Volume 3, Issue 4, April 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Techniques for Malware Analysis

Savan Gadhiya
M.E. Student-IT Systems and Network Security,
Department of Computer Science,
Gujarat Technological University, India.

Kaushal Bhavsar
Ph.d. Student,
Department of Computer Science,
CHARUSAT, India.

Abstract: - Malware is a worldwide epidemic. Studies suggest that the impact of malware is getting worse. Two types of malware analysis are described here. One is Static Malware Analysis and the other is Dynamic Malware Analysis. Static Malware Analysis has some limitations. So, Dynamic Malware Analysis is preferable for Malware Analysis. There are lots of tools available for Dynamic Malware Analysis. This paper includes all the stuff Limitations of Static Malware Analysis and tools of Dynamic Malware Analysis.

Keywords: - Malware, Virus, Worm, Trojan Keylogger, Malware Analysis, Static Malware Analysis, Dynamic Malware Analysis etc.

I. Introduction

Now a day, Internet has become an essential part of the daily life of many people. On Internet many services are available and also increasing day by day. More and more people are making use of these services. The Internet has evolved from a basic communication network to an interconnected set of information sources enabling, among other things, new forms of interactions and market places for the sale of products and services. Online banking or advertising are the examples of the commercial services of the Internet. Just as in the physical world, there are people on the Internet with malevolent intents that strive to enrich themselves by taking advantage of legitimate users whenever money is involved. Malware like software of malicious intent helps these people accomplishing their goals. There are two types of malware analysis, Static and Dynamic.

II. What Is Malware?

Malware stands for malicious software, designed to damage or to infiltrate a computer system without the owner's informed consent. Viruses, Worms, Trojan, Keyloggers and Spyware are the examples of malware. In other words we can also say Software that "deliberately fulfils the harmful intent of an attacker" is commonly referred to as malicious software or malware. Terms, such as "worm", "virus", or "Trojan horse" are used to classify malware samples that exhibit similar malicious behaviour.

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 surreptitiously 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, or her bank respectively. 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.[1]

III. What Is Malware Analysis?

Malware analysis is the process of determining the purpose and functionality of a given malware sample such as a virus, worm, or Trojan horse. This process is a necessary step to be able to develop effective detection techniques for malicious code. In addition, it is an important prerequisite for the development of removal tools that can thoroughly delete malware from an infected machine. Traditionally, malware analysis has been a manual process that is tedious and time-intensive. Unfortunately, the number of samples that need to be analyzed by security vendors on a daily basis is constantly increasing [2]. 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.[1] Static and Dynamic analysis is described in detail in next sections.

IV. Static Malware Analysis

Analyzing software without executing it is called static analysis. Static analysis techniques can be applied on different representations of a program. If the source code is available, static analysis tools can help finding memory corruption flaws and prove the correctness of models for a given system.[1] Static analysis tools can also be used on the

binary representation of a program. When compiling the source code of a program into a binary executable, some information gets lost. This loss of information further complicates the task of analyzing the code.[1]

The process of inspecting a given binary without executing it is mostly conducted manually. For example, if the source code is available several interesting information, such as data structures, used functions and call graphs can be extracted. This information gets lost once the source code has been compiled into a binary executable and thus impedes further analysis. Within the malware domain typically the latter is the case, since the source code of a current malware binary is typically not available. [8]

Various techniques are used for static malware analysis. Some of those are described below.

- *File fingerprinting:* Beside examining obvious external features of the binary this includes operations on the file level such as computation of a cryptographic hash (e.g., md5) of the binary in order to distinguish it from others and to verify that it has not been modified.
- *Extraction of hard coded strings:* Software typically prints output (e.g., status- or error-messages), which end up embedded in the compiled binary as readable text. Examining these embedded strings often allows conclusions to be drawn about internals of the inspected binary.
- *File format:* By leveraging metadata of a given file format additional, useful information can be gathered. This includes the magic number on UNIX systems to determine the file type as well as dissecting information of the file format itself. For example from a Windows binary, which is typically in PE format (portable executable) a lot of information can be extracted, such as compilation time, imported and exported functions as well as strings, menus and icons.
- *AV scanning:* If the examined binary is well-known malware it is highly likely to be detected by one or more AV scanners. To use one or more AV scanner is time consuming but it becomes necessity sometimes.
- *Packer detection:* Nowadays malware is mostly distributed in an obfuscated form e.g., encrypted or compressed. This is achieved using a packer, whereas arbitrary algorithms can be used for modification. After packing the program looks much different from a static analysis perspective and its logic as well as other metadata is thus hard to recover. While there are certain unpackers, such as PEiD2, there is accordingly no generic unpacker, making this a major challenge of static malware analysis.
- *Disassembly:* The major part of static analysis is typically the disassembly of a given binary. This is conducted utilizing tools, which are capable of reversing the machine code to assembly language, such as IDA Pro. Based on the reconstructed assembly code an analyst can then inspect the program logic and thus examine its intention. Usually this process is supported by debugging tools such as OllyDbg.

