Semantic monitoring mechanisms dedicated to security monitoring in IaaS cloud

Yacine Hebbal*†, Sylvie Laniepce*, Jean-Marc Menaud†
*Orange Labs, Security Department, Caen, France.
†Ecole des Mines de Nantes, Nantes, France
{yacine.hebbal, s.laniepce}@orange.com
menaud@mines-nantes.fr

***Abstract*—Virtual Machine Introspection (VMI) is a technique that enables monitoring virtual machines at the hypervisor layer. This monitoring concept has gained recently a considerable focus in computer security research due to its complete but semantic-less visibility on virtual machines activities and isolation from them. This paper aims to introduce my PhD thesis subject related to Virtual Machine Introspection techniques.**

***Keywords*-Virtual machine introspection, Semantic gap, Security-as-a-service, Intrusion detection.**

I. Introduction

Virtual Machine Introspection (VMI) [1] is a technique of monitoring Virtual Machines (VMs) from the hypervisor level or from a privileged VM. VMI aims to inspect states and activities of the guest operating systems running inside the VMs. It can be used for intrusion detection and prevention, malware analysis, memory forensics, etc.

Monitoring virtual machines from the hypervisor level – namely VMI – was first introduced by Garfinkel and Rosenblum in 2003. Doing so combines the advantages of network-based intrusion detection systems that are highly isolated from the untrusted VM and host-based ones that have a good visibility on their activities [2]. In addition, VMI-based systems inherit the properties of the hypervisor, and hence they are:

i) Isolated from the VMs which are assumed to be unable to tamper with the hypervisor, and hence they are tamper resistant

ii) Efficient: VMI systems have a complete view on VMs activities and can access to all states of their guest OSes (CPU registers, memory, devices ...)

iii) Capable of modifying and interfering with any state and activity of any guest OS due to the interposition of the VMI system between the VM and the underlying hardware.

When performing VMI, hardware level states of VMs, such as the ones of processor, memory and devices are available to the VMI system and can be interpreted at the hypervisor level. High level information about in-VM states such as processes information, files manipulation, network connection usage etc... are also required in most of security monitoring applications. However, the hypervisor view of this information is just a series of raw binary data that have no semantics and that do not reflect guest OS activities and states. This challenge is known as the "semantic gap" [3].

In the next section, we briefly present how the different categories of existing VMI techniques address the semantic gap problem. For each category, we present its advantages and weaknesses. After that we conclude and we present the goals of this thesis in section III.

II. Bridging the semantic gap

In this section, we organize existing VMI techniques according to, from where and how the semantic information about VMs states and activities is obtained [4]. Existing VMI techniques can be classified into four main categories: in-VM (II-A), out-of-VM 'delivered' (II-B), out-of-VM 'derived' (II-C) or a combination of these approaches namely 'hybrid' (II-D).

A. In-VM

In this category of VMI techniques [5][6], the hypervisor coordinates with in-VM agents (e.g. hooks and handlers) to monitor VMs states and activities. The role of in-VM agents is to expose VM states and activities to the hypervisor which enforces desired security policies and protects the in-VM agents against tampering. Hence, in-VM VMI techniques avoid the semantic gap problem by using in-VM agents.

However, using in-VM agents may bring the weaknesses of host-based IDS such as exposition to attacks, lack of transparency and limitation of visibility to the one exposed by OS APIs.

In contrary, out-of-VM techniques – that are considered as the real VMI – perform VMI completely at the hypervisor level and bridge the semantic gap using semantic information that is delivered explicitly to the VMI system or derived automatically from the knowledge of the underlying hardware architecture.

B. Out-of-VM delivered

This category covers early VMI techniques [1][7][8] that monitor VMs activities passively and bridge the semantic gap using delivered semantic information. Delivered semantic information can be specific knowledge about guest OS internals [9] and location/definition of OS data structures of interest, OS source code or kernel symbols.

Unfortunately, delivered semantic information depends heavily on OS type and version. Hence, delivered semantic information of an open source OS will not help for a closed source one. In addition, major OS updates can make delivered semantic information obsolete. Also, kernel symbols can be corrupted or simply not available which breaks VMI systems that depend on them. Moreover, monitoring VMs passively and hence periodically gives malware a chance to carry their attacks and then hide their effects between two security scans [10]. Finally, using assumptions and delivered semantic information about guest OSes makes this category of VMI techniques vulnerable to attacks on kernel data which can make delivered assumptions and semantic information unusable [11] [12].

