Poster: Limitations and Improvement of Dynimic Analysis Environment for Malware Analysis

HoMook Cho
Cyber Security Research Center
KAIST
Daejeon, Korea
chmook79@kaist.ac.kr

SangYong Choi
Cyber Security Research Center
KAIST
Daejeon, Korea
csyong95@kaist.ac.kr

KiMoon Han
Cyber Security Research Center
KAIST
Daejeon, Korea
linuzen@kaist.ac.kr

KwanSik Yoon
Cyber Security Research Center
KAIST
Daejeon, Korea
ksyoon@kaist.ac.kr

I. Motivation

Threats using malware are categorized as the most severe threat in the cyber space. Recent malwares are becoming more complex and elaborate to make the analysis difficult and over 50% are maintained undetected in anti-virus programs [1]. In this study, a new approach to increase the degree of completion for the dynamic analysis environment is to be presented based on the result of the verification on actually collected recent malwares, in regards to the theoretical techniques that make the analysis difficult.

II. Limitations of Malware Analysis Evnerionment

Traditionally, there are two ways to analyze malware. First is the static analysis, which doesn't actually run the malware but uses reverse engineering technique. While static analysis can analyze the structure and performance characteristics of malware with no limitations on running condition, it is hard to automate the analysis functions, and requires significant time and effort because of concealing techniques such as encryption or packing [2]. Second is the dynamic analysis, which actually runs the malware and analyze whether it performs any malicious behaviors on the system or not. The monitored subjects in dynamic analysis are process, registry, file system, and network changes. While dynamic analysis can increase the detection rate than static analysis and can automatically perform the analysis, which reduced the analysis time, specific environment or conditions are required for the malware to run, and the malware may recognize the analysis environment and evade [3]. However, in order to efficiently analyze new malwares that are increasing explosively, studies on dynamic analysis are becoming an important issue recently [4].

There are 4 approaches to the dynamic analysis of malware [5].

- ***Function call monitoring***: this approach uses API hooking method to detect the API that the malware calls and analyzes malicious behaviors.
- ***Parameter analysis***: this approach traces the parameter and function return value of each function in the same object and performs relation analysis.
- ***Information flow analysis***: this approach analyzes how the program processes data.
- ***Command tracing***: this approach analyzes the order of processed commands while the program is running.

In order to increase the effectiveness of dynamic analysis of malware, the malware needs to run more completely to draw accurate behaviors. For dynamic analysis-based technique for behavior information collection, methods using emulator and virtual machine were proposed. Emulator-based method is divided into two types depending on the emulating resource. First, memory & CPU emulator consists of core elements to run commands, and is used by many anti-virus program engines because it can analyze malicious behaviors of specific file quickly with no influence on the actual system. The second approach, system emulator, emulates not only the memory & CPU, but also devices like network card and storage, and can analyze malicious behaviors more precisely. Virtual machine-based analysis method is more advanced than the two previous methods. This method uses the virtual environment that is close to the actual computer, and has been used most frequently in recent dynamic analysis because it can analyze malware more precisely [6].

However, recent malwares are applied with analysis environment avoidance technology, so limitations started to appear. There are 4 fundamentals that recognize the analysis environment applied to the malware [4].

- ***Hardware detection***: Virtual machine's device is easily identified, mainly the VMWare's network interface recognizing the defined pcnet32.
- ***Running environment detection***: It checks to see if the environment can monitor processes like debugger state.
- ***External application***: It checks to see if monitoring applications like process monitor are running in the environment where malware is running.
- ***Operation behavior***: It uses the system timer to recognize the difference in the running time of commands with specific authority, to alter the initial operation time.

It was confirmed through the experiment that, besides above 4 analysis environment detection techniques, there were recent malwares that required interactions with users to run, as see in Table 1.

Approximately 824 recent malwares collected in the second half of 2014 were checked for automatic start on actual computer. This is because using a virtual machine would fall under the "hardware detection" among the analysis environment detection techniques mentioned above, and new detection techniques can't be verified. For the same reason, the experiment was run in a clean environment with no detection application installed.

