Patrick Tobin, M-Tahar Kechadi.

Virtual Machine Forensics by means of Introspection and Kernel Code Injection

School of Computer Science and Informatics, University College Dublin

Abstract

Virtual Machine Introspection offers the ability to access a virtual machine remotely and extract information from it. Virtual machine introspection allows all processes, local data, and network traffic to be tracked and made available to the investigation process. These properties offer the possibility to monitor a suspect virtual machine (VM). Moreover, the access to a VM data is far from being trivial; there are various complex tasks to be dealt with. For instance the returned data is in a raw format, and it is necessary to remap into a user friendly representation (canonical representation). In this paper we propose a method of bridging this semantic gap, and provide a graphical reconstruction of events. This proposal is essentially, the recreation of a virtual machine at a remote location and the subsequent recreation of all processes, data, network traffic in a virtual machine as they occur in the original. This should be achieved in real-time, which will give an opportunity to quickly make decisions based on the evidence as we collect them in real-time. Our approach involves recreating a virtual machine and injecting into it all code and data within the original virtual machine, presenting an identical copy for examination. The approach proposed also has another advantage by allowing all data to be saved for further analysis and verification.

Keywords: Virtualisation, Virtual Machine Introspection, Digital Forensics, Kernel Injection

1 Introduction

The expansion in use of cloud computing has occurred in a relatively short period of time, services have been developed and adapted for use in cloud computing, and new services are proposed. The growth in popularity and use of cloud computing as a commodity has undoubtedly led to an expansion of its use in crime. Criminals are generally quick to identify and avail of opportunities, whereas the response of law enforcement (LE) is primarily reactive, rather than proactive. It is not always possible to foresee what crimes may be committed so it is therefore not always possible to prepare for them, software and tools generally being developed in response to digital crime, rather than developed to anticipate crimes.

What is proposed in this paper will not solve all the problems associated with investigating digital crime on virtual machines, but can provide a framework from which these crimes may be investigated. This proposal will facilitate the capture of evidence from a VM as it is being processed, analyse processes and extract relevant data, urls, email addresses and other evidence. It will also be possible to recover encryption keys, identify users and record times and movements of data. This would provide investigators with an opportunity to make instant decisions based on the best available evidence. It would also allow evidence to be gathered which would provide a precise timeline of events and actions conducted on the original VM, recording each event and allow later reconstruction of the virtual machine. This approach will not interfere with any other VM, nor will it compromise the confidentiality of others, both very important factors to be considered. This paper explores some of the issues associated with the forensic examination of VMs and proposes a new model to assit in the capture of evidence from them.

In Section 2 we will discuss the approach to the problem to be investigated, current investigative processes will be briefly outlined. Section 3 will discuss virtualisation and introduce the Xen hypervisor and some of its utilities. Section 4 will look at virtual machine replication and examine issues foreseen thus far, code injection and memory injection will also be discussed. Section 5 concludes the paper and gives some future research directions.

2 Approach and Problem Overview

Cloud computing is a frequently used expression among many in the computing and wider industries, but what is could computing? Many definitions for the term may be found in academia and industry, but one definition appears to stand out from others. The U.S. National Institute of Standards and Technology have described 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* (Mell and Grance, NIST, 2011).
Cloud computing offers a solution to many companies IT needs, it can deliver scalable, elastic services and its pay per use pricing structure is attractive to companies wishing to control costs (Armbrust, et al., 2010). It

offers companies an economic way of outsourcing their computing requirements rather than incurring costs maintaining legacy computer systems, allowing them to focus on their core activities. It provides an effective solution to their data storage needs by allowing them to migrate their data to cloud servers with very high availability, of typically up to 99.99% (Padhy & Patra, 2013). In a forensic context these features pose new challenges to Law Enforcement (LE) in terms of practices, tools and procedures.

In traditional digital forensics there is material evidence to investigate, such as harddrives, flash memory, CDs and DVDs. Migrating computing requirements to the cloud leads to an absence of real, physical evidence (Taylor, et al., 2011). Traditional digital investigations require storage devices to be seized for subsequent forensic analysis; consequently it is not feasible to carry out a traditional investigation on a cloud server, in the absence of such material evidence. Using traditional techniques, to seize one virtual machine may require taking a server offline and copying, or removing mass storage devices from that server. The critical importance of maintaining services for many users, perhaps numbering into the hundreds or thousands, takes importance over the need to preserve possibly a single virtual machine. Most importantly the requirement to preserve the privacy and confidentiality of other users is paramount (Ruan, Carthy, Kechadi and Crosbie, 2011) . A different approach is therefore required.

