World Journal of Engineering Research and Technology
WJERT
www.wjert.org
SJIF Impact Factor: 4.326

A STUDY OF TECHNIQUES AND TOOLS USED IN MALWARE ANALYSIS

Prof. B. Rajeswari*

Assistant Professor, MERI, India.

Article Received on 24/01/2017 **Article Revised on 14/02/2017** **Article Accepted on 07/03/2017**

***Corresponding Author**
Prof. B. Rajeswari
Assistant Professor, MERI, India.

ABSTRACT

Malicious software is one of the major and serious threats on the cyber space today. The malwares being designed by attackers have the ability to change their code as they propagate. The effectiveness of traditional defenses which typically use signature based techniques is undermined by the diverse variants of malware which are highly complex making them undetectable. This paper discusses malware and its analysis by providing definitions, stages, techniques and tools to use to successfully perform malware analysis.

KEYWORDS: Botnet, Computer Security Incident Management, Malware, Manual Code Reversing.

I. INTRODUCTION

Malware analysis is the study or process of determining the functionality, origin and potential impact of a given malware sample such as a virus, worm, trojan horse, rootkit, or backdoor. Malware or malicious software is any computer software intended to harm the host operating system. It is stealth software that steals sensitive data from users, organizations or companies without their consent or permission.

Consider the following scenario which illustrates the distribution of malware and its effects. A bot is a remotely controlled piece of malware that has infected an Internet-connected computer system. This bot allows an external entity, the so called bot master, to remotely control this system. The pool of machines that are under control of the bot master is called a botnet. The bot master might rent this botnet to a spammer who misuses the bots to send

spam emails containing links to a manipulated web page. This page, in turn, might secretly install a spyware component on a visitors system which collects personal information, such as credit card numbers and online banking credentials. This information is sent back to the attacker who is now able to misuse the stolen information by purchasing goods online. All involved criminals make money at the expense of the infected user. With the rise of the Internet and the number of attached hosts, it is now possible for a sophisticated attacker to infect thousands of hosts within hours after releasing the malware into the wild.

II. NEED FOR MALWARE ANALYSIS

Malware analysis is an interesting, exciting, and challenging field of computer security research. The complexity of malware analysis is only one area of the security profession that is constantly evolving. There are three typical use cases that drive the need for malware analysis:

Computer security incident management: If an organization suspects that some malware may have entered into its systems, a response team may wish to perform malware analysis on any potential samples that are discovered during the investigation process to determine if they are malware and, if so, what impact that malware might have on the systems within the target organizations' environment.

Malware research: Academic or industry malware researchers may perform malware analysis simply to understand how malware behaves and the latest techniques used in its construction.

Indicator of compromise extraction: Vendors of software products and solutions may perform bulk malware analysis in order to determine potential new indicators of compromise; this information may then feed the security product or solutions to help organizations better defend themselves against attack by malware.

III. TYPES OF MALWARE

Viruses (Merriam-Webster Online, 2007) – a computer program that is usually hidden within another seemingly innocuous program and that produces copies of itself and inserts them into other programs and usually performs a malicious action (as destroying data).

Worms (Merriam-Webster Online, 2007) – a usually small self-contained and self-replicating computer program that invades computers on a network and usually performs a destructive action.

Trojans Horse (Merriam-Webster Online, 2007) – a seemingly useful computer program that contains concealed instructions which when activated perform an illicit or malicious action (as destroying data files).

Spyware (Merriam-Webster Online, 2007) – software that is installed in a computer without the user's knowledge and transmits information about the user's computer activities over the Internet **Adware** – software installed that provides advertisers with information about the users browsing habits, thus allowing the advertiser to provide targeted ads.

Backdoors (Skoudis and Zeltser, 2003) – Bypasses normal security controls to give an attacker unauthorized access.

Rootkits (Skoudis and Zeltser, 2003) – Trojan horse backdoor tools that modify existing operating system software so that an attack can keep access to and hide on a machine.

Sniffers – an application used to monitor and analyze network traffic.