The main advantage of static malware analysis is that it allows a comprehensive analysis of a given binary. That is, it can cover all possible execution paths of a malware sample. Additionally, static analysis is generally safer than dynamic analysis as the source code is not actually executed. However, it can be extremely time-consuming, cumbersome and thus requires expertise. [8]

V. Limitation Of Static Malware Analysis

Generally, the source code of malware samples is not readily available. That reduces the applicable static analysis techniques for malware analysis to those that retrieve the information from the binary representation of the malware. Analyzing binaries brings along intricate challenges. Consider, for example, that most malware attacks hosts executing instructions in the IA32 instruction set. The disassembly of such programs might result in ambiguous results if the binary employs self modifying code techniques. Additionally, malware relying on values that cannot be statically determined (e.g., current system date, indirect jump instructions) exacerbate the application of static analysis techniques. The other is that malware authors know of the limitations of static analysis methods and thus, will likely create malware instances that employ these techniques to thwart static analysis. Therefore, it is necessary to develop analysis techniques that are resilient to such modifications, and are able to reliably analyze malicious software.[1]

VI. Dynamic Malware Analysis

Executing a given malware sample within a controlled environment and monitoring its actions in order to analyze the malicious behaviour is called dynamic malware analysis. Since Dynamic Malware Analysis is performed during runtime and malware unpacks itself, dynamic malware analysis evades the restrictions of static analysis (i.e., unpacking and obfuscation issues). Thereby it is easy to see the actual behaviour of a program. Another major advantage is that it can be automated thus enabling analysis at a large scale basis. However, the main drawback is so-called dormant code: That is, unlike static analysis, dynamic analysis usually monitors only one execution path and thus suffers from incomplete code coverage. In addition there is the danger of harming third party systems, if the analysis environment is not properly isolated or restricted respectively. Furthermore, malware samples may alter their behaviour or stop executing at all once they detect to be executed within a controlled analysis environment. [8]

Mainly two basic approaches for dynamic malware analysis can be distinguished:

- *Analyzing the difference between defined points:* A given malware sample is executed for a certain period of time and afterwards the modifications made to the system are analyzed by comparison to the initial system state. In this approach, Comparison report states behaviour of malware.[8]
- *Observing runtime-behavior:* In this approach, malicious activities launched by the malicious application are monitored during runtime using a specialized tool. [8]

An example for the first approach is Truman (The Reusable Unknown Malware Analysis Net). Thereby malware is executed on a real Windows environment rather than within a Virtual Machine. During runtime Truman provides a virtual Internet for the malware to interact with. After execution the host is restarted and boots a Linux image, which then mounts the previously used Windows image in order to extract the relevant data, such as the Windows registry and a complete file list. Finally the Windows environment is reset to its initial clean state. By using a native environment Truman is able to circumvent possible anti-debugging measures of malware. However, since the result is only a snapshot of the infected system, information related to dynamic activities such as spawned processes and temporarily created files are lost. [8]

Hence observing the runtime-behavior of an application is currently the most promising approach. It is mostly conducted utilizing sandboxing. A sandbox hereby refers to a controlled runtime environment which is partitioned from the rest of the system in order to isolate the malicious process. This partitioning is typically achieved using virtualization mechanisms on a certain level. While in principle existing tools, such as chroot could be used to deploy such a controlled environment several sandbox environments dedicated to malware analysis exist implementing specialized techniques. [8]

VII. Malware Analysis Tools

Here is an overview of the existing approaches and tools that make use of the presented techniques to analyze unknown and potentially malicious software. The analysis reports generated by the tools in this section give an analyst valuable insights into actions performed by a sample. These reports lay the foundation for a fast and detailed understanding of the sample. This knowledge is necessary for developing countermeasures in a timely manner.

