International Journal of Security and Its Applications
Vol.9, No.1 (2015), pp.369-376
http://dx.doi.org/10.14257/ijsia.2015.9.1.35

Efficient Methods to Trigger Adversarial Behaviors from Malware during Virtual Execution in SandBox

Jung-Uk Joo[1], Incheol Shin[2], Minsoo Kim[2*]

[1]*Interdisciplinary Program of Information & Protection, Mokpo National University*
[2]*Dept. of Information Security Engineering, Mokpo National University*
[1]*jju1012@mokpo.ac.kr*, [2]*{ishin, phoenix}@mokpo.ac.kr*

Abstract

Recently, various organizations are confronting a grater attack surface, the growing proliferation of malware and the number of malicious codes has been consistently growing for several years. To respond actively against these malicious codes, analysts employ automated investigation tools on the malware. However, there has been advent of malware employing the various techniques to avoid the detection of the SandBox, which makes hard to identify the adversarial behaviors of the samples codes. In this paper, we propose efficient methods to trigger adversarial behaviors from the sample codes during virtual execution in the Sandbox in order to perform the analysis of malware.

Keywords: *malicious code, sandbox, anti-debugging, malicious behavior, anti-vm*

1. Introduction

There has been limitation on the existing signature based malware detection techniques against mutants of the same threat, polymorphic variations. Current malware are equipped with high intelligence to infect to legitimate systems and attack to target systems, and various tools to generate polymorphic variants of the same threats boost the increasing rate of malware in automated generation manners [1]. In order to respond the evolvement of malware, the automated software analysis tools have been developed to investigate and identify their programmatic behavior of the massive volume of the malicious codes without any involvement of manual efforts. As of these, there has been development on the various analyzing tools for the malware investigation. For instance, Sandbox based automated malware analysis tools analysis and identify the malicious behaviors by comparison between a pre-state of the system image before execution of the samples and its post-state after its execution.

Recently, a growing number of malware is equipped with several detour techniques against Sandbox detection mechanisms, which can nullify the defense system with ease. Malware detecting the execution in the virtual testing environments would disable its operation in the duration of the virtual execution using API, RegOpenKeyExA(). The API investigates the unique virtual machine services like vmicheatbeat, vmdebug, vmmouse, vmscis, VMTools, vmware, vmx86, vmhgfs or vmxnet, and GetFileAttributeA() also looks for mouse drivers that are uniquely associated with virtual machines.

In accordance with [2], there are three types of the avoidance techniques against the Sandbox detection approaches: 1) malicious activities triggered by only specific user inputs (Malware with this stealth function activate the malicious behaviors by various user inputs, such as mouse clicks or movements.) 2) malicious activities triggered by time constraints (Malware with this stealth technique, also called time-bomb malware, operates in time scheduling manner.) 3) malicious activities disabled by detecting virtual execution (Malware detecting the execution in the virtual testing environments would disable its

* Corresponding author

ISSN: 1738-9976 IJSIA

operation in the duration of the virtual execution.) The intelligent evolvement of malware has intervened the detection to the virus identifying systems.

In this study, we propose efficient methods to trigger the abnormal or adversarial behaviors from the suspicious code samples with the possible avoidance techniques for the Sandbox approaches. In order to generate the suspicious activities, we facilitate the fuzzing techniques against the malicious samples.

2. Related Works

2.1. Malware Analysis Techniques

The process of analyzing a given program during execution is called dynamic analysis, while static analysis refers to all techniques that analyze a program by inspecting it.

Investigating software without executing it, is called static analysis using debugging tools, which can be applied on different form of a program [3]. The investigators perform function-oriented analysis through assembly codes derived from the binary executable files. It is even possible to identify the function routines, which are not to be executed during the operation since it reveals not only the every corner of the codes but also the existence of packers or polymorphic mutants. However, it might take too longer time to analyze the codes depending on their size and impossible to looking into the codes if they are encrypted, packed or compressed.

Analyzing the actions performed by a program while it is being executed is called dynamic analysis [4]. During the operation of processes from the testing software, the monitoring program intercept function calls by hooking regarding access of libraries, system files, registries, various files, network sockets, memories, account information and so on and make logs on them. Analytics employ sandbox based analysis tools for dynamic analysis. One of the advantageous features of dynamic analysis tools is enabling to identify the programmatic behaviors of the software even they are packed, encrypted or compressed because it traces the access actions to the system resources.

