International Journal of Security and Its Applications
Vol.9, No.10 (2015), pp.165-176
http://dx.doi.org/10.14257/ijsia.2015.9.10.15

The User Action Event Generator Design for Leading Malicious Behaviors from Malware in Sandbox

Jung-Uk Joo[*], Incheol Shin[*], Tong-Wook Hwang[**] and Minsoo Kim[*1]

[*]*Mokpo National University, Republic of Korea*
jjuzz1012@gmail.com, {ishin, phoenix}@mokpo.ac.kr
[**]*Korea Information Security Agency, Republic of Korea,*
twhwang@kisa.or.kr

Abstract

The number of malwares has been consistently growing for several years and the working platform of them was diversified. To analyze these malwares, an analyst uses automated investigation tools as sandbox. However, current malwares apply the various techniques to avoid the detection of the sandbox. Especially, it is hard to be analyzed when the malicious behavior is triggered by user events. In this paper, we propose methods to enter malicious behavior routine in the sample malware codes, which is happened during the virtual execution in the sandbox in order to perform the analysis of malware. We design the methods as the user action event generator using fuzzing. The malicious behaviors triggered by the generator are exported to the sandbox report as API list. We show the result of the event generator.

Keywords: *sandbox, malicious behavior, anti-analysis, anti-VM, fuzzing*

1. Introduction

This paper proposes an improvement on the existing automated analysis tools against malware with an efficient investigation structure for the augment. 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. 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. Figure 1 shows that the volume of malware has rapidly grown for a last decade [1]. In addition, the damages from the malicious codes are varied: leaking information, launching DDoS (Distributed Denial-of-Service) attack, downloading certain malicious software, destructing physical systems and so on [2].

In addition, the threats have been equipped with the special technique not only to detour the malware detection systems but also to disturb the analyzing tools delaying the detection. Those techniques intervene the proper generation of analytic results on the sample software, which leads to the new development of automated analysis tools against malware using virtual execution environment. CWSandbox[3], ZeroWine[4], Anubis[5], and Cuckoo Sandbox[6], that executes the sample software under analysis either natively or in a virtual Windows environment, which is implemented by hook functions monitoring on the API level.

[1] Correspondent author

ISSN: 1738-9976 IJSIA

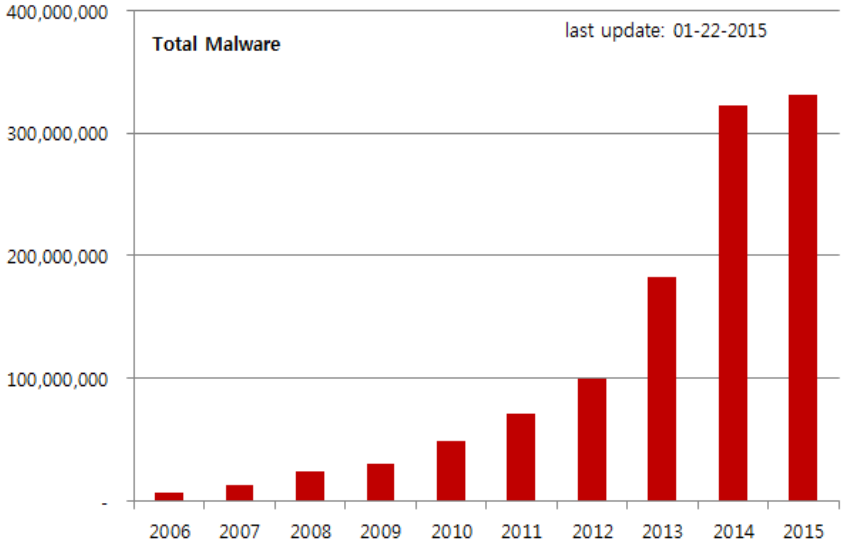

Figure 1. The Number of Malware for a Last Decade [1]

The automated malware analysis tools provide several useful information regarding the testing software but would not be able to detect the various types of malware[7]. That is, the tools help analytics in extracting specific features to identify the malware. However, there is a limitation on the automated analysis techniques against mutants of the same threats, so that this paper studies how they work and what the results are from the tools with the countermeasures against polymorphic variations. Based on this investigation, we propose improvement on the structure of existing automated analyzing tools based on sandbox.