Anubis: The analysis of unknown binaries project is based on TTAnalyze. Anubis executes the sample under analysis in an emulated environment consisting of a Windows XP operating system running as the guest in Qemu. The analysis is performed by monitoring the invocation of Windows API functions, as well as system service calls to the Windows Native API. Additionally, the parameters passed to these functions are examined and tracked.[3]

CWSandbox: CWSandbox, created by Carsten Willems executes the sample under analysis either natively or in a virtual Windows environment. The analysis functionality is implemented by hook functions that perform the monitoring on the API level. Additionally, monitoring of the system call interface is implemented. The system is designed to capture the behavior of malicious software samples with respect to file system and registry manipulation, network communication, and operating system interaction. The virtual system is a full installation of a Win32 class operating system under which the sample under analysis is executed together with the analysis components. [4]

Norman Sandbox: The Norman Sandbox is a dynamic malware analysis solution which executes the sample in a tightly-controlled virtual environment that simulates a Windows operating system. This environment is used to simulate a host computer as well as an attached local area network and, to some extent, Internet connectivity. The core idea behind the Norman Sandbox is to replace all functionality that is required by an analyzed sample with a simulated version thereof. The simulated system thus has to provide support for operating system relevant mechanisms such as memory protection and multi-threading support. Moreover, all required APIs have to be present to give the sample the fake impression that it is running on a real system. Because the malware is executed in a simulated system, packed or obfuscated executables do not hinder the analysis itself. [5]

Norman Sandbox focuses on the detection of worms that spread via email or P2P networks, as well as viruses that try to replicate over network shares. In addition, a generic malware detection technique tries to capture other malicious software. [5]

JoeBox: During the dynamic analysis of a potentially malicious sample, Joebox [Buehlmann and Liebchen] creates a log that contains high level information of the performed actions regarding file system, registry, and system activities. Joebox is specifically designed to run on real hardware, and not to rely on any virtualization or emulation technique. The system is designed as a client server model where a single controller instance can coordinate multiple clients that are responsible for performing the analysis. Thus, it is straight forward to increase the throughput of the complete system by adding more analyzing clients to the system. All analysis data is collected by the controlling machine. [6]

VIII. Conclusion

We have learnt Malware basics ,malware analysis and techniques of analyzing malware. We have also learnt limitations of static malware analysis. After the discussion between static and malware analysis, Dynamic malware anlysis is the best way to anlyze malware samples. In this we have gone through the some tools for malware analysis. After studying malware and malware analyis techniques, sandbox enviornment is the best way to analyze dynamic malwares.

References

1. *A Survey on Automated Dynamic Malware Analysis Techniques and Tools,* http://www.seclab.tuwien.ac.at/papers/malware_survey.pdf
2. *TTAnalyze: A Tool for Analyzing Malware,* http://www.auto.tuwien.ac.at/~chris/research/doc/eicar06_ttanalyze.pdf
3. *Anubis. Analysis of unknown binaries.* http://anubis.iseclab.org
4. *Toward automated dynamic malware analysis using CWSandbox.* http://dl.acm.org/citation.cfm?id=1262675
5. *Norman SandBox Whitepaper.* http://download.norman.no/whitepapers/whitepaper Norman SandBox.pdf
6. *Joebox: a secure sandbox application for Windows to analyse the behaviour of malware.* http://www.joebox.org/

7. *The Malware Analysis Body of Knowledge By, Craig Valli and Murray Brand ,School of Computer and Information Science Edith Cowan University http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1043&context=adf*
8. *MasterThesis-MartinBrunner, Integrated Honeypot Based Malware Collection and Analysis http://www.martinbrunner.net/doc/MasterThesis-MartinBrunner.pdf*
9. *A New Generic Taxonomy on Hybrid Malware Detection Technique,*
10. *http://arxiv.org/ftp/arxiv/papers/0909/0909.4860.pdf*
11. *A Malware Instruction Set for Behavior-Based Analysis,*
12. *http://www.mlsec.org/malheur/docs/mist-tr.pdf*
13. *A Survey of Malware Detection Techniques, http://www.serc.net/system/files/SERC-TR-286.pdf*
14. *Dynamic Analysis of Malware, http://0xbadcab1e.lu/papers/analyse.pdf*
15. *Honeysand: An Open Source Tools Based Sandbox Environment for Bot Analysis and Botnet Tracking, http://research.ijcaonline.org/comnetcs/number1/comnetcs1007.pdf*
16. *Malware Analysis & its Application to Digital Forensic, http://www.enggjournals.com/ijcse/doc/IJCSE12-04-04-023.pdf*
17. *Malware Detection using Windows Api Sequence and Machine Learning , http://research.ijcaonline.org/volume43/number17/pxc3878715.pdf*