IV. STAGES OF MALWARE ANALYSIS

Examining malicious software involves a variety of tasks, some simpler than others. These efforts can be grouped into stages based on the nature of the associated malware analysis techniques.

a) Fully-Automated Analysis

The easiest way to assess the nature of a suspicious file is to scan it using fully-automated tools, some of which are available as commercial products and some as free ones. These utilities are designed to quickly assess what the specimen might do if it ran on a system. They typically produce reports with details such as the registry keys used by the malicious program, its mutex values, file activity, network traffic, etc.

b) Static Properties Analysis

An analyst interested in taking a closer look at the suspicious file might proceed by examining its static properties. Such details can be obtained relatively quickly, because they don’t involve running the potentially malicious program. Static properties include the strings embedded into the file, header details, hashes, embedded resources, packer signatures, metadata such as the creation date, etc. Virus Total is an example of an excellent online tool whose output includes the file’s static properties.

c) Interactive Behavior Analysis

After using automated tools and examining static properties of the file, as well as taking into account the overall context of the investigation, the analyst might decide to take a closer look at the specimen. This often entails infecting an isolated laboratory system with the malicious program to observe its behavior. Behavioral analysis involves examining how sample runs in the lab to understand its registry, file system, process and network activities. Understanding how the program uses memory (e.g., performing memory forensics) can bring additional insights. This malware analysis stage is especially fruitful when the researcher interacts with the malicious program, rather than passively observing the specimen.

d) Manual Code Reversing

Reverse-engineering the code that comprises the specimen can add valuable insights to the findings available after completing interactive behavior analysis. Some characteristics of the specimen are simply impractical to exercise and examine without examining the code. Insights that only manual code reversing can provide include:

- Decoding encrypted data stored or transferred by the sample;
- Determining the logic of the malicious program's domain generation algorithm;
- Understanding other capabilities of the sample that didn't exhibit themselves during behavior analysis.

Manual code reversing involves the use of a disassembler and a debugger, which could be aided by a decompiler and a variety of plugins and specialized tools that automate some aspects of these efforts.

e) Combining Malware Analysis Stages

The process of examining malicious software involves several stages. However, viewing these stages as discrete and sequential steps over-simplifies the steps in malware analysis process. In most cases, different types of analysis tasks are intertwined, with the insights gathered in one stage informing efforts conducted in another. Perhaps the stages could be represented by a "wash, rinse, repeat" cycle, that could only be interrupted when the analyst runs out of time.

V. TYPES OF MALWARE ANALYSIS TECHNIQUES

There are different methodologies of malware analysis. They are as under:

Static Malware Analysis: Static or Code Analysis is usually performed by dissecting the different resources of the binary file without executing it and studying each component. The binary file can also be disassembled (or reverse engineered) using a disassembler such as IDA. The machine code can sometimes be translated into assembly code which can be read and understood by humans: the malware analyst can then make sense of the assembly instructions and have an image of what the program is supposed to perform. Some modern malware is authored using evasive techniques to defeat this type of analysis, for example by embedding syntactic code errors that will confuse disassemblers but that will still function during actual execution.

Dynamic Malware Analysis: Dynamic or Behavioral analysis is performed by observing the behavior of the malware while it is actually running on a host system. This form of analysis is often performed in a sandbox environment to prevent the malware from actually infecting production systems; many such sandboxes are virtual systems that can easily be rolled back to a clean state after the analysis is complete. The malware may also be debugged while running using a debugger such as GDB or WinDbg to watch the behavior and effects on the host system of the malware step by step while its instructions are being processed. Modern malware can exhibit a wide variety of evasive techniques designed to defeat dynamic analysis including testing for virtual environments or active debuggers, delaying execution of malicious payloads, or requiring some form of interactive user input.

Hybrid Malware Analysis: It is clear that a single view either static or dynamic is not sufficient for efficiently and accurately classifying malicious programs because of the obfuscation and execution-stalling techniques. So, researches have adapted a hybrid technique which incorporates both static and dynamic features simultaneously for better malware detection and classification.

Santos et al.[6] proposed a hybrid unknown malware detector called OPEM, which utilizes a set of features obtained from both static and dynamic analysis of malicious code. The static features are obtained by modeling an executable as a sequence of operational codes and dynamic features are obtained by monitoring system calls, operations and raised exceptions. The approach is then validated over two different data sets by considering different learning algorithms for classifiers Decision Tree, K-nearest neighbor, Bayesian network, and Support

Vector Machine and it has been found that this hybrid approach enhances the performance of both approaches when run separately.