The most important purpose of sandbox-based tools is to record various information performed while analyzing malware. Our research in this paper is focusing on a method to analyze malicious codes which have the techniques to avoid from sandbox-based analyzing. They can see if it is working on virtual machine, delay performing malicious behavior, and perform malicious behavior routine with user action. This paper suggest a sandbox structure to oppose these three methods. Also we suggest UAEG (User Action Event Generator) to counter the third method.

We design UAEG for leading malicious behaviors from malware in sandbox. User action is generally not occurred in sandbox, so malicious behaviors triggered by user action don't handle in sandbox. The user action event generator performs the function that generates user actions in sandbox randomly. We apply fuzzing to the event generator for reducing event list and parameter value of user events. We implement the user action event generator and test it.

This paper is organized as follows. Section II briefly describes various types of malware. Section III is a review of current problems derived from existing Sandbox-based malware analyzing tools. Section IV suggests improvement to accomplish higher reliability on the analytic results from the tools. The conclusion of this work is placed in Section V.

2. Malware Analyzing Techniques

2.1. Analysis Method of Malware

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 [8, 9].

2.1.1. Static Analysis: Investigating software without executing it, is called static analysis using debugging tools, which can be applied on different form of a program. 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 far longer time to analyze the codes depending on their size and impossible to looking into the codes if they are encrypted, packed or compressed.

2.1.2. Dynamic Analysis: Analyzing the actions performed by a program while it is being executed is called dynamic analysis. 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.

2.1.3. Behavior-based Analysis: Generally, the mutants from the same malware would share the single common behavior even if they have various and different flow of codes, which means that they have individually unique signatures based on the codes. Even though a signature is properly extracted through static or dynamic analysis, it would not be able to identify other mutant derived from the same malware. This is the reason why the behavior-based analysis has been continuously developed to overcome the shortcoming.

There have been wide range of researches which have been devoted to improve the behavior-based detection technique, but it has several problems to apply it in practice. Park in [10] made a comparison results between static analysis and dynamic analysis to generate API behavioral pattern, but the drawbacks from static analysis approach has not been resolved. A Method to calculate the similarity after the converting API behaviors to a code graph is introduced in [11]. However, it is not feasible to apply it in practice due to the excessive time requirement for the comparison. In addition, both methods would not be an analytic solution over the anti-virtualization technique.

2.2. Automated Malware Analysis Tool

In order to analyze the massive volume of malicious codes, there have been various reporting tools for the malware investigation. In accordance with [3], 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.

2.2.1. 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 are 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-debugging 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 [4].

2.2.2. 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 advantageous to track/analysis CPU instructions due to the fact that it utilizes 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 [5].

2.2.3. 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 [6]. The sample code would be tested in an available virtual client picked by the server, and the programmatic 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 [12].

2.3. Limit of Malware Analysis Tools

There have been growing number of infection to various systems by hiding malware in web-sites and web-disks after hacking those online services [13]. Most cyber attack incidents to the financial banking services or broadcasting stations in Korea are resulted from the malware.

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.

Existent Sandbox based malware analysis tools embed several problems as follows[7]. ZeroWine takes too much time to analysis malicious behavior since it does not have filtering service on APIs from the execution. Anubis is too slow to report the analytic result due to the fact that it is web-based analyzing services. In addition, entire executed APIs are not illustrated in the reports except the suspicious behaviors. The analytic result from the Cuckoo Sandbox is easily influenced by its plug-in functions while they are essential requirements for the operation. Table 1 shows the information to be extracted for malware analysis tools. ZeroWine and Cuckoo Sandbox have an function of Anti-VM detection, but they do not detect the evasion method of sandbox.

Table 1. Information to be Extracted for Malware Analysis Tools

	ZeroWine	Anubis	Cuckoo Sandbox
API	○	×	○
DLL	○	○	○
Registry	○	○	○
Files	○	○	○
Process	○	○	○
Network	×	○	○

String	○	×	○
Packing Status Detection	○	○	○
Anti-VM Detection	○	×	○

3. Evasion Method of Sandbox-based Analysis Tools

Recently, a growing number of malware is equipped with several detour techniques against Sandbox detection mechanisms, which can nullify the defense system with ease [14].