To address these limitations, another category of VMI techniques leverages the underlying hardware architecture to infer information about guest OSes activities whose part of them is implemented in the hardware architecture (e.g. process switching, system calls ...) namely out-of-VM derived.

C. Out-of-VM derived

Deriving semantic information about VMs activities from the knowledge of the underlying hardware architecture makes this category of VMI techniques: OS-agnostic, resistant to kernel data attacks and to malware evasion. VMI techniques of this category can be classified in their turn into two subcategories: those that proceed by handling traps to the hypervisor triggered natively by guest OS activities that perform sensitive operations [13] and those that leverage hardware-based hooks to force traps on VM activities of interest for monitoring [14].

Despite that out-of-VM derived VMI techniques are reliable and OS-agnostic, their visibility is limited to VM activities that are assisted by the hardware. In addition, triggering a trap and hence stopping the VM execution on each monitored activity causes a performance overhead that increases proportionally to the number of trapped activities.

D. Hybrid techniques

Hybrid VMI techniques combine in-VM, delivered and derived techniques in order to increase robustness, reliability and to extend the range of possible VMI applications. However, the properties of a hybrid VMI system do not combine necessarily all advantages of used techniques and can hence inherit from the limitations of some of them.

III. Discussion and Conclusion

Existing VMI techniques in the literature are limited to a single VM monitoring or to a specific OS type and they lack automation, scalability and portability. Hence, they are not mature enough to monitor real IaaS cloud environments which run a large number of VMs with a lot of different OS versions. This thesis aims to address limitations of existing VMI techniques, its two main goals are:

- propose new practical and reliable VMI techniques that are applicable to IaaS cloud and which bridge the semantic gap and reconstruct automatically information about VM states and activities independently from the OS they run,
- exploit the obtained information at the hypervisor level for VMs security monitoring.

The infrastructure is a valuable point of presence - between virtual machines to be monitored and the underlying hardware - for cloud providers to build hypervisor-based security offering isolation from the monitored virtual machines, efficiency and cost effectiveness. The expected result from the thesis is to make the cloud infrastructure smart enough to enable security cloud services at low layers.

References

[1] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *In Proc. Network and Distributed Systems Security Symposium*, pp. 191–206, 2003.

[2] S. Laniepce, M. Lacoste, M. Kassi-Lahlou, F. Bignon, K. Lazri, and A. Wailly, "Engineering intrusion prevention services for iaas clouds: The way of the hypervisor," in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pp. 25–36, IEEE, 2013.

[3] P. Chen and B. Noble, "When virtual is better than real [operating system relocation to virtual machines]," in *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, pp. 133–138, May 2001.

[4] Y. Hebbal, L. Sylvie, and J.-M. Menaud, "Virtual machine introspection: Techniques and applications," in *International Conference on Availability, Reliability and Security*, 2015.

[5] B. Payne, M. Carbone, M. Sharif, and W. Lee, "Lares: An architecture for secure active monitoring using virtualization," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 233–247, May 2008.

[6] M. I. Sharif, W. Lee, W. Cui, and A. Lanzi, "Secure in-vm monitoring using hardware virtualization," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, (New York, NY, USA), pp. 477–487, ACM, 2009.

[7] X. Jiang, X. Wang, and D. Xu, "Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, (New York, NY, USA), pp. 128–138, ACM, 2007.

[8] B. Dolan-Gavitt, T. Leek, M. Zhivich, J. Giffin, and W. Lee, "Virtuoso: Narrowing the semantic gap in virtual machine introspection," in *Proceedings of the IEEE Symposium on Security and Privacy (Oakland)*, May 2011.

[9] M. Russinovich, D. Solomon, and A. Ionescu, *Windows internals*. Pearson Education, 2012.

[10] B. Jain, M. B. Baig, D. Zhang, D. E. Porter, and R. Sion, "Sok: Introspections on trust and the semantic gap," in *Security and Privacy (SP), 2014 IEEE Symposium on*, pp. 605–620, IEEE, 2014.

[11] "Direct kernel object manipulation." http://www.blackhat.com/presentations/win-usa-04/bh-win-04-butler.pdf.

[12] 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," in *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, pp. 82–91, Oct 2010.

[13] S. T. Jones, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Antfarm: Tracking processes in a virtual machine environment," in *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, ATEC '06, (Berkeley, CA, USA), pp. 1–1, USENIX Association, 2006.

[14] C. Pham, Z. Estrada, P. Cao, Z. Kalbarczyk, and R. Iyer, "Reliability and security monitoring of virtual machines using hardware architectural invariants," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, pp. 13–24, June 2014.