2.2. Malware Analysis Techniques based on Virtual Machines

In order to analyze the massive volume of malicious codes, there have been various automated reporting tools for the malware investigation, such as emulation based analysis methods and virtual machine based detection approaches.

Emulation approaches indirectly execute software programs on the virtual hardware through an previously generated emulator image files based on real systems [5]. The malware detection techniques based on the emulation approaches perform the inspection of the codes, executable files, through the virtual CPUs and memories by a scanning engine in a given system, and generate the reporting results without any harmful impacts on the systems.

Sandbox is a kind of virtual machine supporting independent execution environments without affecting any parts of a system where sample software runs on. Recently, there have been developed several security services for unverified codes or suspicious programs to be inspected and reported through the dynamic manner by providing online testing environment, Sandbox. The existing Sandbox security tools commonly to obtain a valuable source of information monitor and record the API invocation, process manipulation, registry accesses and network socket tracking to produce expressive report of the relative activities.

A sandbox, one of the dynamic analysis tools to assist the risk assessment, is a security mechanism to provide a restrictly controlled set of resources for untested code or untrusted program samples to run in by using scratch space on disk, memory and constrained network accesses. In addition, as a system with the virtual machine based approach would be able to provide not only separate but also independent program

running environments to individual samples, it is easy to recover the damage or impact by the malicious activities by the samples. Obviously, this would help in drastically reducing the time to analyze the programmatic behaviors even from a massive number of samples.

For instance, CWSandbox using the well-known design of virtual machine approaches has been developed by the structure described by Figure 2 [6]. It operates the executable files through their threads through the injection of the CWMonitor.dll file to monitor the programmatic behaviors. After the injection, it tracks their all the procedures or APIs and report to the CWSandbox main execution module.

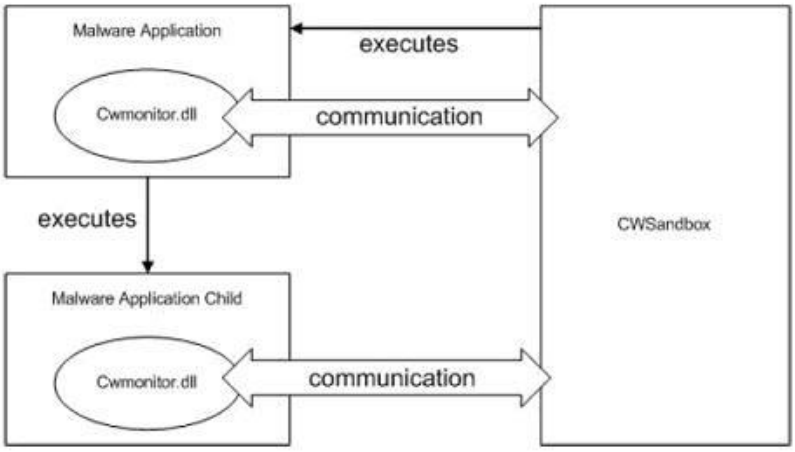

Figure 1. The Structure of CW Sandbox

Furthermore, all the Windows APIs that are called during the virtual execution would be delivered to the CWSandbox through API hooking techniques. The CWSandbox hand over the tossed Windows APIs to the Windows kernel levels as a normal procedure and the called APIs are delivered through CWSand as well. However, the stealth technique would prevent the proper detection procedures and the malicious activity identification performances of the Sandbox would be dropped worse against malware.

2.3. Malware Analysis Tools

In order to analyze the massive volume of malicious codes, there have been various reporting tools for the malware investigation. In accordance with [6], sandbox based automated malware analysis tools analysis and identify the malicious behaviors by comparison between a pre-state of the system image before execution of the samples and its post-state after its execution. We introduce the features of those tools.

1) ZeroWine

ZeroWine is an analysis tool for Windows malware operating under Debian OS(Operating System) as a QEMU (Quick EMUlator) virtual machine image. Analysis results is to be reported after the upload of suspicious files to ZeroWine. It is able to provide not only the static analysis on PE (Portable Executables) by scanning headers and character strings in the files but also the dynamic analysis tracking API calling sequence on target PE file. In addition, it is equipped with detection of anti-debuggin and anti-vm and provides functions of data dump in the memory. The analytic results report the API calling sequence, character strings, signature, and modification of file/registry information [7].