3.1. Malicious Activities Triggered by Specific User Inputs

Malware with this stealth function activate the malicious behaviors by various user inputs, such as mouse clicks or movements. As described in Figure 2, 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 [15]. 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 [16].

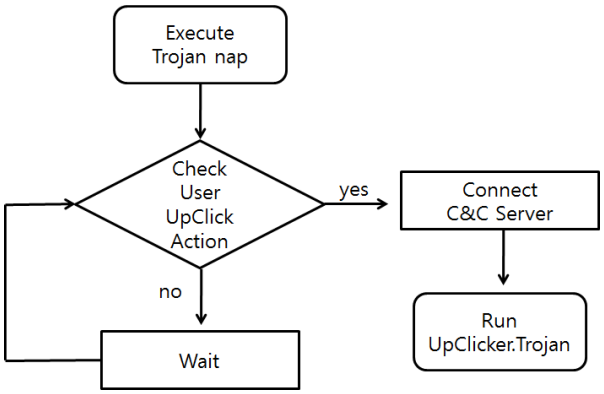

Figure 2. Activation of UpClicker. Trojan by Mouse Clicks

Another way to make malicious behavior by learning user's behavior is to display a dialogue box requiring user's answer. A dialogue box can be made by MessageBox() or MessageBoxEx() of Windows API, while EXE or DLL is executed. Malicious behavior can be performed when a user clicked the button of dialogue box. This type of malware seems to be legitimate until specific inputs are detected, which makes it hard to be identified by the Sandbox based analysis tools.

3.2. Malicious Activities Triggered by Time Constraints

Malware with this stealth technique, 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 [17]. Figure 3 shows the malware, Trojan Nap identified at February 2013, exploiting the API of sleep() to hide its existence of malicious behavior for a certain period of time.

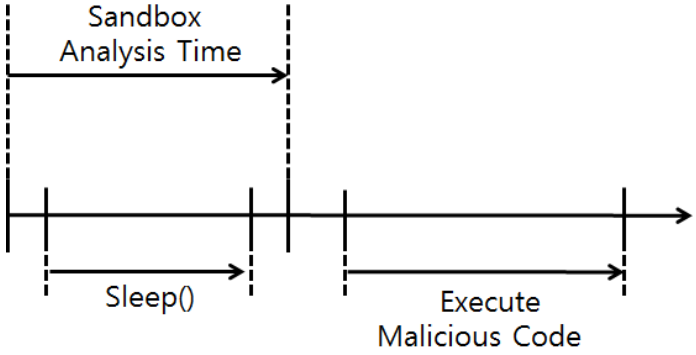

Figure 3. Hibernation of Malicious Activity against Detection

This type of malware uses scheduling technique performing malicious behavior within due time. It performs its malicious behavior on appointed time and date, and it cannot be detected if the behavior is monitored in sandbox before the appointed time. We can find this method on Trojan horse virus called Hastati which happened on March, 2013, in Korea. This causes confusion to the anti-virus software, as any legitimate programs could use it.

3.3. Malicious Activities Disabled by Detecting Virtual Execution

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 [18]. Malware checks the virtual testing environment with files, ports, and registers about the virtual machine like these.

4. Improvement on the Sandbox-based Malicious Code Analyzing Tools

4.1. Improved Architecture of Sandbox-based Tool

There are four classes of problems on existing Sandbox-based malware analysis tools. First of all, it takes too much time to analyze the sample codes using APIs if it does not filter the unnecessary results of APIs, un-relevant to malicious behaviors. Second of all, the malware that is activated by certain user inputs cannot be identified if the conditions have not been triggered[19]. Third of all, disabling time constraints out of the sample programs is required for them to operate during the execution in the virtual environment. Finally, if the malware detect the virtual testing condition, then the Sandbox is useless against them.

