An Effective Framework of Behavior Detection-Advanced Static Analysis for Malware Detection

Maya Louk[1], Hyotaek Lim[2]
Department of Ubiquitous IT, Graduate School
Dongseo University
Busan, South Korea
mayalouk@gmail.com
[2]Division of Computer and Engineering Dongseo University
Sasang-Gu, Busan 617-716, Korea
htlim@dongseo.ac.kr

HoonJae Lee[3] , Mohammed Atiquzzaman[4]
[3]Division of Computer and Engineering Dongseo University
Sasang-Gu, Busan 617-716, Korea
hjlee@dongseo.ac.kr
[4]School of Computer Science, University of Oklahoma,
Norman, OK 73019-6151
atiq@ou.edu

***Abstract*— The recent development of malwares is rapidly progressing and intruders are getting better at concealing malwares to deceive users while researchers are developing security systems. One of the methods which is commonly used is behavior detection. In this paper, a more efficient behavior detection method and the framework of intrusion malware security system is presented. In addition, the implementation of the prototype and the result of the discussion is presented under advanced static analysis which is added PE Header study. Our proposed framework will (1) contribute to improve the security system for malware detection, especially to detect sophisticated malware, (2) show the effectiveness of behavior detection to memory performance, and (3) how advanced static analysis matches the algorithm for malware detection.**

***Keywords*—Advanced Static Analysis, Efficient Behavior Detection, Framework, Malware, Sophisticated malware.**

I. INTRODUCTION

At the current time, a new trend emerges in which sophisticated malware is equipped with obfuscation code with the result that malware detector is getting impeded in detecting malware. Securelist reports that in 2013, the most sophisticated Android Trojan is called Obad.

Obad's code is infused with obfuscation method and the code exploits three of the unpublished weakness in the current detection method. Among these infamous Obad is one that enables Trojan to take on the role of extended Device Administrator, without listing it on the device as a known rightful program. It makes the malware is near impossible to be removed from the device because it appears to be a friendly known program. In addition, Obad uses multiple other methods to infect the device. Fake Google Play store, spam text message and redirected cracked sites are among the methods used by Obad to spread itself. These cybercriminals behind the Obad are capable of controlling the Trojan via pre-defined strings in text messages. The Trojan itself is able to perform several malicious actions, like sending text messages, pinging a specified resource, operating as a proxy server, connecting to a specified address, downloading and installing a specified file, sending a list of apps installed on the device, sending information on a specific app, sending the victim's contacts to the server and performing commands specified by the server [9]. McAfee catalogs 100,000 new malware samples every day or about one malware every second. Malwares are getting more sophisticated. As the volume is increasing, the nature of the threat continues to become more dangerous. One of the best ways to construct a standard security system is to electronically "mark" malware using an extracted fabrication of previously known malware [5]. Behavior analysis has been used and developed by many researchers. This paper will describe a malware or tainted data detection framework in the context of taunting system. All files and links will be scrutinized using a more efficient behavior analysis detection approach. The approach is added with static analysis which is differently made in advanced scale from the previous version, in order to make it more efficient and expeditious which added with PE Header study. This paper will elucidate the result of the prototype implementation in which will be continued to be developed for mobile device usage, especially in the accuracy of detecting and total time spent.

II. RELATED WORK

Since 2007, there has been lot of effort to obtain efficiently non-time-and-memory-consuming behavior detection. Wagener, et al. in malware behavior analysis explain the existence of a malware behavior analysis based on using experimental classification process via phylogenetic tree [3]. Baldangombo et al. use static analysis approach with data mining method to classify: SVM, J48, and Naïve Bayes [7]. You et al are contributing in launching a brief survey about malware obfuscation [6]. Wang et al are contributing with the approach in behavior feature generation, made specially to detect obfuscated malware in behavior dependence graph development [15]. Vinod et al. in a survey on malware methods are also contributing in behavior detection and development and the characters of obfuscation method that used by intruders [1]. Kirda et al are contributing in behavior spyware by using dynamic and static analysis approach [17]. Liao is contributing for testing

tainted and benign from various data with PE-Header Study [8]. However, since 2007, a method has been developed in behavior detection by combination techniques and approach, just like data mining. This paper is presenting an approach for efficient non-time-and-memory-consuming behavior detection in a computational system and an advanced static analysis

III. SYSTEM'S FRAMEWORK

This framework system aims to detect sophisticated malwares which are hard to detect because of obfuscation code fortification. Malwares which are moving inside a computation system could be detected through the malware's behavior itself.

