ITGY400 – REVERSE ENGINEERING AND MALWARE ANALYSIS

Topic – Malware Analysis

Malware Analysis

- Malware analysis is the study of malware's behavior.
- The objective of malware analysis is to understand the working of malware and how to detect and eliminate it.
- It involves analyzing the suspect binary in a safe environment to identify its characteristics and functionalities so that better defenses can be built to protect an organization's network.

Importance Malware Analysis

- The primary motive behind performing malware analysis is to extract information from the malware sample, which can help in responding to a malware incident.
- The following are some of the importance of perform malware analysis:
  - To determine the nature and purpose of the malware.

  For example, it can help you determine whether malware is an information stealer, HTTP bot, spam bot and so on.

  - To gain an understanding of how the system was compromised and its impact.

Importance Malware Analysis

- To identify the network indicators associated with the malware, which can then be used to detect similar infections using network monitoring.

For example, during the analysis, if you determine that a malware contacts a particular domain/IP address, then you can use this domain/IP address to create a signature and monitor the network traffic to identify all the hosts contacting that domain/IP address.

- To extract host-based indicators such as filenames, and registry keys, which, in turn, can be used to determine similar infection using host-based monitoring.

For instance, if you learn that a malware creates a registry key, you can use this registry key as an indicator to create a signature, or scan your network to identify the hosts that have the same registry key.

Importance Malware Analysis

- To determine the attacker's intention and motive.

For instance, during your analysis, if you find that the malware is stealing banking credentials, then you can deduce that the motive of the attacker is monetary gain.

Types of Malware Analysis

- To understand the working and the characteristics of malware and to assess its impact on the system, you will often use different analysis techniques.
- The following is the classification of these analysis techniques:
  - Static analysis
  - Dynamic analysis (Behavioral Analysis)
  - Code analysis
  - Memory analysis (Memory forensics)

Types of Malware Analysis

- **Static Analysis:**
- This is the process of analyzing a binary without executing it.
- It is easiest to perform and allows you to extract the metadata associated with the suspect binary.
- Static analysis might not reveal all the required information, but it can sometimes provide interesting information that helps in determining where to focus your subsequent analysis efforts.

Types of Malware Analysis

- **Dynamic analysis (Behavioral Analysis):**
- This is the process of executing the suspect binary in an isolated environment and monitoring its behavior.
- This analysis technique is easy to perform and gives valuable insights into the activity of the binary during its execution.
- This analysis technique is useful but does not reveal all the functionalities of the hostile program.

Types of Malware Analysis

- **Code analysis:**
- It is an advanced technique that focuses on analyzing the code to understand the inner workings of the binary.
- This technique reveals information that is not possible to determine just from static and dynamic analysis.
- Code analysis is further divided into
  - Static code analysis - involves disassembling the suspect binary and looking at the code to
  - understand the program's behaviour
  - Dynamic code analysis - involves debugging the suspect binary in a controlled manner to understand its functionality.

Types of Malware Analysis

- **Code analysis:**
- Code analysis requires an understanding of the programming language and operating system concepts.

Types of Malware Analysis

- **Memory analysis (Memory forensics):**
- This is the technique of analyzing the computer's RAM for forensic artifacts.
- It is typically a forensic technique, but integrating it into your malware analysis will assist in gaining an understanding of the malware's behavior after infection.
- Memory analysis is especially useful to determine the stealth and evasive capabilities of the malware.

Static Analysis

- Static Analysis is the technique of analyzing the suspect file without executing it.
- It is an initial analysis method that involves extracting useful information from the suspect binary to make an informed decision on how to classify or analyze it and where to focus your subsequent analysis efforts.

Static Analysis - Determining the File Type

- During the analysis, determining the file type of a suspect binary will help to identify the malware's target operating system (Windows, Linux, and so on) and architecture (32-bit or 64-bit platforms).
- For example, if the suspect binary has a file type of Portable Executable (PE), which is the file format for Windows executable files (.exe , .dll , .sys , .drv , .com , .ocx , and so on), then you can deduce that the file is designed to target the Windows operating system.