Table 1. The Execution Result of Malware Analysis Environment

Execution	Count	Reason of Failure
Success	702	(85.2%)
Failure	66	Need Interactive Action such as Install, Mouse Click, etc.(8.0%)
	56	Need Execution Environment such as .net framework, DLL files, etc.(6.8%)
Total	824	

As a result, 14.8% of malwares did not start automatically. Main reasons included waiting for user's input for installation, such as mouse clicks, or the operation environment was needed to run the malware. Of course the operation environment to run the malware can be solved by presetting it in the analysis environment, but besides that, about 8% of malwares were applied with new detection techniques.

III. Suggestion for the Environment for Malicious Behavior Collection

As previously mentioned, complete collection of malicious behaviors should be prioritized for more accurate identification of malwares. The recent collection in virtual environment is showing its limitations, and the experiment result showed that the malware operation environment should be adjusted dynamically, and the user behavior information that the malware requires should be entered dynamically for complete automatic collection. For this, this study limits the technique of dynamic handling of malware requirements by detecting the system screen in actual computer environment. In other words, this method runs the malware and analyzes the system screen to dynamically apply the environment that the malware requires and user input. The system architecture for this is seen in Figure 1.

The suggested approach can counteract do the malware's first virtual environment detection technique, "hardware detection", through the use of Real-Machine. In addition, the operation environment that the malware requires is dynamically handled, not only it can counteract to the fourth virtual environment detection technique, "operation behavior', it can also counteract to new detection techniques that appear in the experiment. Therefore, it can provide more complete malware dynamic analysis-based environment that becomes the basis of malware collection.

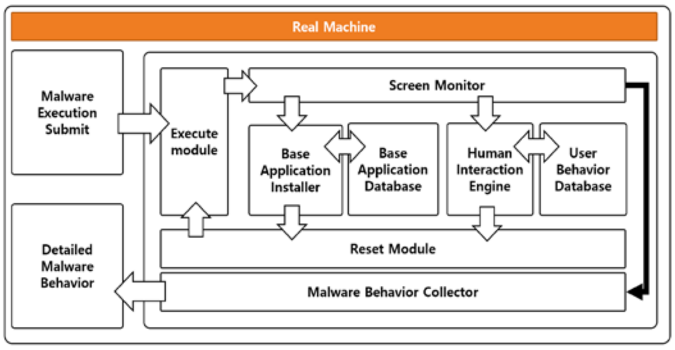

Fig. 1. Suggested Architecture based on Real-Machine

IV. Open Questions

The method suggested in this study can resolve 2 of the malware analysis environment detection techniques, but a new idea is needed to counteract to malware collection method. In other words, the debugging state, API hooking, and monitoring program that are generally used for behavior collection are problems that still need to be solved. To raise the effectiveness of suggested approach, a collection technique that avoids malware's detection technique is required.

V. Conclustion

In order to improve the accuracy of malware analysis, an accurate behavior information collection based on complete operation of malicious binary should be prioritized. However, it was proven through the experiment using actual malwares that the known dynamic analysis technique has limitations in providing such basis. This study suggested a dynamic analysis-based environment that becomes that basis for more accurate analysis of malware behaviors. Suggested method can be utilized as a tool of the best analysis that can effectively counteract to continuously increasing new malwares. In order to raise the degree of completion of dynamic analysis-based technique, future study would include the development of mechanism that counteract to human interaction engine and virtual analysis environment avoidance , and the algorithm that can classify malicious behaviors in the collected binary behavior information.

References

[1] European Union Agency for Network and Information Security (ENISA): ENISA threat landscape 2014, http://www.enisa.europa.eu-/activities/risk-management/evolving-threat-environment/enisa-threat-landscape/enisa-threat-landscape-2014, Dec 2014

[2] A. Moser, C. Krügel, and E. Kirda, "Exploring multiple execution paths for malware analysis", IEEE Security and Privacy, pp. 231-245, May 2007.

[3] V. Thomas and P Ramagopal, "The rise of autorun-based malware", McAfee, 2009.

[4] M. Egele, T Scholte, E. Kirda, and C. Kruegel, "A Survey on Automated Dynamic Malware-Analysis Techniques and Tools," ACM Computing Surveys (CSUR), Vol. 44, No. 2, pp. 1-42, Feb. 2012.

[5] Zovi, D. D. 2006. Hardware Virtualization Based Rootkits. in Black Hat Briefings and Training USA 2006