Bringing Memory Forensics and Virtual Machine Introspection to Production Environments

Benjamin Taubmann, Hans P. Reiser

ABSTRACT

Virtual machine introspection is a valuable tool that can help to increase the security of systems because it provides an untampered view on the system state and can detect malicious software that hides itself. Additionally, due to the hardware virtualization layer, VMI-based monitoring solutions cannot be easily detected and attacked. In this paper we describe problems and possible solutions that are related to implement VMI in production environments. In detail, we discuss the problem of memory access, provide solutions that efficiently interpret and extract information from main memory and discuss how VMI can be adopted to certain use cases.

1 INTRODUCTION

Virtual machine introspection (VMI) is the process of analyzing the low level state (e.g., contents in memory and CPU registers) of a virtual machine and extracting high level information out of it such as the process list of the operating system. It can be considered as a special form of memory forensics with the focus on virtual machines. It has been shown, that VMI is a valuable method for IT-security, e.g., for analyzing the behavior of malware and attackers [1, 2] as well as, for intrusion detection systems [3–5]. This is mainly caused by the fact that it is much harder for an attacker to detect VMI based monitoring and to tamper the acquisition process compared to in-guest agents such as virus scanners.

One challenge for doing memory forensics in practice is to get access to the main memory of a system. This usually works in small research environments but is not available in production environments such as cloud computing or on mobile devices. The biggest challenge of memory forensics is the problem of interpreting the low level data sources, e.g., bridge the semantic gap efficiently. For off-line analysis this has already been implemented by tools such as volatility and rekall. But these tools are not fast enough for VMI applications aim to extract ephemeral data on-line from running systems without killing the performance of the monitored system. To sum up, we addresses the following research questions:

(1) How to access the memory of production systems such as cloud environments or mobile devices?
(2) How to locate and extract high level information efficiently from main memory?
(3) How to adopt VMI methods to the requirements of production environment use cases?

2 DATA ACQUISITION

The main requirement for memory forensics is to have access to the system state which is mainly spread across CPU registers, main memory and hard disk. The focus of this research is the analysis of the state which is stored in main memory and in the content of the CPU registers. We discuss the problem of memory access for two different types of systems which are commonly used nowadays: virtual machines in cloud computing and mobile devices such as smartphones.

Virtual Machines in Cloud Data Centers. Getting the contents of main memory of a virtual machine is usually easy as long as access to the hypervisor is given and when the hypervisor provides functions to access it. However, cloud computing providers do not expose this interface to their customers. Thus, customers do not have any means to perform VMI based analysis on their virtual machines. The CloudPhylactor [6] architecture discusses an approach that enables cloud customers to perform VMI operations on their virtual machines in cloud environments. Therefore, we introduce the concept of two types of virtual machines: production (PVM) and monitoring VMs (MVM) whereas the MVM can access the memory of a PVM. Thus, a user can freely choose which kind of VM he wants to start at the cloud provider and perform VMI on his VMs. With this concept we move the VMI monitoring tool from the most powerful VM (dom0 in the case of XEN) to a VM with restricted permissions. Thus, if the interpretation routine of the monitoring tool is attacked with crafted data in main memory, an attacker can gain only access to the VMs of the same user but not to VMs of all customers running on the same cloud node.

Mobile Devices. Mobile Devices contain many information that can be used for forensic investigations. We described two approaches to access the memory of mobile ARM-based devices for memory forensics and virtual machine introspection. This first approach employs a cold boot attack to access the contents in main memory of a smart-phone. By using a minimal operating system most of the data is kept unmodified and available for analysis [7]. Additionally, we discussed whether processor extensions of mobile devices such as the ARM TrustZone can be used to monitor the system state of the normal operating system in order to improve the trustworthiness of the data acquisition [8]. Therefore, we aimed to implement a library in the TrustZone that has a similar interface as libvmi so that already implemented VMI tools can be ported to the TrustZone.

3 INFORMATION EXTRACTION

The key aspect for on-line memory forensics and VMI is the performance of the information extraction routine. We have discussed two approaches that for finding the ephemeral cryptographic key material from TLS connections in the memory of applications.

Brute Force. In the first case, we extract the key material from the address space of a user space program running in a virtual machine whenever we detect in the network traffic that it negotiates a new TLS connection. To extract the key we implemented a brute force approach that scans the memory of an application and tries to decrypt the first TLS message with all byte sequences in main memory [9]. The runtime of this approach depends on the size of the address space and is very resource intensive. Even by applying several optimizations that improve the performance, this approach is infeasible for applications with many concurrent TLS connections since taking a snapshot is expensive as well as the brute force approach.