Static Analysis - Determining the File Type

- Most Windows-based malware are executable files ending with extensions such as .exe , .dll , .sys , and so on.
- But relying on file extensions alone is not recommended.
- File extension is not the sole indicator of file type. Attackers use different tricks to hide their file by modifying the file extension and changing its appearance to trick users into executing it.
- Instead of relying on file extension, File signature can be used to determine the file type.

Static Analysis - Determining the File Type

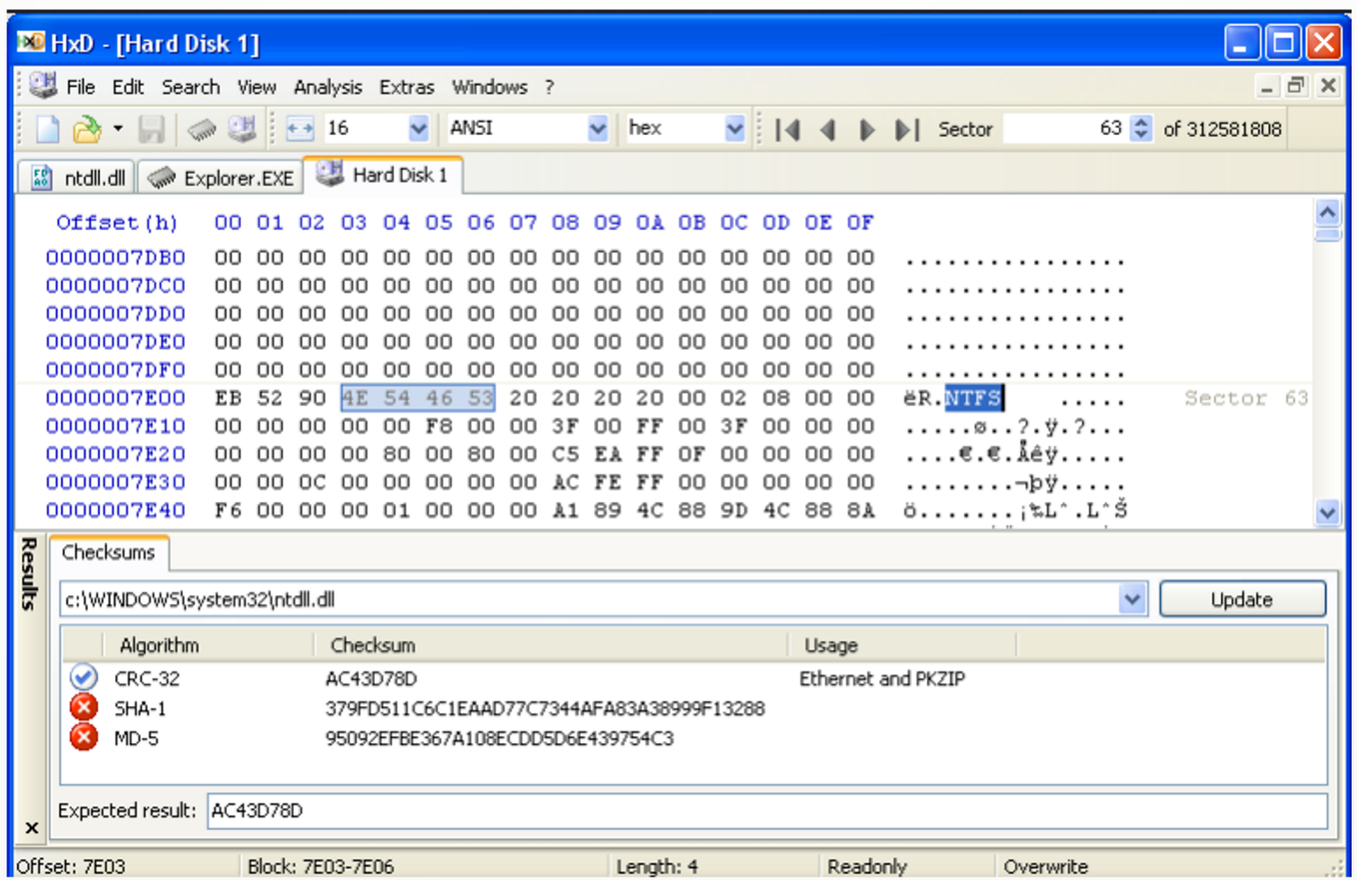
- A file signature is a unique sequence of bytes that is written to the file's header.
- Different files have different signatures, which can be used to identify the type of file.
- The Windows executable files, also called PE files (such as the files ending with .exe , .dll , .com , .drv , .sys , and so on), have a file signature of MZ or hexadecimal characters 4D 5A in the first two bytes of the file.