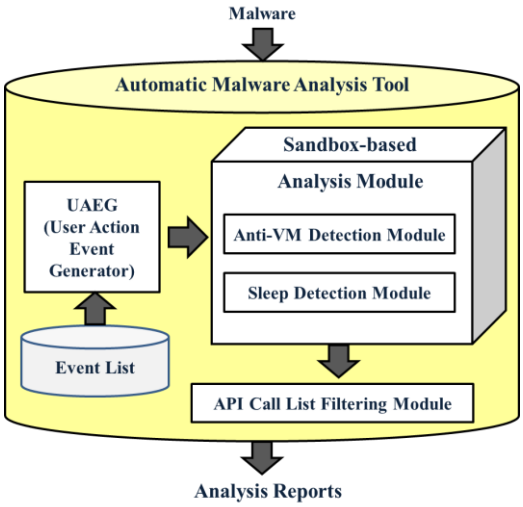

Figure 4. The Proposed Detection Architecture against Malicious Codes

As a results, improvement over those issues are required to achieve the higher accuracy on detecting malware through sandbox approaches, which are illustrated as Figure 4. UAEG (User Action Event Generator) produces various types of user inputs randomly. Anti-VM Detection Module investigates the existence of detection modules against virtual testing environment in the sample codes during the execution. Sleep Detection Module should be able to remove the time constraints for the operation during the virtual execution by manipulating the value on the API, Sleep(). In order to minimizing the time of analysis, API Call List Filtering Module should filter the APIs out of the list to report only malware relevant APIs by constructing a set of APIs in legitimate programs as a whitelist[20].

4.2. UAEG

UAEG performs the function that generates user actions in sandbox randomly. Existing sandbox-based analysis tools have a problem that it is hard to analyze the malware with malicious behavior triggered by user action. The system doesn't know which events can activate a malicious behavior, so the event generator might randomly occur events in sandbox. We configured the user actions to event list. Figure 5 shows the process that the event selected randomly from the list is inserted into the program as the input parameter. The event list can consist of the actions such as mouse click, mouse scroll, button click, specific key press, and so on.

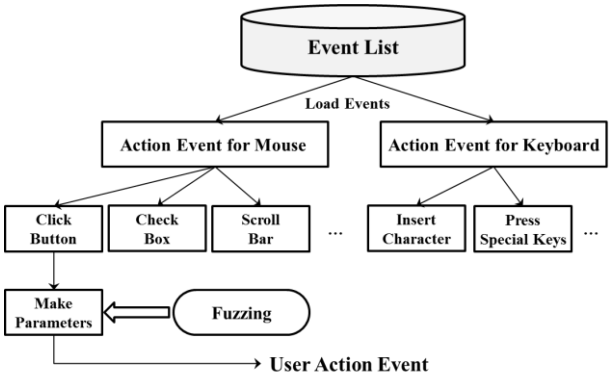

Figure 5. The Generating Flow of User Action Event

The event functions in the event list can be formed by fuzzing method in the event creation step on the UAEG, as you can see in Figure 6. The program including malicious behavior performs the behavior when a user action is happened. It is important to configure the situation. Fuzzing method is technique finding the vulnerability of software by inserting the random or abnormal values[19]. Therefore, fuzzing method can detect the unexpected vulnerabilities of software. It is helpful to apply the function to the user event generator when we analyze the unknown malwares detecting a user action.