Recomputing data structures. In the second case we use a different approach to improve the key extraction process [10] We instrument the control flow of Android applications to extract the key material whenever it calls functions of the TLS library that access the key material. We have tested this approach on a smartphone where the exact layout of the data structure holding the information was not known. The challenge of this approach is to find the exact position in the data structures that hold the TLS key. To reconstruct the data structure layout, we implemented an algorithm that bridges the semantic gap and partially reconstructs the data structure layout of the running TLS library. To bootstrap this approach, we use the brute force method to locate the TLS key in the address space of the application. Then, we compute a path from the parameters of functions (pointers on the stack) accessing the data structure to the exact position of the key in it. Finally, we use the computed path to extract the TLS key at run time of the application whenever the control flow is intercepted. We have shown that our proposed approach works and is feasible to extract the TLS sessions from Android applications and improves the performance of data extraction.

4 APPLICATIONS

VMI is a very useful method but fully tracing a system is often not feasible since the performance impact is big. Nevertheless, we found applications where VMI can be adopted to meet the performance requirements.

Honeypots. The performance impact of tracing is not the most important aspect of honeypots as they do not have any production value. However, VMI-based stealthy tracing allows to catch new attacks which might not occur when an attacker detects that he is monitored [2].

Malware Analysis. VMI is a valuable technique for malware analysis since applications are not able to directly detect any in-guest agent and thus do not change their behavior and hide malicious activity. We used VMI to trace the executed system calls of malware and stored them in a database [1]. This data might help to build efficient VMI-based virus scanners in the future by tracing only relevant data.

Intrusion Detection System. IDS can benefit from the untampered view on the system state provided by VMI. However, it is important that the monitoring does not affect the performance of the production VMs. Therefore, we discussed the approach of performing light-weight VMI tracing to detect attacks and heavy weight tracing in the case of an incident [4, 5]. Leight-weight is for example regularly extracting the process list or scanning for known malware signatures. A Heavy-weight mechanism is for example the tracing of all system and library/function calls. By selecting between these modes we can adopt VMI to each specific use case.

5 CONCLUSION

We presented practical solutions for problems in the fields of virtual machine introspection and memory forensics in production environments. We discussed how to acquire data, bridge the semantic gap in order to efficiently extract information and finally showed practical use cases.

REFERENCES

[1] B. Taubmann, B. Kolosnjaji, Architecture for resource-aware vmi-based cloud malware analysis, in: SHCIS'17.

[2] B. Taubmann, S. Sentanoe, H. P. Reiser, Virtual machine introspection based ssh honeypot, in: SHCIS'17.

[3] B. Jain, M. B. Baig, D. Zhang, D. E. Porter, R. Sion, Sok: Introspections on trust and the semantic gap, in: IEEE Symposium on Security and Privacy (SP).

[4] A. Fischer, T. Kittel, B. Kolosnjaji, T. K. Lengyel, W. Mandarawi, H. P. Reiser, B. Taubmann, E. Weishäupl, H. de Meer, T. Müller, M. Protsenko, CloudIDEA: a malware defense architecture for cloud data centers, in: C&TC 2015.

[5] F. Menges, F. Bhm, M. Vielberth, A. Puchta, B. Taubmann, N. Rakotondravony, T. Latzo, Introducing DINGfest: An architecture for next generation SIEM systems, Short Paper, GI Sicherheit 2018 (2018).

[6] B. Taubmann, N. Rakotondravony, H. P. Reiser, CloudPhylactor: harnessing mandatory access control for virtual machine introspection in cloud data centers, in: IEEE TrustCom-16.

[7] B. Taubmann, M. Huber, L. Heim, G. Sigl, H. P. Reiser, A lightweight framework for cold boot based forensics on mobile devices, in: ARES 2015.

[8] M. Guerra, M. Correia, B. Taubmann, H. P. Reiser, ITZ: an introspection library for ARM TrustZone, in: Proceedings of INFORUM 2017, INFORUM, 2017.

[9] B. Taubmann, C. Frädrich, D. Dusold, H. P. Reiser, TLSkex: harnessing virtual machine introspection for decrypting TLS communication, in: DFRWS EU 2016.

[10] B. Taubmann, O. A. Abduljaleel, H. P. Reiser, Droidkex: Fast extraction of ephemeral tls keys from the memory of android apps, in: DFRWS USA.