Glossary

Intro to Malware Analysis and Reverse Engineering

Created By: Pratyay Milind, Teaching Assistant

1. **Malware -** It is any software intentionally designed to cause damage to a computer, server, client, or computer network (by contrast, software that causes unintentional harm due to some deficiency is typically described as a software bug).
2. **Malware Analysis** - It 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.
3. **Reverse Engineering** - Also called back engineering, is the process by which a man-made object is deconstructed to reveal its designs, architecture, or to extract knowledge from the object.
4. **Workstation** – It is a special computer designed for technical or scientific applications.
5. **Dynamic Analysis** - It is the analysis of computer software that is performed by executing programs on a real or virtual processor.
6. **Static Analysis** – It is the analysis of computer software that is performed without actually executing programs, in contrast with dynamic analysis, which is analysis performed on programs while they are executing.
7. **Virtual Machine (VM)** – It is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.
8. **Indicators of Compromise (IoC)** – It is an artifact observed on a network or in an operating system that, with high confidence, indicates a computer intrusion.
9. **Application Programming Interface (API) -** It is a computing interface which defines interactions between multiple software intermediaries.
10. **Snapshot** – It is the state of a system at a particular point in time.

11. **YARA** – It is the name of a tool primarily used in malware research and detection. It provides a rule-based approach to create descriptions of malware families based on textual or binary patterns.
12. **Sandbox** – It is a security mechanism for separating running programs, usually in an effort to mitigate system failures or software vulnerabilities from spreading.
13. **Debugger** – It is a computer program used to test and debug other programs. The main use of a debugger is to run the target program under controlled conditions that permit the programmer to track its operations in progress and monitor changes in computer resources (most often memory areas used by the target program or the computer's operating system) that may indicate malfunctioning code.
14. **Disassembler** – It is a computer program that translates machine language into assembly language—the inverse operation to that of an assembler.
15. **Domain Generation Algorithms (DGA)** – DGA are algorithms seen in various families of malware that are used to periodically generate a large number of domain names that can be used as rendezvous points with their command and control servers.
16. **Assembly Language** – It is any low-level programming language in which there is a very strong correspondence between the instructions in the language and the architecture's machine code instructions. Because assembly depends on the machine code instructions, every assembler has its own assembly language which is designed for exactly one specific computer architecture. Assembly language may also be called symbolic machine code.
17. **Parsers** – It is a software component that takes input data (frequently text) and builds a data structure – often some kind of parse tree, abstract syntax tree or other hierarchical structure, giving a structural representation of the input while checking for correct syntax.
18. **Portable Executables (PE)** – It is a file format for executables, object code, DLLs and others used in 32-bit and 64-bit versions of Windows operating systems.
19. **Packers** – It is a file binder that has the ability to make its "signature" mutate over time, so it is more difficult to detect and remove. It is commonly used by hackers to insert Malware into packages which contain the user content.
20. **Integrity Checking** – It is an internal control or process that performs the act of validating the integrity of operating system and application software files using a verification method between the current file state and a known, good baseline.

References

- https://en.wikipedia.org/
  - https://en.wikipedia.org/wiki/Malware_analysis
  - https://en.wikipedia.org/wiki/Malware
  - https://en.wikipedia.org/wiki/Portable_Executable
  - https://en.wikipedia.org/wiki/Assembly_language
  - https://en.wikipedia.org/wiki/Parsing
  - https://en.wikipedia.org/wiki/Domain_generation_algorithm
  - https://en.wikipedia.org/wiki/Disassembler
  - https://en.wikipedia.org/wiki/Debugger
  - https://en.wikipedia.org/wiki/Sandbox_(computer_security)
  - https://en.wikipedia.org/wiki/YARA
  - https://en.wikipedia.org/wiki/Snapshot_(computer_storage)
  - https://en.wikipedia.org/wiki/Application_programming_interface
  - https://en.wikipedia.org/wiki/Indicator_of_compromise
  - https://en.wikipedia.org/wiki/Virtual_machine
  - https://en.wikipedia.org/wiki/Static_program_analysis
  - https://en.wikipedia.org/wiki/Dynamic_program_analysis
  - https://en.wikipedia.org/wiki/Workstation
  - https://en.wikipedia.org/wiki/Reverse_engineering
- Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software – Book
- The IDA Pro Book