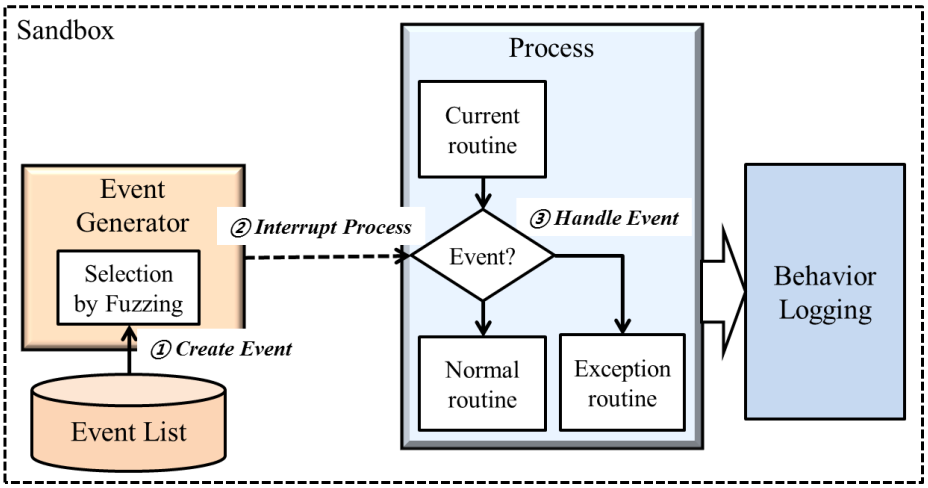

Figure 6. The Process of Generating an User Action Event in UAEG

4.3. Using of Concolic Testing

In this paper, we apply Concolic Testing among the fuzzing method to UAEG. Concolic Testing is used to verify a program, and it makes the program handle the unused routine [21, 22]. The other fuzzing method can test a number of cases, but Concolic Testing can test a rarely used routine. It takes long time to constitute event list and parameter in random. It could be a burden when it is applied to sandbox. To reduce the amount of work, we need to reduce event list and parameter value. Concolic Testing has an effect on reduction of number of events.

There is an exceptional routine in case for a malware which operates in a certain event. Generally ordinary routine is handled, but, in a specific situation, exceptional routine is occurred for malicious behavior. This is why it is important to find an exceptional routine. UAEG proposed in this paper uses Concolic Testing to find such exceptional routine. When user action event is generated by UAEG, malware triggered by user action can be analyzed by operating malicious behavior.

4.4. Sandbox-based Analyzing Tool with UAEG

In Figure 6, UAEG can generate various event cyclically through fuzzing method. This event interrupts malware program while the program is performed. If there's an exceptional routine for the event, the program flow will be changed from ordinary routine to exceptional routine. Like this, we can analyze the malicious behavior which is performed according to user action event.

The most important purpose of sandbox is to record various information performed while analyzing malware. However, sandbox won't get user action as an input, as it is an automatic tool. Especially it is hard to analyze a malicious code designed for evading sandbox analysis by handling user action. When UAEG is operated, the evasion method can be disarmed by generating user action by force.

4.5. Result of Applying UAEG

We analyzed Upclicker program in Cuckoo Sandbox not applying user action, and found that nothing happened but to call two APIs, NtAllocateVirtualMemory() and SetWindowsHookExA(), because mouse click event was not occurred in sandbox. After using UAEG, we got different result. Sample code was submitted to temporary directory, and a file named MCPUPlayer.exe in the system32 directory was made. Also we could find a new process and key value of registry.

Table 2. Analysis Information before and after Using UAEG

	Extracted Information
Before using UAEG	[API] NtAllocateVirtualMemory → SetWindowHookExA
After using UAEG	[Registry] HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\ CurrentVersion\Run HKEY_LOCAL_MACHINE\SOFTWARE\Classes\http\shell\ open\command [File] C:\WINDOWS\system32:MCPUPlayer.exe [Process] Explorer.exe [API calls in Unplicker.exe] WriteProcessMemory → CreateRemoteThread [API calls in Explorer.exe] CreateProcessInternalW → NtAllocateVirtualMemory →WriteProcessMemory → CreateRemoteThread

Explorer process is invoked from Upclicker process through user action event. Table 2 shows the difference between before and after user action event. The result of analyzing behavior of Upclicker process says API is operated for injecting thread in Explorer process. API calls in Explorer process say a malicious code was injected in the process. As a result, we confirmed that a malicious behavior was triggered by user event generator and the behavior information was logged.

5. Conclusion

In order to produce more reliable analytic results, this paper is to improve Sandbox-based malicious detection techniques over the stealth functions which the malware have equipped with. There are a few methods to avoid being analyzed by sandbox, which is recently used: operation with user action, delaying malicious behavior, and checking virtual machine's operation. We designed UAEG for countering first method. User action is generally not occurred in existing sandbox, so malware triggered by user action doesn't handle malicious behavior while being in the sandbox. A module designed in this study makes user action occur in sandbox, which leads malware to do some malice. Concolic Testing is applied for leading user behavior. This module may increase probability of detection for malware which has detour of sandbox. This is expected to help analyzing malware more efficiently.