Digital investigative tools and techniques have been in use in the LE communities for many years and are accepted by Courts, investigators and legal professionals. These tools, practices and frameworks have been designed for use in the static analysis of physical computer systems, they have not been designed for the analysis of VMs. They further assume that the storage media are available and under the control of investigators (Grispos, Glisson and Storer, 2011). Current tools provide the ability to reveal evidence within a computer system which can include log files, deleted files, documents and programs, and hidden and obfuscated evidence, but generally fail to capture data in volatile random access memory (RAM) (Hay and Nance, 2008).

A traditional digital forensics investigation follows a number of predefined steps. These are the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of evidence (Palmer, 2001). These are important steps, and are usually applied to quiescent systems, i.e. those that have been powered off. Powering off a computer, either by shutting it down, cutting power through removal of the battery or pulling the plug, can cause the loss of valuable potential evidence from volatile memory such as RAM, with no possibility of recovery of those data (Damshenas, et al., 2012). Similarly closing a VM has the same effect and can result in the same loss of volatile data.

The collection phase of the traditional investigative process usually involves seizing a storage device from the suspect computer (Kerr, 2005; Mason and George, 2011), any ancillary storage devices and then carrying out analysis on those devices. Forensic investigation of virtual machines in a cloud computing environment precludes this seizure phase, simply because the physical mass storage devices are absent from the virtual machine, these being hosted by the cloud service provider. Sometimes the seizure phase involves recovering data from RAM and when it does, very valuable evidence including details of processes that are running, passwords, encryption keys, documents, images, etc., can be recovered, if these data can be captured and analysed they could provide useful evidence. One more important factor worth consideration is that hard drives, including virtual hard drives, are today commonly encrypted, users might also encrypt their data before saving them to the cloud, however data resident in RAM are generally not encrypted. Being able to recover unencrypted data is a very significant advantage to our approach (Casey and Stellatos, 2008).

A key element in the provision of cloud computing and cloud based services is virtualisation (Xing and Zhan, 2012). This is partly due to some key features of virtualisation such as isolation, consolidation and workload sharing across peers. Isolation provides security to cloud users by not allowing others to access any other user’s data and resources. Consolidation allows users to centralise their storage, servers and work processes. Workload sharing distributes the workload, improving performance. These properties of virtualisation in a cloud environment are important and have helped its expansion and adoption. This effectively means that a VM may share a single pool of resources with other VMs.

Virtual machine monitors (VMMs), also known as hypervisors, such as Citrix's Xen, VMWare ESXi and Microsoft HyperV, are used to manage VMs and their various hardware resources, CPUs, RAM, NICs, hard drive, etc. These hypervisors can be used to pause, stop, save, destroy, create, clone and dump a VM. Cloning allows a VM to be copied in its entirety and to be started in a separate hardware environment from that in which it was started. A core dump allows saving the machine current state, including operating system processes and all its associated processes states and its memory content. This would allow offline analysis of the core dump facilitating the identification and extraction of evidence which might otherwise be unavailable. In the terms of this project these are both important attributes which will be used.

3 Virtualisation

Virtualisation has been a feature of computing since 1966 when IBM first introduced time sharing on the VM/370 System (Creasy, 1981). Virtualisation has developed since and was further expanded by the introduction of Type I and Type II virtual machine monitor (VMM) implementations (Goldberg, 1974) Type I hypervisors are widely used in 'cloud' computing technologies They run directly on the host hardware, monitoring their VMs and administering CPU time, and other hardware usage. Type II hypervisors run inside an operating system and support VMs running concurrently, through the resident operating system. A Type I hypervisor - the Xen hypervisor - will be used throughout this research.

VMs differ from traditional systems which may be seized for investigation, they share CPU's, harddrives, NICs, RAM, etc. with other VMs or systems. In addition they may run on remote hardware, which can be located anywhere, perhaps even in a different country. This makes it difficult to take physical possession of them and adds complexity to their forensic investigation. The hardware may be beyond the control or powers of the investigating LE, probably subject to different legislative frameworks and search and seizure powers. These issues require a more specialised investigative approach. The framework being developed will subject that evidence to, and be accessible using, local jurisdictional powers.