Static Analysis - Determining the File Type

- **Identifying File Type Using Manual Method**
- The manual method of determining the file type is to look for the file signature by opening it in a hex editor.
- A hex editor is a tool that allows an examiner to inspect each byte of the file; most hex editors provide many functionalities that help in the analysis of a file.
- The following screenshot shows the file signature of MZ in the first two bytes when an executable file is opened with the HxD hex editor (https:/ / mh- nexus. de/ en/ hxd/):

Static Analysis - Determining the File Type

- **Identifying File Type Using Manual Method**



Static Analysis - Determining the File Type

- **Identifying File Type Using Tools**
- The other convenient method of determining the file type is to use file identification tools.
- On Linux systems, this can be achieved using the file utility. In the following example, the file command was run on two different files.
- From the output, it can be seen that even though the first file does not have any extension, it is detected as a 32-bit executable file (PE32) and the second file is a 64-bit (PE32+) executable:

Static Analysis - Determining the File Type

- **Identifying File Type Using Tools**

```
$ file mini
mini: PE32 executable (GUI) Intel 80386, for MS Windows

$ file notepad.exe
notepad.exe: PE32+ executable (GUI) x86-64, for MS Windows
```

Static Analysis - Determining the File Type

- **Identifying File Type Using Tools**
- On Windows, CFF Explorer, part of Explorer Suite (http:/ / www. ntcore. com/ exsuite. php), can be used to determine the file type; it is not just limited to determining file type.
- It is also a great tool for inspecting executable files (both 32-bit and 64-bit) and allows you to examine the PE internal structure, modify fields, and extract resources.

Static Analysis - Determining the File Type

- **Determining File Type Using Python**
- In Python, the python-magic module can be used to determine the file type.
- On Windows, to install the python-magic module, you can follow the procedure mentioned at https:/ /github.com/ahupp/ python- magic .

Static Analysis - Fingerprinting the Malware

- Fingerprinting involves generating the cryptographic hash values for the suspect binary based on its file content.
- The cryptographic hashing algorithms such as MD5, SHA1 or SHA256 are considered the de facto standard for generating file hashes for the malware specimens.
- The following list outlines the use of cryptographic hashes:

Static Analysis - Fingerprinting the Malware

- Identifying a malware specimen based on filename is ineffective because the same malware sample can use different filenames, but the cryptographic hash that is calculated based on the file content will remain the same. Hence, a cryptographic hash for your suspect file serves as a unique identifier throughout the course of analysis
- During dynamic analysis, when malware is executed, it can copy itself to a different location or drop another piece of malware. Having the cryptographic hash of the sample can help in identifying whether the newly dropped/copied sample is the same as the original sample or a different one. This information can assist you in deciding whether the analysis needs to be performed on a single sample or multiple samples.

Static Analysis - Fingerprinting the Malware

- File hash is frequently used as an indicator to share with other security researchers to help them identify the sample.
- File hash can be used to determine whether the sample has been previously detected by searching online or searching the database of multi Anti-virus scanning service like VirusTotal.

