H. Xiong, Z. Liu, W. Xu and S. Jiao, LibVMI: A library for bridging the semantic gap between guest OS and VMM, *Proceedings of the Twelfth IEEE International Conference on Computer and Information Technology*, pp. 549–556, 2012.

[48] F. Yao, R. Sprabery and R. Campbell, CryptVMI: A flexible and encrypted virtual machine introspection system in the cloud, *Proceedings of the Second International Workshop on Security in Cloud Computing*, pp. 11–18, 2014.

[49] F. Zhang, J. Chen, H. Chen and B. Zang, CloudVisor: Retrofitting protection of virtual machines in a multi-tenant cloud with nested virtualization, *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pp. 203–216, 2011.