ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

Malware Detection and Analysis

Mohd. Hamzah Khan
Department of Computer Science and Engineering
Jamia Hamdard
New Delhi, India

Ihtiram Raza Khan
Department of Computer Science and Engineering
Jamia Hamdard
New Delhi, India

Abstract: The purpose of malware analysis is to obtain and provide the information needed to rectify a network or system intrusion. Our goals will be to find out exactly what happened, and to make sure that all infected machines and files are located . When we analyse potential malware, the intended result is typically to determine what a suspected malware can do, how to detect it once it is in our network, and how to measure and contain the damage. Once we identify which files require full analysis, it's time to develop signatures to detect malware infections on our network. Malware analysis can be used to develop host-based and network signatures. This paper makes the detection and analysis of malware simpler by introducing a framework for detection of unwanted signatures. Framework makes user aware of the contents of the binary file and help them to analyze malicious executables using simple programming languages such as python. Readily scan through the otherwise complex code to derive useful structural information that may provide a valuable insight into the specific functional behaviour of the malware. Using existing tools and techniques the normal users can make their computers more secure by using python scripts.

Keywords: Malware, Framework, Static analysis, Dynamic analysis, Remote analysis, Local analysis

I. INTRODUCTION

The reason for malware examination is typically to obtain the data we need to react to a system interruption. Our objectives will regularly be to decide precisely what happened, and to guarantee that we've found every single tainted machine and records. When dissecting presumed malware, our objective will commonly be to decide precisely what a specific malware can do, how to identify it on our system, and how to remove and contain its harm[2]. When we recognize which records require full examination, it's a great opportunity to create marks to identify malware on our system. Malware examination can be utilized to create have based and arrange marks. Have based marks, or pointers, are utilized to distinguish harmful code on infected PCs. These markers regularly recognize documents made or adjusted by the malware or particular changes that it makes to the registry. Not at all like antivirus marks, malware pointers concentrate on what the malware does to a framework, not on the attributes of the malware itself, which makes them more compelling in recognizing malware that progress in shape . Arrange marks are utilized to identify malicious code by checking system activity. Arrange marks can be made without malware investigation, yet marks made with the assistance of malware analysis are as a rule much more successful, offering a higher identification rate and less false positives. In the wake of getting the marks, the last target is to make sense of precisely how the malware functions. This is frequently the most made inquiry by senior administration, who need a full clarification of a noteworthy interruption. The top to bottom methods will enable you to decide the reason and capacities of malware. Malware Analysis Systems Frequently, when performing malware investigation, we'll have just the malware executable, which won't be intelligible. With a specific end goal to comprehend it, we'll utilize an assortment of modules and traps, each noteworthy a little measure of data. we'll have to utilize an assortment of modules keeping in mind the end goal to see the full picture.

There are two principal ways to deal with malware analysis: Static and Dynamic analysis[1][5][7][9][10].

• Static Analysis includes analyzing the malware without running it. Dynamic analysis includes running the malware. Both systems are additionally sorted as essential or as progressed. Static analysis comprises of inspecting the executable document without survey of the genuine guidelines. Static analysis can affirm whether a document is harmful, give data about its usefulness, and then give data that will enable you to create basic system marks. Static analysis is clear and can be speedy, however it's generally ineffective against refined malware, and it can miss imperative practices[1][2].

• Dynamic analysis methods include running the malware and watching its conduct on the framework so as to terminate the contamination, deliver successful marks, or both. Before you can run malware securely, we should set up a situation that will enable us to concentrate the running malware without danger of harm to our framework or system[5]. When performing malware examination, by using identifying features that can be found that you can regularly accelerate your investigation by making instructed surmises about what the malware is attempting to do. Obviously, you'll have the capacity to improve security to the event that you know against the sorts of things that malware generally does[1][2][4][5].

II. TYPES OF MALWARE

Classes that most malware falls into are[3]:

• Backdoor malicious code that introduces itself onto a PC to permit the intruder get indirect access to interface with the PC with practically zero confirmation and execute commands on the PC.

• Botnet Like a secondary passage, permits the attacker access to the framework, however all PCs tainted with the same botnet get similar guidelines from a solitary in-charge and control server[3][4].

• Data Theft Malware, It gathers data from a victim PC and sends it to the intruder. Cases incorporate sniffers, watchword hash grabbers, and keyloggers. This malware is commonly used to access online records, for example, email or web based managing an account[3][4].

• Rootkit Code intended to hide the presence of other code. Rootkits are normally combined with other malware, for example, a secondary passage, to enable remote access to the intuder and make the code troublesome to identify[3][4].

• Scareware Malware intended to panic a contaminated client into purchasing something. It more often than not has a UI that makes it resemble an antivirus or other security program. It illuminates clients that there is vindictive code on their framework and that the best way to dispose of it is to purchase their "product," when truly, the product it's offering does just expel the scareware[3][4].

• Spam-sending Malware that contaminates a client's machine and afterward utilizes that machine to send spam. This malware creates income for assailants by enabling them to offer spam-sending softwares[3][4][8].