One technology that may provide insight into VMs is Virtual Machine Introspection (VMI). VMI is the external examination of processes and data in a virtual machine to understand what is happening within that virtual machine. VMI was introduced by Garfinkel and Rosenblum (Garfinkel and Rosenblum, 2003) and used to detect intrusion. They describe how VMI can be used to inspect the hardware state of a VM, how processes can be monitored and how to extract data from a system.

Applying VMI to digital forensics is not new and its forensic possibilities have been researched by Nance, Bishop and Hay (Nance, Bishop and Hay, 2008, Nance et al, 2009). They suggest that VMI may be able to perform live analysis in such a way that it is not detectable by the target system. This is important in the context of this paper, if a user knows they are being scrutinised then there is little likelihood that they will do anything wrong. Nance *et al.* also insisted on the importance of not changing the state of the target system. The approach outlined in this paper will result in some modifications of the VM kernel through code injection, but these changes are minimal and can be fully detailed and explained. This is significant as this complies with Principle 1 of the ACPO Good Practice Guide for Digital Evidence (Williams, 2012). The approach proposed by Nance *et al.* suggests that live analysis may be carried out on a VM, Hay and Nance (Hay and Nance, 2008) also described how VMI can be used as a tool to examine volatile data. Our approach will provide this by allowing all data collected in the VM, including volatile data, to be streamed to a monitoring station where the VM can be recreated, data can be analysed and evidence identified and saved.

3.1 The Xen Hypervisor

Xen is a popular and widely used hypervisor (Younge et al., 1011). It runs in a modified Linux kernel and is available for many distributions of Linux, including Debian, Gentoo, Red Hat Linux, and their derivatives. It provides VMs with a hardware abstraction that resembles the underlying hardware allowing software that can run on that hardware to run in a virtual machine (Mendel and Rosenblum, 2005). It is a powerful Type I hypervisor which provides strong isolation between instances running within it, it manages resources and can host a variety of operating systems (Barham, et al). Xen is used by Amazon, Google, Rackspace among others, either in its native or customised adaptation.

Xen has some very useful features which can be used in the context of the proposed approach. It facilitates migration or cloning of a VM, whether it is live, paused or shut down. It also supports live core dumps, where the core dump is taken from a running VM, without interfering with and without any interaction with the VM. These are very useful features in the context of this research.

4 Virtual Machine Replication

Cloning a VM in Xen is not difficult, but reconstructing a VM will not be trivial. The VM would first need to be initialised on a monitoring station, then to maintain the cloned VM in a state of execution identical to the original VM, would require a core dump of the original at very regular intervals. The time intervals between core dumps, to maximise efficient and accurate reconstruction, have not yet been determined, this will be resolved after experimentation. It is expected that very large volumes of data will need to be processed and transferred via the network. To reduce the volume of data to be transferred it will be necessary to map the most recent core dump against the core dump immediately preceding, identify changes (the change file) and only transmit and inject those changes. These changes can include processes started or closed, ports opened or closed, data transmitted and received, files opened, etc. This reconstruction model would provide much efficiency and should substantially reduce the volume of data to be transmitted. It would expose the core dump for off-line extraction of passwords, usernames, data, files and processes, independent to the reconstruction phase. Another advantage is that each core dump, and its change file, can be stored and reopened at any time.

Extraction from the core dump of only those data that have changed would also produce significant reductions in terms of the complexity of reconstructing the target. A smaller volume of data would need to be injected, the time taken to reconstruct would be significantly reduced, allowing more frequent updates to be carried out. There will also be complications to overcome, particularly in mapping memory addresses. Injecting a change file into the replicated VM would require injection into memory addresses corresponding to those of the target. This would displace data at those addresses, and re-mapping those data to their new addresses will be a complicated issue. One avenue being researched is identifying those data displaced, compared against previous core dumps and relocating those memory addresses to each new address and reusing these data.