2) Anubis

It is a TTAnalyze based malware analysis system developed by UC Santa Barbara research team. Sample files are executed in Windows XP under QEMU to monitor the Windows API and system service calls with recording their parameters. It is advantgeous to track/analysis CPU instructions due to the fact that it utilize a CPU emulation technique, but it is slow. The analytic results report the information regarding DLL access, packing status, registry access, file access, process generation and network access [8].

3) Cuckoo Sandbox

Cuckoo Sandbox is an open source project of an automated malware analysis tool consisting of servers and client associated by virtual network interfaces in a virtual machine. The sample code would be tested in an available virtual client picked by the server, and the programmic behaviors extracted from the execution are reported. As this is an open project, appropriate modification would be possible for various applicable conditions. The analytic results include the information of API calling sequences, file/process/network access and screen captures of abnormal behaviors during the execution [9].

3. Techniques Preventing Malware Detection

3.1. Limitations on Malware Analysis

Current malware employ the packing or obfuscation anti-virtualization techniques to evade the detection of anti-virus scanners. The packing technique wraps packers over binary executable files, and the obfuscation technique automatically conceals specific trigger-based behavior from these malware analyzers. When a virtualized system is detected, the malware terminate by anti-virtualization techniques. There have been rootkit or system manipulation techniques to nullify the malware detection systems.

In order to overcome the limitations of the static approach, investigating the programmatic actions conducted by software by their executing has been intensively studied, so-called dynamic analysis. Semantically richer representation in the implementation details for a specific task of a program can abstract its sequence of function calls by recording their invocation during the execution. This method to obtain an overview of the behavior of the programs requires a process of intercepting function calls, hooking, and its target is API, system calls, Windows native APIs and so on [10].

There have been improvement on detection avoidance techniques to the malware, anti-virtual machine technique, time-delay technique and user input recognition technique, against the Sandbox approach.

First of all, in the case of anti-virtual machine, malware detecting the execution in the virtual testing environments would disable its operation in the duration of the virtual execution using API, RegOpenKeyExA(). The API investigates the unique virtual machine services like vmicheatbeat, vmdebug, vmmouse, vmscis, VMTools, vmware, vmx86, vmhgfs or vmxnet, and GetFileAttributeA() also looks for mouse drivers that are uniquely associated with virtual machines. Second of all, malware with the time delay techniques, also called time-bomb malware, operates in time scheduling manner. We cannot analyze the malware by execution in limitless time frame because of system resource constraints. That is, it is hard to detect the malware if it does not operate in the duration of the virtual testing. Trojan Nap identified at February 2013 exploiting the API of sleep() to hide its existence of malicious behavior for a certain period of time. This causes confusion to the anti-virus software with an obvious reason that any legitimate programs could use it. At last, Malware with this stealth function activate the malicious behaviors by various user inputs, such as mouse clicks or movements.

3.2. Detouring Techniques against Sandbox Malware Detection

In this work, we design a countermeasurable detection techniques of the Sandbox against the malware equipped with the input-recognition techniques among the avoidance techniques. Due to the fact that the suspicious sample codes would be run in the Sandbox without any manual inputs, attackers take advantage of this to create the malware avoiding the Sandbox detection by hibernating the malicious activities during the virtual execution.

3.2.1. Malicious Activities Triggered by Mouse Clicks: As described in Figure 3, UpClicker, a type of Trojan horse identified December 2012, triggers its malicious activities, communication through malicious C&C servers, by detection of the mouse click. UpClicker monitors lower level mouse inputs through the WH_MOUSE_LL connection procedure after the calling an API, SetWindowsHookExA. Then, calling an API, UnhookWindowhookEx, after sensing the mouse inputs, has terminated the monitoring and it triggers the malicious activities. This type of malware is legitimate software until detecting specific inputs on it, which makes it hard to be identified by the Sandbox based analysis tools [11].