• Worm or Virus that can duplicate itself and taint extra PCs. Malware has numerous classes. For instance, a program may have a keylogger that gathers passwords and a worm part that sends it to the intruder[3][4][9].

• Malware can further be grouped on whether the aggressor's goal is mass or focused on. Mass malware, for example, scareware, adopts the shotgun strategy and is intended to influence however many machines as would be prudent. Of the two destinations, it's the most widely recognized, and is generally the less modern and simpler to identify and guard against in light of the fact that security programming targets it. Directed malware, similar to a stand-out indirect access, is customized to a particular target. Directed malware is a greater danger to systems than mass malware, in light of the fact that it is not known and your security items presumably won't shield you from it. Without thorough examination of focused malware, it is almost difficult to secure your system against that malware[2][3].

III. MALWARE DETECTION TECHNIQUES

The paper is based on the static analysis technique of malware analysis. The analysis has been done by:

A. Local Analysis

The analysis is being done on the local machine, i.e. on the users system. Modules further associated with local analysis are:

- Suspicious Api
- Anti-virtual machine
- Anti- debugger
- Url analysis
- File information
- String analysis
- Packer information

B. Remote Analysis

The executables are checked via online antivirus with the help of API'S provided by antivirus and automating them with python scripts[6][9].

IV. EXISTING SYSTEMS

The existing systems consist of security frameworks like Metasploit, Backtrack etc, in which several software based attacks were performed and hence these are not safe for performing pentesting. Existing systems are vulnerable to malware intrusion and need more security measures to secure it.

V. PROPOSED SOLUTION

Upon application of this paper to develope a framework the user can use the framework where the user can directly run the code or they can also use the GUI provided to directly perform the penetration testing in their system or network[9].

The framework has been divided into various modules which analyze the various aspect of the malware. Inspection of each is as follows.

A. Remote analysis

During the Remote Analysis technique, the malicious executable is being sent to various anti viruses. The executable is then checked using the API'S provided by each antivirus. Thus, the result achieved by remote analysis is to check whether a particular executable is malicious or not using a mass scrutiny via various anti viruses. If all the anti viruses in the remote analysis show the executable is malicious then it can be seen as a executable of high malicious intensity as different anti viruses use different analysis technique. Using the result of this analysis, the analyst can decide whether the malware is to be dissected using static analysis or dynamic analysis[1][2][8][9].

B. Local analysis

This analysis is used to analyze the following modules in various aspects to find malware:

• SUSPICIOUS: Checks for suspicious API'S, Functions and dll's. Executable is checked against list of suspicious API'S, if the list matches with the import table of executable, alert is generated. The suspicious module can be implemented using the Pefile module of python 2.7 which is used to extract the metadata out of the executable. Thus all the suspicious functions and API's can be extracted out of the malicious executable. Some of the suspicious API functions can be: accept', 'AddCredentials', 'bind', 'CertDeleteCertificateFromStore', 'CheckRemoteDebuggerPresent', 'closesocket', 'connect', 'ConnectNamedPipe', 'CopyFile' etc. The suspicious API'S and other suspicious functions are matched with the coded functions and the alert is generated.

• Anti-Virtual Machine: Malware can Hide or alter its functionality attempt to breakout by Looking for VM artifacts in processes, file system, or registry Look for VM artifacts in memory ,Look for VM-specific virtual hardware, Look for VM-specific processor instructions and capabilities. Anti-VM techniques are most commonly found in malware that is widely deployed, such as bots, scareware, and spyware ,mostly because honeypots often use virtual machines and because this malware typically targets the average user's machine, which is unlikely to be running a virtual machine. The ANTI-VM module is implemented using the RED Pill technique of VM detection. The instructions are detected using Regular expression module in python, thus regular expression module is imported using IMPORT RE.

• Anti-debugger: Malware can be made anti-debug so that it cannot be debug and hence making it hard to dissect it and its working. Malware uses a variety of techniques to scan for indications that a debugger is attached, including using the Windows API, manually checking memory structure for debugging artifacts, and searching the system for residue left by a debugger. Debugger detection is the most common way that malware performs anti-debugging.

• URL Analysis: Malwares are analyzed of suspicious connection with: Dll, Strings, functions. Malwares sometimes connects to certain websites and servers to send information about the compromised system. URL analysis analyze which function and API'S used in the executable is establishing a connection.
This is achieved using URLLIB module of python and PEFILE format to extract the executable.

• Strings analysis: This is a really very important step for any forensics investigator or reverse engineers who is dissecting the malware. This is not rocket science. String analysis is nothing but analyzing the sequence of all those characters that are written in the code. This may include print messages, URLs, and comments and it may reveal information about the website, the malware's functionality, the author's name/nickname, and more. ." A program contains strings if it prints a message, connects to a URL, or copies a file to a specific

location. Scanning every string can be a way to get a idea about the functionality of a program. For example, if the program accesses a URL, then you will see the URL accessed stored as a string in the program

• Packing analysis : Malware writers often use packing or obfuscation to make their files more difficult to detect or analyze. Obfuscated programs are whose real purpose the malware author has attempted to hide. Packed and obfuscated code will often include at least the functions LoadLibrary and GetProcAddress, which are used to load and gain access to additional functions. All packer programs take a executable file as a input and produce a packed executable file as output which is much more harder to re-engineer and identify. Most packers use a compression algorithm to compress the original executable. A packer designed to make the file difficult to analyze may encrypt the original executable and employ anti-reverse-engineering techniques, such as anti-disassembly, anti-debugging, or anti-VM. Packers can pack the entire executable, including all data and the resource section, or pack only the code and data sections. To maintain the functionality of the original program, a packing program needs to store the program's import information. The information can be stored in any format, and there are several common strategies.