Many factors will have to be considered in replicating a VM at a remote location. These will include the volume of data needed for reconstruction and which data would be needed, how to update the replica and which data would be needed to update, clone the entire VM or inject data into a 'bare bones' duplicate of the operating system, these questions need to be researched. However if it is possible to reconstruct a duplicate of a target VM the forensic possibilities of this, and therefore the evidential value of it, can be enormous. Being able to view what a suspect is doing in real-time, being able to account for changes to files, data, processes and memory, to recover passwords, user names, user specific and hidden data, being able to locate and recover malware and being able to detail each and every change a user makes or causes, are just some of the possibilities of reconstruction. Not only can it help investigators, but to be able to present to a Court, a jury or others an exact timeline of what was carried out and by whom, to whom data were sent, what those data were and what changes occurred in a system, and to be able to precisely and graphically replicate these, is evidence that carries great weight. Together these facts can provide a very secure body of evidence supported through a variety of logs, including operating system logs, network logs, cloud service provider logs detailing each operation, evidence that would be very difficult to refute.

4.1 Code Injection and Memory Injection

Code injection attacks are a known method of using a vulnerability to inject malicious code into an application and executing that code (Wei et al). This form of attack is usually used to compromise a kernel for example by injecting a rootkit, (Seshadri, Luk, Qu and Perrig; Rhee, Riley, Xu and Jiang) to allow an attacker to control the kernel or to control or corrupt a process and its output. This kind of attack strategy could also be used to insert known good code into a kernel and allow that kernel execute that code without issue (Chiueh, Conover and Montague). If the injection is carried out properly this should not cause any exceptions and should execute correctly.

Code will be injected into both the target VM and the replicated VM. The replicated VM will have RAM from the target injected into it at regular intervals, but the target VM will require a different process. A kernel hook will have to be inserted into the target VM's kernel, similar to that used in a rootkit. It will have to satisfy the forensic standards described in the ACPO Good Practice Guide (Williams, 2012). The hook will have to follow a certain set of rules. It would be needed to stream data from the target VM to the replicated VM. It would have to be able to remain undetected, not interfere with processes and consume few of the resources of the target VM. It is also proposed that the hook injected should be able to receive limited instructions from an external user. Such instructions could include those to prevent shutdown of the VM or any process, prevent the deletion, destruction or corruption of possible evidence, if this was found to be necessary, and to preserve the VM for further examination. Further research will be necessary to establish whether this is possible or not.

5 Conclusion

Advantages of this approach include the accessibility of all evidence in a VM at a local level The opportunity of saving data to local storage, thus circumventing the threat of that evidence being deleted or being removed from the reach of local authorities and making them available to local investigators. This proposal also allows data to be recorded, as evidence that an offence or activity has taken place. Actions taken by the suspect can also be recorded (Mason and George, 2011). The risk that it may be alleged that evidence has been tampered with, or selectively chosen is minimised as all data, including logs are saved detailing each change. Just as the obstacles to be overcome are enormous, the benefits will be equally enormous. Cloud computing will become a safer enterprise for all. Users can engage with their customers safer in the knowledge that their transactions and data are more secure and LE can undertake investigations knowing that they will only target those that they are investigating and be able to preserve the integrity and privacy of all other users.

The approach proposed does not involve any significant modifications to the VM kernel. Detection should not be possible, thereby minimising the possibility of a user interfering with this process. The cloned VM is quickly updated providing investigators with a near real-time view of what is happening within the target VM.

5.1 Future Work

This paper presents a very high level overview of what is involved. Much more work has to be carried out to analyse the approach and make this concept a reality. Research will have to be carried out into code injection into a kernel and how that will affect current running processes. The forensic implications of code injection will have to be studied and its effect on evidence will have to be assessed. This framework will open new possibilities of secure, forensically sound evidence gathering. This will give new opportunities to LE to gather evidence, preserve virtual machines and identify new suspects.

References

Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee et al. "A view of cloud computing." *Communications of the ACM* 53, no. 4 (2010): 50-58.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. "Xen and the art of virtualization*." ACM SIGOPS Operating Systems Review 37, no. 5 (2003): 164-177.*

Casey, Eoghan and Gerasimos J. Stellatos. "The impact of full disk encryption on digital forensics." *ACM SIGOPS Operating Systems Review,* 43, no. 3 (2008): 93-98

Chiueh, T., Conover, M., and Montague, B. "Surreptitious Deployment and Execution of Kernel Agents in Windows Guests." In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pp. 507-514. IEEE Computer Society, 2012.