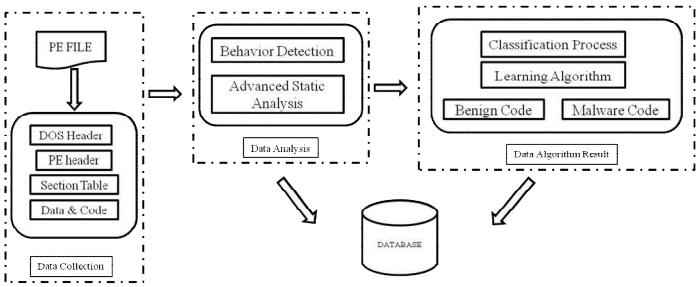

Fig. 1. Malware Analysis System

A. Taunting Warning

The framework system in this malware's analysis is shown in figure 1. The system is located in a computation system as the "main gate" before the users want to install a certain program or simply browse a link outside the particular computation system. The "main gate" is programmed with taunting warning system to assume that every data unknown to the system or which has not been seen previously is dangerous. Every would-be-installed application or link will warn the system to show a taunting warning to the user as follow:

> *Warning: installing this application may harm your computational system. Would you like to check it?*

The taunting warning is applied to all would-be-installed application. Its role is to warn users before installing anything unknown to the computation system.

B. PE File Study

The "Portable Executable" (PE) file format is the configuration of the binary programs (exe, dll, sys, scr) for MS Windows NT, Windows 95 and win32s. It can also be utilized for object files (bpl, dpl, cpl, ocx, acm, and hatchet). The configuration was designed by Microsoft and afterward in 1993 institutionalized by the Tool Interface Standard Committee (Microsoft, Intel, Borland, Watcom, IBM and others) , clearly focused around the "common object file format" (COFF), the configuration utilized for object files and executables on a few UNIX and VMS Oses. The PE information structures are: DOS (Disk Operating System) header, DOS stub, PE File Header, Image Optional Header, Section Table - which has a trailing exhibit of Section Headers - Data Directories - these indexes contain pointers to information in the individual areas and, finally, the individual Sections themselves [19].

DOS Header begins with enchantment number 4d 5a 50 00, and the last 4 bytes is the area of PE header in the binary executable record. Different fields are not that intriguing. The PE header contains more information [18]. In Figure 2, please discover the structure of PE Header. We just show the data that are intriguing for this paper.

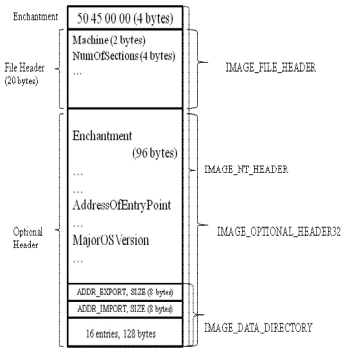

Fig. 2. Structure of PE Header

C. Behavior Detection Method

Behavioral Analysis examines the malware against the computation system which is consisting of the file itself, the registry (windows), the network, and the OS components. Monitoring process is required in behavioral analysis to scan the whole file involved in the malware **Memory analysis** examines memory of the infected system to extract artifacts relevant to the malicious program. Memory analysis saves time and allows the investigator to take shorter pathway when studying the specimen's behavior. Both Processes is interconnected one another.

D. Data Algorithm

Elements as follows:

- Data Collection
  Data collection will save all entry data, and in the particular example, the PE file will be used. The spotlight is directed to PE header, because the windows loader actually loads PE header into a process's address space during run time of a binary executable. There are some well defined data structures defined in winnt.h for each of the major part of the PE header. As shown in Figure 2, PE header consists of three parts: (1) a 4-byte magic code, (2) a 20-byte file header and its data type is IMAGE_FILE_HEADER, and (3) a 224-byte optional header (type: IMAGE_OPTIONAL_HEADER32). The optional header itself has two parts: the first 96 bytes contain information such as major operating systems, entry

point, etc. The second part is a data directory of 128 bytes. It consists of 16 entries, and each entry has 8 bytes (address, size) [18].

