**Malware Analysis**

Malwares are the malicious program that can damage or exploit the computer or network and compromise its confidential information. An analysis is performed to detect and observe the malicious executable and to check its behavior by running it on virtual or real environment. This analysis is known as malware analysis.

**How is it possible?**

First of all we have to identify the malicious file and it can be possible through host based and network base signature. Host base signature are used to detect the malware on the victim’s computer. Indicators also identify which file is created, modified and deleted on the computer and what infections are produced by the malware. Whether the Network Base Signatures are used to detect the malware code by monitoring the network traffic. After signature a hot question arise from higher authorities what this malicious executable does and what was the purpose of it in our network or computer.

**Hot Question from higher authorities:** What was the purpose of this executable in our network or computer? How can we defend and protect our environment?

To answer these questions we have to understand the following terms of malware analysis

* Static analysis
* Dynamic analysis

Compiled Result = Report of Static analyst + Report of Dynamic Analyst

**Static analysis:**

Malware are actually exe file and only the code instruction can tell about what the executable have to perform or to produce signatures for their detection so the techniques that are analyzing the exe in static mode (without running it) is called static analysis of malware.

Types of static analysis:

* Basic static analysis
* Advance static analysis

**Basic static analysis** examine the exe file without viewing its instruction code. It can tell about the maliciousness of the file and its functionality sometime it allow us to produce simple network signature.

**Advance static analysis** consists of reverse engineering by loading the malicious exe into disassembler and looking the program instructions. This assembly code can tell us that what this malicious exe have inside it and statistically these analyst can make a report about the expected functionality of this exe.

**Dynamic Analysis:**

In most cases malware writer deceives the malware analyst because the behavior of malicious exe is different from which the advance static analyst expect. For conducting complete analysis we perform dynamic analysis which is only possible by running the malicious exe. Then analysis generates a report about the actual behavior of program in a running condition. This type of analysis is known as dynamic analysis.

Type of dynamic analysis:

* Basic dynamic analysis
* Advance dynamic analysis

**Basic dynamic analysis** occurwhen we want to observe the behavior of malware by running it on virtual environment. So that we can remove the infection. This type of analysis is called basic dynamic analysis.

**Advance dynamic** use debugger to examine the internal state of malware in running condition. This technique is useful when we are trying to obtain information that is difficult to gather from other techniques.

**Basic Static Analysis in Detail**

For performing basic static analysis various questions arises in our mind as follows

**What techniques will we use to perform basic static analysis?**

* Antivirus scanning
* Hashing
* String finding

**Antivirus scanning** is a technique in which malicious file is detected by the antivirus software. Various questions arise in mind.

* Is antivirus reliable to detect malware?
* Are such malware exists which are not detected by antivirus?
* By which techniques the malicious program become undetectable?

In static analysis antivirus are considered a reliable source for the detection of malicious code but some sophisticated malwares exists which can defeat antivirus and is not detected by the security provider software. These software’s rely on the database of identifiable piece of known suspicious code (file signature) and behavioral pattern matching (heuristic search). The malware writers can easily modify their code by changing the program signature which is not present in the virus scanning program’s database. That’s why malware become undetectable.

**Hashing** technique is used to identify the malware. It use hash calculating software that takes input and produce a one way unique output known as hash. Malicious file is put into MD5 or HashCalc software that produce a unique value of hash of this file. To compare this hash value with other hashes if hashes match then the other file will be malicious otherwise not.

**String finding** is a technique in which we can check the strings present in malicious code exe. We can guess about its functionality by seeing the strings most malware are link to website so http or URL info can tell us about its functionality and most try to find the vulnerability in OS. So the .dll files can help us to check these files. For doing this experiment I download string.exe file from <https://technet.microsoft.com/en-us/sysinternals/bb897439> this link use cmd, after giving the path I type strings and the name of malicious exe. The strings are produced in cmd.

**By which technique the malicious exe become difficult in reverse engineering?**

**Packed or obfuscated programs** are those programs whose execution is hidden that contains packed files having the malicious program code in compress form. When a packed file runs the wrapper program also run that decompress the packed file and then run the unpacked file (containing the actual source code). This packed behavior creates difficulty for the malware analyst because files are packed and are linked with the other libraries, if string finding techniques is applied here then the packed file show only the few string which is not enough for processing.

By studying the behavior of packed or obfuscated files following questions arises in our mind as

* **Is packed file detectable?**
* **Can a packed file be unpacked?**
* **What format the packed file have?**

Yes, packed files are detectable and can be unpacked. Various tools are available to detect the packer like PEiD. UPX tool is used for unpacking the program. The PE (Portable Executable) file format is used by the windows executables, object code and DLLs. PE file format is the data structure that contains the information necessary for the windows OS loader to manage the wrapped exe code. PE files begins with header that contains the information about the code, the type of application, required library functions and space requirement.

**Why Linked Library and functions are important in the study of malware analysis?**

If malware analyst know about what functions and libraries are included in the malware exe then he can easily tell about that what functionality the malware will perform. Imports are functions used by one program that are actually stored in different program, such a code library contain functionality common to many programs. Code libraries can be connected to the main exe by linking.

* **How the code libraries are linked with each other?**

The code libraries can be linked statically, dynamically or at runtime. Knowing how the library code is linked is critical to our understanding of malware because the information that we get from PE header file depends on how the library code has been linked.

**Static linking:** when a library is statically linked to an executable, all code from that library is copied to an executable, which makes the exe grow in size. When analyzing code it is difficult to differentiate between statically linked code and the exe own code, because nothing in the PE header indicate that the file contains linked code.

**Runtime linking** is commonly used in malware, especially when it is packed or obfuscated. Exe that use runtime linking connect to library only when that function is needed, not at program start, or as with dynamically linked programs. Example GetProcAdress LoadLibrary.

**Dynamic linking:** when libraries are linked dynamically, the OS search for necessary libraries when the program is loaded. When the program called from the linked library function, that function executes within the library. Dynamic linked functions can be explored with the help of dependency Walker tool.

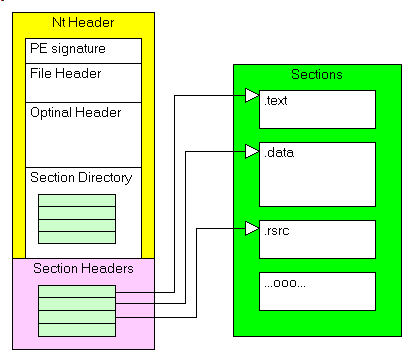
**Types of malware:**

* Backdoor
* Botnets
* Downloader
* Launcher
* Rootkit
* Scareware
* Spam-sending malware
* Worm or virus
* Information stealing malware

**PE Header**

PE header contains the meta data and the information of all imports and exports functions, the header is divided into the following sections

* .text
* .data
* .rdata
* .rsrc



**.text** section contains the instructions that the CPU executes and all executable code.

**.data** sections contains all global data of the program, which is accessible from anywhere in the program. Local data is stored in this section.

**.rdata** section contains the import and export information **.idata** is used to import the function and **.edata** is used to export the function.

**.rsrc** section stores the resources needed by the executable.

We have examined the PE header by the PEview, PEBrowser, PEExplorer and Dependency walker tools.