Creasy, R. J., "*The Origin of the VM/370 Time-sharing System*", IBM Journal of Research and Development, vol 25, no. 5, pp. 483,490, Sep. 1981

Damshenas, Mohsen, Ali Dehghantanha, Ramlan Mahmoud and Solahuddin bin Shamsuddin. "Forensics Investigation Challenges in Cloud Computing Environments". *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012, International Conference on,* pp. 190-194. IEEE 2012.

Goldberg, Robert P., *Architectural Principles for Virtual Computer Systems,* Harvard University, Cambridge, Ma, Division of Engineering and Applied Physics, 1973

Grispos, G., Glisson, W. and Storer, T. "Calm Before the Storm: The Challenges of Cloud", Emerging Digital Forensics Applications for Crime Detection, Prevention, and Security (2011), p 4

Halfond, W. G. J., and Orso, A. "AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks." In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, pp. 174-183. ACM, 2005.

Hay, B. and Nance, K. "Forensic Examination of Volatile System Data using Virtual Introspection", *ACM SIGOPS, Operating System Review, Volume 42:3 p 74 – 82.*

Hu, W., Hiser, Hiser, J., Williams, D., Filipi, A, Davidson, J. W., Evans, D., Knight, .J C., Nguyen-Tuong, A. and Rowanhill, J. "Secure and practical defense against code-injection attacks using software dynamic translation." In Proceedings of the 2nd international conference on Virtual execution environments, pp. 2-12. ACM, 2006.

Kerr, Orin S. "Digital evidence and the new criminal procedure." *Columbia Law Review* (2005): 279-318
Mason, S., and George, E. "Digital evidence and 'cloud'computing." Computer Law & Security Review 27,

no. 5 (2011): 524-528.

Mell P., Grance, T., *The NIST Definition of Cloud Computing*, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, September, 2011

Nance, K., Bishop, M., and Hay, B. "Virtual machine introspection: Observation or interference?" Security & Privacy*, IEEE 6, no. 5 (2008): 32-37*

Nance, K., Hay, B. and Bishop, M.. "Investigating the implications of virtual machine introspection for digital forensics" *International Conference on Availability, Reliability and Security, 2009. ARES'09., pp. 1024-1029. IEEE, 2009.*

Padhy, R. P., & Patra, M. R., Managing IT Operations in a Cloud-driven Enterprise: Case Studies. American Journal of Cloud Computing, 1(1), (2013), 1-18.

Palmer, G.. "*A road map for digital forensic research*", Report from the First Digital Forensic Research Workshop (DFRWS), 2001, p. 17

Rhee, J., Riley, R., Xu, D., and Jiang, X.. "Defeating dynamic data kernel rootkit attacks via vmm-based guest-transparent monitoring." In Availability, Reliability and Security, 2009. ARES'09. International Conference on, pp. 74-81. IEEE, 2009

Rosenblum, M., Garfinkel, T, (2003), "A Virtual Machine Introspection Based Architecture for Intrusion Detection", *Proc. Network and Distributed Systems Security Symposium*

Rosenblum, M., Garfinkel, T. (2005) "Virtual machine monitors: Current technology and future trends", *Computer 38, no. 5*, p. 39-47

Ruan, Keyun, Joe Carthy, Tahar Kechadi and Mark Crosbie. "Cloud Forensics." In *Advances in digital forensics VII*, pp 35-46. Springer Berlin Heidelberg, 2011

Seshadri, A., Luk, M., Qu, N., and Perrig, A.. "SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes." In ACM SIGOPS Operating Systems Review, vol. 41, no. 6, pp. 335-350. ACM, 2007

Taylor, Mark, John Haggerty, David Gresty, and David Lamb. "Forensic investigation of cloud computing systems." *Network Security "*2011, no. 3 (2011): 4-10.

Williams, J., (2012), ACPO Good Practice Guide for Digital Evidence (online) http://library.npia.police.uk/docs/acpo/digital-evidence-2012.pdf, accessed 4th. July, 2013.

Xing, Yuping and Yongzhao Zhan. "Virtualization and Cloud Computing." in *Future Wireless Networks and Information Systems*, pp 305-317, Springer Berlin Heidelberg, 2012.

Younge, Andrew J., Robert Henschel, James T. Brown, Gregor von Laszewski, Judy Qiu, and Geoffrey C. Fox. "Analysis of virtualization technologies for high performance computing environments." In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 9-16. IEEE, 2011.