- Data Analysis
  The analysis uses a more efficient behavioral detection method and advanced static analysis. Behavior detection differs from the surface scanning method. Behavior detection identifies the action performed by malware rather than the occurrence of particular binary pattern. As assumed, any malwares perform similar behavior, thus dissimilar syntax with similar behavior will fall under suspicion. This mechanism helps in detecting malwares which keep on generating new mutations as these malwares will always use the system resources and services in similar behaviors. [1]. The combination between the scrutinized PE file which is an execution from binary code with behavior suspicious analysis will make the system even greater than the previous.
- Data Algorithm Result
  This part is meant to classify the data that have been collected so far and scrutinizing those data via behavior and advanced static analysis to show the result, whether those data fall under benign or tainted category.

The system is connected with the database, in which the database collects the characteristic of any of the previous scrutinized malware. In this paper, the prototype implementation is using self-made local database.

IV. Effective Behavior Detetction

The effective Behavior Detection, system will focus on several items. Leverages profiles are also used to detect typical malware actions before the occurrence of any infections. It figures out what the malware will do, when and where, thus gaining the upper hand to prevent it before it does any damages. Several things to know [16]:

- Registry settings
  Normal executables rarely update registry, configuration, or system file settings; so any activity of this sort warrants investigation.
- Processes/services
  Several processes and executables are always launched by specific particular processes and executables. If the patterns are somewhat different, this difference might indicate the existence of malware.
- Injected code
  Intruders are using code injection so that computer worms or malware are enabled to make propagation in a computation system.
- Malware writers always need to be updated, so it is efficient for them to inject a stub program called a dropper in order to generate itself. This dropper accesses the network and automatically downloads the updated malware code to the device. It is imperative to stop any executable that acts like a dropper. Programs that add or change any executables should be fall under heavy suspicion.
- New executables(EXE)
  The ever growing EXE malware are able to become anything from key-logging program or spreading-adware.
- Domains/protocols
  A new Trojan program disguising itself using advertisements uses a DNS-based email validation protocol called the Sender Policy Framework (SPF) to enable attacker remotely control the Trojan with certain instructions without being detected, according to security researchers from Symantec.

A related technique of environment-aware malware is to make exhibited behavior depend on the execution context, such as the process in which the malware code is running. A common trick used by Trojans is to register one of the malware components using App-Init Dlls. These Dlls get loaded as part of any process running on an infected host, including Internet browsers or Anti Virus software. When the Trojan code is started in the context of another program, it can use the process name (such as the executable name or command-line) to determine its execution environment. Typically, when running as part of AV software, malware disables the security tool and terminates the Anti-Virus service. Otherwise, when executed as part of a browser, the attackers are able to steal user's important private data [2].

Behavior detection in a computation system is able to recognize, Source (SC) and TaintSource:

- SC{FilepathTaint, RegistryTaint, ServicesTaint, PIDTaint, NetworkTaint, ProtocolsTaint}
  FilepathTaint recognizes routines for manipulating filename paths in a way compatible with the target operating system-defined file paths, **RegistryTaint** recognizes database configuration, **ServicesTaint** recognizes malware service executables, **PIDTaint** recognizes active process in a computation system, **NetworkTaint** recognizes malware websites detection, and **ProtocolsTaint** recognizes DNS Based e-mail validation.
- Taint{BehaviorTaint, CodeTaint, MemoryTaint}
  BehaviorTaint recognizes malware behavior SC {FilepathTaint, RegistryTaint, ServicesTaint, PIDTaint, NetworkTaint, ProtocolsTaint}, **CodeTaint** recognizes injected code in a PE file or EXE file (presumably sophisticated malware), **MemoryTaint** recognizes memory corruption or buffer overflow.

V. Advanced Static Analysis

The first task of the static analysis step is to disassemble the target binary and generate a control flow graph from the disassembled code. A control flow graph (CFG) is defined as a directed graph $G = (V, E)$ in which

vertices $u, v \in V$ represent basic blocks and an edge $e \in E : u \rightarrow v$ represents a possible flow of control from *u* to *v*.