Static Analysis - Multiple Anti-Virus Scanning

- Scanning the suspect binary with multiple anti-virus scanners helps in determining whether malicious code signatures exist for the suspect file.
- The signature name for a particular file can provide additional information about the file and its capabilities.
- By visiting the respective antivirus vendor websites or searching for the signature in search engines, you can yield further details about the suspect file.
- Such information can help in your subsequent investigation and can reduce the analysis time.

Static Analysis - Extracting Strings

- Strings are ASCII and Unicode-printable sequences of characters embedded within a file.
- Extracting strings can give clues about the program functionality and indicators associated with a suspect binary.
- Strings extracted from the binary can contain references to filenames, URLs, domain names, IP addresses, attack commands, registry keys, and so on.
- Although strings do not give a clear picture of the purpose and capability of a file, they can give a hint about what malware is capable of doing.

Static Analysis - Extracting Strings

- To extract strings from a suspect binary, you can use the strings utility on Linux systems.
- The strings command, by default, extracts the ASCII strings that are at least four characters long.
    - String -a filename.extension
- On Windows, pestudio (https:/ /www.winitor.com) is a handy tool that displays both ASCII and Unicode strings.

Static Analysis - Determining File Obfuscation

- Even though string extraction is an excellent technique to harvest valuable information, often malware authors obfuscate or armor their malware binary.
- Obfuscation is used by malware authors to protect the inner workings of the malware from security researchers, malware analysts, and reverse engineers.
- These obfuscation techniques make it difficult to detect/analyze the binary; extracting the strings from such binary results in very fewer strings, and most of the strings are obscured (few).
- Malware authors often use programs such as Packers and Cryptors to obfuscate their file to evade detection from security products such as anti-virus and to thwart analysis.

Static Analysis - Determining File Obfuscation

- A Packer is a program that takes the executable as input, and it uses compression to obfuscate the executable's content.
- This obfuscated content is then stored within the structure of a new executable file; the result is a new executable file (packed program) with obfuscated content on the disk.
- Upon execution of the packed program, it executes a decompression routine, which extracts the original binary in memory during runtime and triggers the execution.

Static Analysis - Determining File Obfuscation

- A Cryptor is similar to a Packer, but instead of using compression, it uses encryption to obfuscate the executable's content, and the encrypted content is stored in the new executable file.
- Upon execution of the encrypted program, it runs a decryption routine to extract the original binary in the memory and then triggers the execution.
- Most legitimate executables do not obfuscate content, but some executables may do it to prevent others from examining their code. When you come across a sample that is packed, there is a high chance of it being malicious.
- To detect packers on Windows, you can use a freeware tool such as Exeinfo PE

Static Analysis - Determining File Obfuscation

- A Cryptor is similar to a Packer, but instead of using compression, it uses encryption to obfuscate the executable's content, and the encrypted content is stored in the new executable file.
- Upon execution of the encrypted program, it runs a decryption routine to extract the original binary in the memory and then triggers the execution.
- Most legitimate executables do not obfuscate content, but some executables may do it to prevent others from examining their code. When you come across a sample that is packed, there is a high chance of it being malicious.
- To detect packers on Windows, you can use a freeware tool such as Exeinfo PE

Static Analysis - Comparing And Classifying The Malware

- Comparing the suspect binary with previously analyzed samples or the samples stored in a public or private repository can give an understanding of the malware family, its characteristics, and the similarity with the previously analyzed samples.
- The following are some of the techniques that can help in comparing and classifying the suspect binary:
  - Classifying Malware Using Fuzzy Hashing
  - Classifying Malware Using Import Hash
  - Classifying Malware Using Section Hash

Static Analysis - Comparing And Classifying The Malware

- **Classifying Malware Using Fuzzy Hashing**
- Fuzzy hashing is a great method to compare files for similarity. ssdeep (http:/ /ssdeep.sourceforge. net) is a useful tool to generate the fuzzy hash for a sample, and it also helps in determining percentage similarity between the samples.
- This technique is useful in comparing a suspect binary with the samples in a repository to identify the samples that are similar; this can help in identifying the samples that belong to the same malware family or the same actor group.

