Memory Forensics Using Virtual Machine Introspection for Malware Analysis

Chin-Wei Tien[1,2] Jian-Wei Liao[1] Shun-Chieh Chang[1] Sy-Yen Kuo[2]

[1]CyberTrust Technology Institute
Institute for Information Industry
Taipei, Taiwan R.O.C.
{jakarence, jianweiliao, scchang}@iii.org.tw

[2]Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan R.O.C.
{d99921020, sykuo}@ntu.edu.tw

***Abstract*—A security sandbox is a technology that is often used to detect advanced malware. However, current sandboxes are highly dependent on VM hypervisor types and versions. Thus, in this paper, we introduce a new sandbox design, using memory forensics techniques, to provide an agentless sandbox solution that is independent of the VM hypervisor. In particular, we leverage the VM introspection method to monitor malware running memory data outside the VM and analyze its system behaviors, such as process, file, registry, and network activities. We evaluate the feasibility of this method using 20 advanced and 8 script-based malware samples. We furthermore demonstrate how to analyze malware behavior from memory and verify the results with three different sandbox types. The results show that we can analyze suspicious malware activities, which is also helpful for cyber security defense.**

Keywords—cyber security; security sandbox; virtual machine introspection; advanced malware analysis

I. INTRODUCTION

Cyber security has become a significant issue in our daily lives, as an increasing number of viruses, ransomwares, zero-days and malwares attempt to enter systems and compromise business processes to gain certain benefits. According to the Symantec Internet Threat Report 2016 [7], more than 430 million new malwares were discovered in 2015, up 36% from the year before. Analysts often use a security sandbox to monitor and analyze the system behaviors of advanced malware in order to determine defense strategies. However, because current sandbox solutions must capture all system call behaviors, they must modify the event monitoring routines of VMExit in hypervisor kernel and it is highly dependent on virtualization hypervisors, e.g. Xen [1], QEMU [5], VMware [2], and VirtualBox [3]. In order to improve Hypervisor dependency, we propose a novel sandbox architecture using memory forensic techniques, which does not require the capturing of system calls. By applying the advantages of virtual machine (VM) introspection, we can watch the live memory data of VMs inside the hypervisor kernel, and analyze system behavior using memory forensics. Memory forensic techniques have better monitoring capability for rootkit behaviors such like DLL injection, API hooking and Hidden processes. Besides, the proposed method do not need to change the Hypervisor source code, thus, it has better compatibility and scalability. Finally, our proposed method is an agentless solution that supports many VM hypervisor types.

In summary, our major contributions are as follows:

- We design a novel security sandbox system, which leverages VM introspection and memory forensic techniques. The system can scan malware live memory data and analyze system behaviors, such as process, file, registry, and network activities.
- We use 28 advanced malwares to demonstrate the analysis of suspicious behavior from live memory data, and verify the result with three different sandbox types. All of the testing malware samples are publicly shared [4].

The remainder of this paper is organized as follows. In section II, we present the implementation and design of the novel sandbox using memory forensics. The experimental results are provided in section III, and finally, we discuss our conclusions in section IV.

II. SYSTEM DESIGN

In this study, we detect malicious behavior by using the Volatility framework [8], Xen hypervisor [9], and LibVMI [6]. Fig. 1 shows the schematic of our detection system. The Xen hypervisor provides services that allow multiple computer operating systems to execute on the same hardware concurrently. LibVMI is a library that can monitor a running VM's low-level details by viewing its memory, trapping hardware events, and accessing the vCPU registers. A malware sample is launched in the VM and malicious behavior is retained in its memory. Then, we can integrate our system with the Volatility framework and LibVMI to detect the VM's memory for high-level analysis.

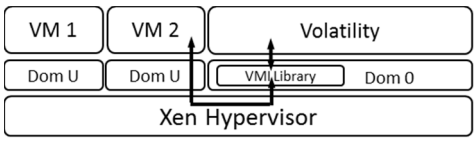

Fig. 1. Using virtual machine introspection and memory forensics technologies to analyze system activities from live memory data in guest VM.

III. EXPERIMENT

In order to determine the behavior of the malware sample effectively, we compared the guest VM memory state before and after the malware execution. This revealed vital information

978-1-5090-5569-2/17/$31.00 ©2017 IEEE

about the sample malware, such as process flow, file, registry, and network behavior, among others. By analyzing files, the registry, and the network operating behavior, we could identify suspicious activities by means of cross-checking with a behavior blacklist [10]. For example, we observed that during the malware sample execution, certain executable files were created and written to \$USER_Local\ Application Data\Temp path, where malware executable files are often placed. Therefore, this type of file operation can aid in identifying threats. Fig. 2 displays the detailed steps mentioned above.

We prepared two malware sample collections for experiments 1 and 2. In experiment 1, we selected 20 advanced malware samples, which included Microsoft Office documents (e.g., .XLS(x), .DOC(x), .PPT(x), and .RTF), portable document format files (i.e., .PDF), and executable files (e.g., .EXE, .SCR). We then used Xen to create the guest VM, which was installed on the Microsoft Windows 7 operating system, with Microsoft Office 2007 and Adobe Reader 8.