Static code analysis launches time-consuming and error-prone manual code review. Different from dynamic testing, in which the software has to be executed in the first place, static code analysis is launched directly on the source code, so the quality checks begins just before the code is ready for integration and test process. Static code analysis preparation will produce structure simplification and structure extraction & annotation. With some static analysis that have been developed and ready to use, like IDA Pro and Pestudio which are included in best 4 list according to statistical analysis launched by SAN institute. In this paper proposed an idea advanced static analysis with PE Header Study that already pronounced by Liao to prove that PE Header study faster to distinguish between malware and benign. The peculiarities in the header fields demonstrate a considerable measure of data about the record. Case in point, the Numberofsections in the document header demonstrates the quantity of areas in the PE File; the Sizeofinitializeddata in the discretionary header demonstrates the measure of the introduced information segment of the record; if the record implants a symbol, it will demonstrate in the .rsrc field in the area header [8]. This endeavor requires certain experience and the analyst needs to know Windows Internals, Assembly language, and compiler code, though the process itself is able to present the extend capabilities of the binary code, including what it can do and conditions required for it to run. This is most effective technique of analysis.

Recognizing C code constructs in assembly [12]:

- Recognizing if & Nested if Statements
  The goal is to learn how to recognize if statements
- Recognizing Loops
  Loops and repetitive tasks are very common in all software, thus the goal is to be able to recognize them manually.
- Analyzing switch Statements
  Switch statements are used by programmers (and malware authors) to make decision based on a character or integer. For example, backdoors commonly select from a series of actions using a single byte value. Switch statements are compiled in two common ways: using the If style or using jump tables.
- Disassembling Arrays
  Arrays are used by programmers to define an ordered set of similar data items. Malware sometimes uses an array of pointers to strings that contain multiple hostnames that are used as options for connections.
- Identifying Structures
  Structures (or *structs*, for short) are similar to arrays, but they comprise elements of different types. Structures are commonly used by malware authors to group information. It's sometimes easier to use a structure than to maintain many different variables independently, especially if many functions need access to the same group of variables
- Analyzing Linked List Traversal

The principal benefit of using a linked list over an array is that the order of the linked items can differ from the order in which the data items are stored in memory or on disk

VI. Analysis & Discussion

The Computational system specification: processor: Intel(R) Core (TM) i3-2100 CPU @ 3.10 GHz, Memory (Ram): 4.00 GB, 64-bit Operating System. In this evaluation, five different kinds of viruses are used.

TABLE I. STEM PERFORMANCE RESULT

Virus Category	Virus Source	System Performance	
		Stabilized Time (ms)	Memory Remains (%)
Trojan Simulator	http://vxheavens.com/vx.php?id=st00	269.36	85.69
Eicar	http://www.eicar.org/download/eicar_com.zip	300.96	92.63
Phishing	http://www.filefactory.com/file/b3d7e10/n/All_Phishing_Pages.rar	201.75	79.61
CMOS	http://oc.gtisc.gatech.edu:8080/search.cgi?search=CMOS	197.52	87.38
PUA	http://nl.inncdn.com/IT/utorrent-64bit.exe	223.25	89.45

Overall, the performance of the computational system is considered fulfilling the target set by the researcher, because largely, the malwares that are used in the test are detected properly, with 80% accuracy. The malwares are:

- Trojan Simulator in which if activated, it will create an active process in the memory. It will also have an auto start entry in the system registry (TSServ.exe creates an auto start entry in the Windows registry, TSServ.exe gets loaded into memory and has a unique fingerprint in its code section).
- Eicar in which the file is a legitimate Denial of Service program which is an attack that causes an application to stop responding so that the user has no other choice but to close the interrupted application. This exploit can be modified into a remote code execution attack via exploitable buffer overflow).
- Phishing is designed to steal data. Phishing can be performed by disguising a malicious website into a known and trusted site (e.g., bank website or a webmail). A fake login screen is popped up to lure an unsuspecting user to fill in his/her personal information.
- CMOS is a method used by malware writers to obfuscate the application. Encryption and encoding are combined in

order to modify the original source code, making it harder to detect

- PUA (Potentially Unwanted Applications) is a term to describe applications that—while not that malicious—are generally considered inappropriate for business networks. The classifications are: adware, dialer, non-malicious spyware, remote administration tools, hacking tools.

Stabilized Time (ms) is the time needed for the system to recognize the whole virus source, via executable binary file. Total Stabilized time T(n) consists of three elements: setup time (ts), execution time (te), and post-processing time (tp). In this evaluation which uses detection Behavior and prove advanced static analysis can detect faster and consume less memory.