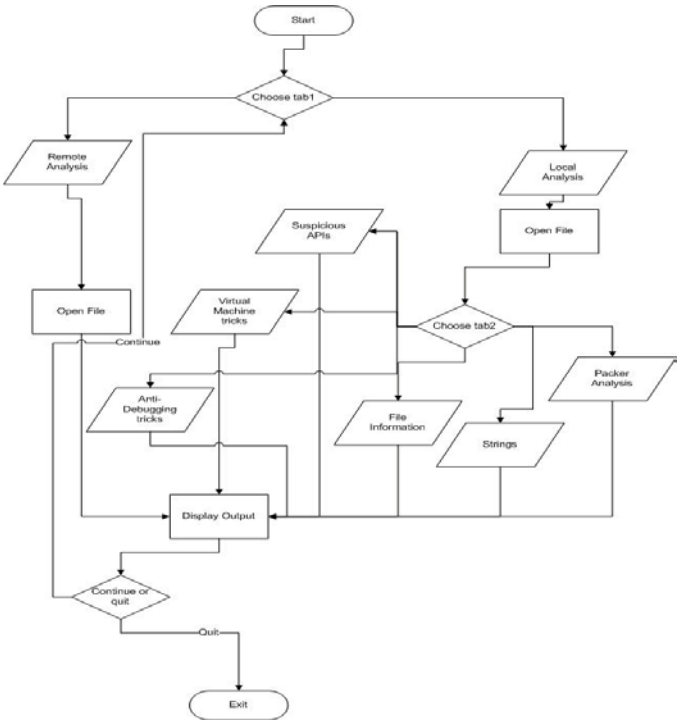

Figure 1. Design flow of framework.

VI. CONCLUSION

Today the threat of malware has increased many times than before, with the ease of resource availability and technical know how new malwares are emerging every day with intention of not being detected. In order to protect important information, security and privacy there a need of a technique that helps in malware analysis on hand and keeps the user safe from further similar attempts by providing necessary security.

In this paper we have described the types of malware and after discussing both techniques of malware analysis it can be concluded that upon development of a framework using the above research a successfull technique can be developed that can prevent against malware attacks of many different types. Remote analysis and Local analysis both modules are important in their own respect as they broaden the users ability to mitigate the malware attack themselves .The framework will provide user safety from anti-debugger , packaged /obfuscated malware that tries to conceal its true functionality. Although with the latest development in malware threats such as polymorphic, metamorphic malware etc, this technique does not completely safeguard against those threats . More work is further to be done in this direction to protect the users from such malware. Machine learning is being used to further combat this malware threat by learning the identifying features of malware and deploying them to further identify similar malware before they can cause damage.

VII. REFERENCES

[1] Syarif Yusirwan S, Yudi Prayudi, Imam Riadi,2015,Implementation of Malware Analysis using Static and Dynamic Analysis Method, International Journal of Computer Applications ,Volume 117 – No. 6, May 2015.

[2] Savan Gadhiya, Kaushal Bhavsar,2013, Techniques for Malware Analysis, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 4, April 2013.

[3] Dolly Uppal,Vishakha Mehra, Vinod Verma,2014, Basic survey on Malware Analysis, Tools andTechniques, International Journal on Computational Sciences & Applications (IJCSA) ,Vol.4 No.1, February 2014.

[4] M. Asha Jerlin, C. Jayakumar,2015, A Dynamic Malware Analysis for Windows Platform - A Survey, Indian Journal of Science and Technology, October 2015.

[5] Navroop Kaur, Amit Kumar Bindal, 2016, A Complete Dynamic Malware Analysis, International Journal of Computer Applications (0975 – 8887),Volume 135 – No.4, February 2016.

[6] Ekta Gandotra, Divya Bansal, Sanjeev Sofat,2014, Malware Analysis and Classification,A Survey, Journal of Information Security, 2014, 5, 56-64.

[7] Nirmal Singh, Dr. Sawtantar Singh Khurmi,2015, Malware Analysis, Clustering and Classification: A Literature Review, International Journal of Computer Science And Technology Vol. 6, Issue 1 Spl- 1 Jan-March 2015.

[8] M. Wagner,, F. Fischer, R. Luh, A. Haberson, A. Rind, D. A. Keim, W. Aigner,2015, A Survey of Visualization Systems for Malware Analysis, Eurographics Conference on Visualization (EuroVis) (2015).

[9] Gray Hat Python – Python Programming for Hackers and Reverse Engineers, 2009 by Justin Seitz.