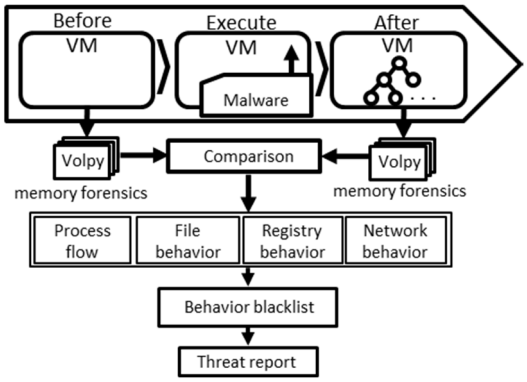

Fig. 1. Suspicious behavior analysis process using memory forensics.

Next, we verified the log files of our method with those of the hypervisor-based sandbox CIA [1], the result of which is shown in Table 1, row 1. The network detection coverage is 100%, because we use the same tool as CIA to detect network activities. This experiment focused on the accuracy rate, verified with CIA, and included process, file, and registry. The highest accuracy rate was that of file, with 95%. Although the accuracy rates of process and registry are not ideal, we needed only to detect one malicious behavior, which was sufficient to consider the sample as malware. Therefore, the accuracy rate of experiment 1 was still as high as 90% (see Table 2, row 1), meaning that only two malwares were false negatives in our method.

TABLE I. VERIFICATION OF ANALYSIS RESULTS

Verification of Analysis Results[a]	Process	File	Registry	Network
Verified with CIA	53%	95%	65%	100%
Verified with Cuckoo Sandbox	67%	29%	57%	14%
Verified with CaptureBAT	94%	75%	50%	13%

[a]Verification result: [(Our method's result and target's result take the intersection/target's result)$_{sample\_1}$ + … +(Our method's result and target's result take the intersection/target's result)$_{sample\_n}$] / n * 100%.

For experiment 2, we selected 8 script-based malware (WSF), and HTML Application (HTA) file types. The guest VM for this experiment was installed on the Microsoft Windows XP operating system, with Microsoft Office 2003 and Adobe Reader 11. The experimental method is shown in Fig. 2, and the experimental tool was Volatility only.

In experiment 2, we verified the log files of our method with those of Cuckoo Sandbox [3] and CaptureBAT [2]. Table 1, row 2 shows the results of verification using Cuckoo Sandbox. Although the amount of certain information collected was not as high as that of Cuckoo Sandbox, pivotal malicious behaviors, such as those relating to connection, were detected. Table 1, row 3 shows the results of verification using CaptureBAT. Obviously, the experimental value is superior to that of Cuckoo Sandbox, and we believe that the detection effect of our method is rather similar to that of CaptureBAT, which equips with global hook technology. For script-based malware, the network accuracy rate was not ideal; thus, it would be preferable to use another network detection tool to replace memory forensics. Finally, the accuracy rate of experiment 2 was 75% (see Table 2, row 2).

TABLE II. DETECTION RATE OF OUR METHOD

Sample type	Detection rate[b]
Experiment 1	90%
Experiment 2	75%

[b]Detection rate: rate of malicious detection.

IV. CONCLUSION

Using memory forensics is an efficient approach to detect malware, with the detection rate of Microsoft Office documents, portable document format files, and executable files being up to 90%. However, the detection rate of script files is only 75%, because process and network activities quickly vanish in volatile memory. We suggest solving this problem by using memory forensics timing, frequency adjustment, and other heuristic methods. Finally, our method overcomes the problem of high dependency on VM hypervisor types, as it is an agentless solution that supports many VM hypervisor types.

REFERENCES

[1] C. H. Lin, C. W. Tien, C. W. Chen, C. W. Tien, and H. K. Pao, "Efficient spear-phishing threat detection using hypervisor monitor," 2015 International Carnahan Conference on Security Technology (ICCST), pp. 299-303, 2015.

[2] C. Seifert, R. Steenson, I. Welch, P. Komisarczuk, and B. Endicott-Popovsky, "Capture – A behavioral analysis tool for applications and documents," Digital Investigation, vol. 4, pp. 23-30, 2007.

[3] Cuckoo Sandbox, [online] Available: https://cuckoosandbox.org/.

[4] Testing malware samples, [online] Available: https://goo.gl/halC6y

[5] Lastline, [online] Available: https://www.lastline.com/.

[6] LibVMI, [online] Available: http://libvmi.com/.

[7] Symantec, "Symantec Internet Threat Report 2016," [online] Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf, 2016.

[8] Volatility, [online] Available: http://www.volatilityfoundation.org/.

[9] Xen, [online] Available: https://www.xenproject.org/.

[10] SANS, "SANS Digital Forensics and Incident Response - Find Evil," [online] Available: goo.gl/H1DgbQ/.