A similar work was done by Islam et al.[7] to classify the executables into malicious and benign files using both static and dynamic features. The static features used in this work include function length frequency and printable sting information and dynamic features used are API function names and API parameters. The experiment was conducted using 2939 executable files including 541 clean files separately for every feature i.e. function length frequency, printable string information and API function calls and then for integrated method for meta classifiers SVM, IB1, DT and RF. The obtained results showed that all meta-classifiers achieve highest accuracy for integrated features and meta-RF is the best performer for all cases. The authors also compared their integrated method accuracy with those of the existing ones and found that their approach is showing the best results.

Anderson et al.[8] proposed a method, in which multiple data sources (the static binary, the disassembled binary file, its control flow graph, a dynamic instruction trace & system call trace, and a file information feature vector) are used. For the binary file, disassembled file, and two dynamic traces, kernels based on the Markov chain graphs are used. For the control flow graph, a graphlet kernel is used and for the file information feature vector, a standard Gaussian kernel is used. Then multiple kernel learning is employed to find a weighted combination of the data sources and support vector machine classifier is used to classify the dataset into malicious and benign.

VI. TOOLS OF MALWARE ANALYSIS

The tools that will be used for behavioral analysis (Dynamic) are listed below along with a brief description of what the tool does.

BgInfo - small application providing import system information such as hostname, IP address, OS version, etc.

Process Explorer – small application that find out what files, registry keys and other objects have open, which DLL's they have loaded.

Process Monitor – small application used to monitor file system, registry, process, thread and DLL activity in real-time.

PSfile - application that shows a list of files on a system that are opened remotely.

RootkitRevealer – application that scans system for known rootkit-based malware.

Streams – application that reveals NTFS alternate streams.

Strings – application that searches for ANSI and UNICODE strings in binary images.

TCPView – application providing information about TCP and UDP connections, including the local and remote address and TCP connection state.

Windump – Windows version of the powerful and flexible tcpdump sniffer.

nmap – world's best port scanner.

Fport – Identifies unknown ports and their associate applications.

Hfind (Part of the Forensic Toolkit) – application that will scan for the disk for hidden files.

Vision – reports all open TCP and UDP ports and maps them to the owning process or application.

Some tools used for behavioral analysis will be also be used for code analysis (Static). The tools used for code analysis are listed below, along with a brief explanation.

IDA Pro – popular interactive, programmable, extendible, multi-processor debugger and disassembler.

Reverse Engineering Compiler – popular decompiler.

ProcDump 32 – unpacker application

PE Explorer -- provides tools for disassembly and inspection of unknown binaries.

Windbg – windows debugging applications.

Livekd – application that allows Windbg debugger to run locally on a live system.

Debugview – an application that monitors debug output on your local or a remote system.

VII. CONCLUSION

Sophisticated malwares are posing a severe threat to the internet and computer systems. By defining and identifying malware analysis and the different stages in the analysis process, an introduction to malware analysis has been provided. This paper also highlights the existing techniques for analyzing malware. The paper also compiles a list of tools used for malware analysis. However these are not adequate to handle challenges arising from the huge amount of dynamic malware. This necessitates the need to transform analysis methods so that their potential can be leveraged to address the challenges posed in cyber space.

VIII. REFERENCES

1. https://www.sans.org
2. https://en.wikipedia.org/wiki/Malware_analysis
3. A Survey on Automated Dynamic Malware Analysis Techniques and Tools, http://www.seclab.tuwien.ac.at/papers/malware_survey.pdf

4. T. T. Analyze: A Tool for Analyzing Malware, http://www.auto.tuwien.ac.at/~chris/research/doc/eicar06_ttanalyze.pdf
5. http://www.scirp.org/journal/jis http://dx.doi.org/10.4236/jis.2014.52006
6. Santos, I., Devesa, J., Brezo, F., Nieves, J. and Bringas, P.G. OPEM: A Static-Dynamic Approach for Machine Learning Based Malware Detection. Proceedings of International Conference CISIS'12-ICEUTE'12, Special Sessions Advances in Intelligent Systems and Computing, 2013; 189: 271-280.
7. Islam, R., Tian, R., Battenb, L. and Versteeg, S. (2013) Classification of Malware Based on Integrated Static and Dynamic Features. Journal of Network and Computer Application, 36, 646-556. http://dx.doi.org/10.1016/j.jnca.2012.10.004
8. Anderson, B., Storlie, C. and Lane, T. Improving Malware Classification: Bridging the Static/Dynamic Gap. Proceedings of 5th ACM Workshop on Security and Artificial Intelligence (AISec), 2012; 3-14.