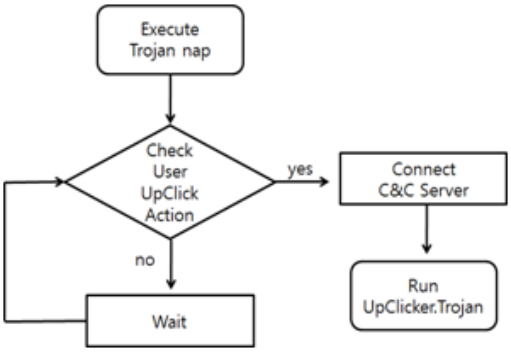

Figure 3. Activation of UpClicker.Trojan by Mouse Clicks

After the 6 month from the identification of this code, BaneChant, activating its operation by 3 mouse clicks, has showed up in the Internet [12].

3.2.2. Activating Abnormal Behavior by Windows Message Box: There has been another way to trigger the abnormal behaviors by creating message boxes to collect the inputs from users. This technique can be conducted through creating message boxes using Windows MessageBox or MessageBoxEx APIs to the EXE or DLL files, and activate the abnormal behavior as the buttons on the box have been clicked [2].

4. Methods to Trigger the Malicious Activities

As previously described, it is important to design efficient methods to derive the malicious activities in order to perform the proper analysis of the sample codes for the malware detection. The reporting results from the Sandbox indicating the programmatic events from the sample codes would be useful information to decide whether the codes are malware or not. We propose efficient methods to trigger abnormal behaviors from malware during virtual execution in SandBox.

4.1. Fuzzing based Triggering Technique

One way to trigger the malicious behaviors of the samples is using fuzzing techniques in order to provide the necessary execution environments of the codes. The fuzzing techniques generate the random or abnormal input values and execute the samples based

on these values in order to verify their soundness [13]. If there is any system disruption or crash by those values, then system administrators would be able to augment the system by patching or updating as further action. There is various ways to perform fuzzing depending on the modification on input types, such as keyboard inputs, file inputs/outputs, network packets and so on, which can derive various responses from the samples.

The fuzzing technique would be able to detect the unexpected actions or malfunctions from software programs and help in augmenting the systems, which can be facilitated in identifying the activation of malicious behaviors in the codes in order to provide the accurate analysis of the samples as well. Especially, the Concolic Testing technique would search and execute the routines of programs that have not been utilized much during the execution[14]. This can also be one of the good alternative ways trigger the malicious behavioral routines in the suspicious sample programs, which are not been executed in the normal executions.

4.2. User Action Events based Triggering Technique

In accordance of Figure 4, we propose an efficient method, User Action Event Generator, to trigger the malicious events in the Sandbox approach. The User Action Event Generator provides the sample code various forms of user inputs randomly in the Sandbox. As described, the malware with the user input recognition technique would be able to avoid the Sandbox detection with ease and the countermeasures against this method can be the one that should employ the functions to generate the user inputs in order to remove the hibernation factors. Our User Action Event Generator would be one of the solutions for this.

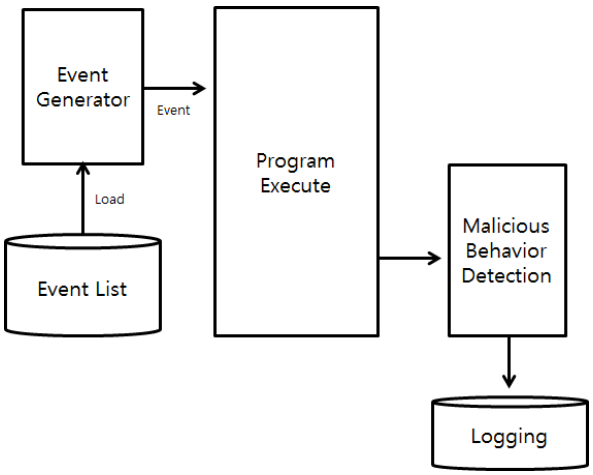

Figure 4. User Action Event Generator

In order to derive the malicious activities hidden in the samples, the Event Generator provide the events from the Event List to the Program Execution modules randomly, and the list is constructed by the fuzzing methods. That is, the fuzzing modules create the events and setup the list, and the inputs for the activation would be generated based on the list with various parameters randomly. In our model, the event list includes the mouse click and scroll , button events and so on, and the parameters also created by random operations for fuzzing processes on the samples.