Static Analysis - Comparing And Classifying The Malware

- **Classifying Malware Using Import Hash**
- Import Hashing is another technique that can be used to identify related samples and the samples used by the same threat actor groups.
- Import hash (or imphash) is a technique in which hash values are calculated based on the library/imported function (API) names and their particular order within the executable. If the files were compiled from the same source and in the same manner, those files would tend to have the same imphash value.
- During your malware investigation, if you come across samples that have the same imphash values, it means that they have the same import address table and are probably related

Static Analysis - Comparing And Classifying The Malware

- **Classifying Malware Using Section Hash**
- Similar to import hashing, section hashing can also help in identifying related samples.
- When an executable is loaded in pestudio, it calculates the MD5 of each section (.text , .data , .rdata , and so on.).

Dynamic Analysis

- Dynamic analysis (behavioral analysis) involves analyzing a sample by executing it in an isolated environment and monitoring its activities, interaction, and effect on the system.
- When performing dynamic analysis, you will be executing the malware specimen, so you need to have a safe and secure lab environment to prevent your production system from being infected.
- During dynamic analysis, when the malware is executed, you will carry out various monitoring activities.
- The objective is to gather real-time data related to malware behavior and its the impact on the system. The following list outlines different types of monitoring carried out during dynamic analysis:

Dynamic Analysis

- The following list outlines different types of monitoring carried out during dynamic analysis:
  - **Process monitoring**: Involves monitoring the process activity and examining the properties of the result process during malware execution.
  - **File system monitoring:** Includes monitoring the real-time file system activity during malware execution.
  - **Registry monitoring:** Involves monitoring the registry keys accessed/modified and registry data that is being read/written by the malicious binary.
  - **Network monitoring:** Involves monitoring the live traffic to and from the system during malware execution.

Dynamic Analysis Tools

- **Process Hacker**
- Process Hacker is an open source, multipurpose tool that helps in monitoring system resources.
- It is a great tool for examining the processes running on the system and to inspect the process attributes.
- It can also be used to explore services, network connections, disk activity, and so on.
- Once the malware specimen is executed, this tool can help you identify the newly created malware process (its process name and process ID), and by right-clicking on a process name and selecting Properties, you will be able to examine various process attributes.
- You can also used it to terminate a process.

Dynamic Analysis Tools

- **Process Monitor**
- Process Monitor is an advanced monitoring tool that shows the real-time interaction of the processes with the filesystem, registry, and process/thread activity.
- When you run this tool (run as Administrator), you will immediately notice that it captures all the system events, as shown in the following screenshot.
- To stop capturing the events, you can press Ctrl + E, and to clear all the events you can press Ctrl+ X.

Dynamic Analysis Tools

- **Noriben**
- Noriben is a Python script that works in conjunction with Process Monitor and helps in collecting, analyzing, and reporting runtime indicators of the malware.
- The advantage of using Noriben is that it comes with pre-defined filters that assist in reducing noise and allow you to focus on the malware-related events.

Dynamic Analysis Tools

- **Wireshark**
- Wireshark is a packet sniffer that allows you to capture the network traffic.
- When the malware is executed, you will want to capture the network traffic generated as a result of running the malware; this will help you understand the communication channel used by the malware and will also help in determining network-based indicators

Dynamic Analysis Steps

- The following list outlines the steps involved in the dynamic analysis:
  - **Reverting to the clean snapshot**: This includes reverting your virtual machines to a clean state.
  - **Running the monitoring/dynamic analysis tools:** In this step, you will run the monitoring tools before executing the malware specimen. To get the most out of the monitoring tools you need to run them with administrator privileges
  - **Executing the malware specimen:** In this step, you will run the malware sample with administrator privileges.
  - **Stopping the monitoring tools:** This involves terminating the monitoring tools after the malware binary is executed for a specified time.

Dynamic Analysis Steps

- The following list outlines the steps involved in the dynamic analysis:
  - **Analyzing the results:** This involves collecting the data/reports from the monitoring tools and analyzing them to determine the malware's behavior and functionality.