Acknowledgment

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] Statistics Malware, AV-TEST - The Independent IT-Security Institute, http://www.av-test.org/en/statistics/malware/
[2] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian , J. Nazario. Automated Classification and Analysis of Internet Malware. In Proceedings of the 10th international conference on Recent advances in intrusion detection (RAID 2007), **(2007)** September
[3] Carsten Willems, Thorsten Holz, and Felix Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. IEEE Security & Privacy, 5, 2 **(2007)**
[4] J. Koret, ZeroWine - Malware Behavior Analysis, http://zerowine.sourceforge.net/
[5] Anubis, Malware Analysis for Unknown Binaries, International Secure Systems Lab., https://anubis.iseclab.org
[6] Cuckoo Sandbox, Cuckoo Foundation, http://www.cuckoosandbox.org/
[7] Jung-Uk Joo, Incheol Shin, Tong-Wook Hwang, and Minsoo Kim. An Improvement on the Sandbox-based Malicious Code Analyzing Tool. Proceedings of SMA, **(2014)** December
[8] M. Christodorescu, S.Jha. Static analysis of executables to detect malicious patterns. The 12th USENIX Security Symposium, **(2003)**
[9] Fabrice Bellard. QEMU, a fast and portable dynamic translator. Proceedings of the annual conference on USENIX Annual Technical Conference, Anaheim, CA, **(2005)** April
[10] N. Park, Y. Kim, B. Noh. A Behavior based Detection for Malicious Code Using Obfuscation Technique. Journal of KIISC, 16, 3 **(2006)**
[11] J. Lee, K. Jeong, and H. Lee. Detecting Metamorhpic Malware using Code Graphs. ACM Internal Symposium on Applied Computing (ACM SAC), **(2010)** March
[12] Digit Oktavianto and Iqbal Muhardianto, Cuckoo Malware Analysis, PACKT Publishing open source **(2013)**
[13] Lucian Constantin. Hackers make drive-by download attacks stealthier with fileless infections. PCWorld, **(2014)** September
[14] Ahnlab, TrusWatcher: The Next-generation Threat Prevention System, White paper, AhnLab, Inc., **(2012)**
[15] Abhishek Singh and Yasir Khalid, Don't Click the Left Mouse Button: Introducing Trojan UpClicker, Threat Research Blog, FireEye, **(2012)** December
[16] Abhishek Singh and Zheng Bu, Hot Knives Through Butter: Evading File-based Sandboxes, Threat Research Blog, FireEye, **(2013)** August
[17] Abhishek Singh and Ali Islam, An Encounter with Trojan Nap, Threat Research Blog, FireEye, **(2013)** February
[18] K. Han, I. Kim, E. Im. Malware Family Classification Method using API Sequential characteristic. Journal of Security Engineering, 8, 2 **(2011)**
[19] Jung-Uk Joo, Incheol Shin, Tong-Wook Hwang, and Minsoo Kim. The Trigger of Malicious Behaviors in Sandbox. Proceedings of FGCN, **(2014)** December
[20] Godefroid, P., Levin, M. Y. and Molnar, D. A.. Automated Whitebox Fuzz Testing. In NDSS, Vol. 8,. **(2008)** February
[21] Xiao Qu, Brain Robinson. A case study of concolic testing tools and their limitations. Proceedings of Empirical Software Engineering and Measurement (ESEM), **(2011)** September
[22] YoungJoo Kim, Moonzoo Kim, Yunho Kim, and Uijune Jung. Comparison of Search Strategies of KLEE Concolic Testing Tool. Journal of KIISE: Computing Practices and Letters, 18, 4, **(2012)**

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.

Tong-Wook Hwang, he received the BE degree in Dept. of Computer Science from the Korea Advanced Institute of Science and Technology in 2010. He is a graduate student in the Master of Engineering, Korea Advanced Institute of Science and Technology. His research interests include malicious code analysis and security in virtualized environments.

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.