VII. Conclusion

An efficient and effective behavior detection framework for security on computation system equipped with advanced static analysis is still being developed to enable the detection of sophisticated obfuscated malwares. With the progression of digital era, criminality is also inevitably increasing. This paper gives a practical solution to detect obfuscated malware by reducing the time and memory needed. The overall framework will be under constant development to present an accurate result and to display recognizing code construct and structure in a greater detail. For future work, this framework will be developed for the Android mobile programming to enable mobile user to access the framework.

Acknowledgment

This research was supported by the National Research Foundation of Korea under Grant 2011-0009349, and by the BB21 project of Busan Metropolitan City.

References

[1] Vinod, P., et al. "Survey on malware detection methods." *Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security (IITKHACK'09)*. 2009.

[2] Kolbitsch Clemens. (April 15 2014). Analyzing.

[3] Wagener, Gérard, and Alexandre Dulaunoy. "Malware behavior analysis."*Journal in computer virology* 4.4 (2008): 279-287.

[4] Moser, Andreas, Christopher Kruegel, and Engin Kirda. "Limits of static analysis for malware detection." *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*. IEEE, 2007.

[5] Mcafee. (April 2 2014). Infographic: The State of Malware 2013. Retrieved from http://www.mcafee.com/us/security-awareness/articles/state-of-malware-2013.aspx.

[6] You, Ilsun, and Kangbin Yim. "Malware Obfuscation Techniques: A Brief Survey." *BWCCA*. 2010.

[7] Baldangombo, Usukhbayar, Nyamjav Jambaljav, and Shi-Jinn Horng. "A STATIC MALWARE DETECTION SYSTEM USING DATA MINING METHODS."*International Journal of Artificial Intelligence & Applications* 4.4 (2013).

[8] Liao, Yibin. "PE-Header-Based Malware Study and Detection." Retrieved from the University of Georgia: http://www.cs.uga.edu/~liao/PE_Final_Report.pdf (2012).

[9] Emm, Raiu. (April 15 2014). Kaspersky Security Bulletin 2013. Malware Evolution. Retrieved from http://www.securelist.com/en/analysis/204792316/Kaspersky_Security_Bulletin_2013_Malware_Evolution?print_mode=1

[10] Goubault, Eric, et al. "Static analysis by abstract interpretation: A mathematical programming approach." *Electronic notes in theoretical computer science* 267.1 (2010): 73-87.

[11] Baysa, Donabelle, Richard M. Low, and Mark Stamp. "Structural entropy and metamorphic malware." *Journal of Computer Virology and Hacking Techniques*9.4 (2013): 179-192.

[12] Sikorski, Michael, and Andrew Honig. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, 2012.

[13] Elhadi, Ammar Ahmed E., Mohd Aizaini Maarof, and Ahmed Hamza Osman. "Malware Detection Based on Hybrid Signature Behavior Application Programming Interface Call Graph." *American Journal of Applied Sciences* 9.3 (2012).

[14] Borello, Jean-Marie, and Ludovic Mé. "Code obfuscation techniques for metamorphic viruses." *Journal in Computer Virology* 4.3 (2008): 211-220.

[15] Wang, Rui, Xiaoqi Jia, and Chujiang Nie. "A Behavior Feature Generation Method for Obfuscated Malware Detection." 2012 International Conference on Computer Science & Service System (CSSS), IEEE, 2012.

[16] Rothman Mike. (April 4 2014). Evolving Endpoint Malware Detection: Behavioral Indicators. Retrieved from https://securosis.com/blog/evolving-endpoint-malware-detection-behavioral-indicators.

[17] Kirda, Engin, et al. "Behavior-based spyware detection." *Proceedings of the 15th USENIX Security Symposium*. 2006.

[18] Dr. Fu's Security Blog. (April 20 2014). Malware Analysis Tutorial 8: PE Header and Export Table. Retrieved from http://fumalwareanalysis.blogspot.kr/2011/12/malware-analysis-tutorial-8-pe-header.html.

[19] Thehackademy. (August 18 2014). PE File Structure. Retrieved from http://www.thehackademy.net/madchat/vxdevl/papers/winsys/pefile/pefile.html