However, it takes too much time for generation of event lists and parameters in random ways, which could be a problem to apply our approach in practical Sandbox tools. In order to reduce the workloads on this procedure, we need to minimize their size for its practical design. The Concolic Testing algorithm would be a good candidate for the solution on this matter, which means that it is relatively easy to choose necessary event

lists for the testing. This module would derive the malicious activities from the samples in order to achieve the accurate analysis on the codes in order to determine the labels on them.

5. Conclusion

In this paper, we propose an efficient method to trigger the malicious activities from the suspicious samples equipped with user input recognition technique against the virtual execution of Sandbox approaches. The user input recognition technique helps in conceal their abnormal programmatic behaviors during the virtual executions, but our User Action Event Generator can provide the necessary conditions the malware to activate themselves by the fuzzing based technique. Consequently, this model would be able to help in triggering malicious behaviors from the suspicious samples to improve the accuracy of the malware identification performance of the Sandbox.

Acknowledgements

This work was supported by ICT R&D program of MSIP/IITP. [14-824-06-001: The Development of Cyber Blackbox and Integrated Security Analysis Technology for Proactive and Reactive Cyber Incident Response].

References

[1] H. Lu, X. Wang and J. Su, "CS: Collaborative Malware Clustering and Signature Generation using Malware Behavioral Analysis.", IJHIT, vol. 5, no. 2, (**2012**) April, pp. 147-152.
[2] Fireeye, Hot Knives through Butter: Evading File-based Sandboxes", (**2013**) August.
[3] M. Christodorescu and S. Jha, "Static analysis of executables to detect malicious patterns", 12th Usenix Security Symposium, (**2003**), pp. 169-186.
[4] F. Bellard, "QEMU - a fast and portable dynamic translator", Proceedings of the annual conference on USENIX Annual Technical Conference, (**2005**) April 10-15, Anaheim, CA, pp. 41-46.
[5] P. Xie, X. Lu, Y. Wang and J. Su, " Eliminate Evading Analysis Tricks in Malware using Dynamic Slicing. IJSIA, vol. 7, no. 3, (**2013**), May, pp. 357-366.
[6] C. Willems, T. Holz and F. Freiling, "Toward automated dynamic malware analysis using cwsandbox", IEEE Security and Privacy, vol. 5, (**2007**), pp. 32-39.
[7] ZeroWine, http://zerowine.sourceforge.net/
[8] Anubis, https://anubis.iseclab.org/
[9] Cuckoo Sandbox, http://www.cuckoosandbox.org/
[10] A. A. E. Elhadi, M. A. Maarof and B. I. A. Barry, "Improving the Detection of Malware Behaviour Using Simplified Data Dependent API Call Graph", IJSIA, vol. 7, no. 5, (**2013**) September, pp. 29-42.
[11] Fireeye, Don't Click the Left Mouse Button: Introducing Trojan UpClicker, (**2012**) December.
[12] BaneChant, Trojan.APT.BaneChant: In-Memory Trojan That Observes for Multiple Mouse Clicks, (**2013**) April.
[13] P. Godefroid, M. Y. Levin and D. Molnar, "Automated Whitebox Fuzz Testing", NDSS, vol. 8, (**2008**).
[14] C. Pasareanu and W. Visser, "A survey of new trends in symbolic execution for software testing and analysis", ICSE (**2009**).

Authors

Jung-Uk Joo, he received the BE degree in Dept. of Information Security Engineering from the Mokpo National University, Korea, in 2013. He is a graduate student in the Master of Interdisciplinary Program of Information & Protection, Mokpo National University, Korea. His research interests include malicious code analysis and web security.

Incheol Shin, he received the BE degree in computer science and engineering from the Hansung University, Seoul, Korea, in 2002. He graduated with the PhD degree at the Department of Computer and Information Science and Engineering, University of Florida, under the supervision of Dr. My T. Thai. He is an assistant professor at Information Security Department of Mokpo National University at Korea. His research interests include network security and group testing.

Minsoo Kim, he graduated with the Bachelor's degree at the Department of Computer and Statistics, Chonnam National University, Korea, in 1993. He received Master's degree and PhD degree respectively from the same university in 1995 and 2000. His PhD thesis topic was related to the intrusion detection system. He is an associate professor in Department of Information Security Engineering of Mokpo National University at Korea. His research interests include malware analysis and computer forensics.