Malware & Reversing

- Interesting Malware
    - Conficker.
    - Morris worm.
    - Zeus malware.
    - Stuxnet.
    - Wannacry.
    - CookieMiner.
    - Sunburst.
- Malware Features
    - Various methods of getting remote code execution.
    - Domain-flux.
    - Fast-Flux.
    - Covert C2 channels.
    - Evasion techniques (e.g. anti-sandbox).
    - Process hollowing.
    - Mutexes.
    - Multi-vector and polymorphic attacks.
    - RAT (remote access trojan) features.
- Decompiling and Reversing
    - Obfuscation of code, unique strings (you can use for identifying code).
    - IdaPro, Ghidra.
- Static and Dynamic Analysis
    - Describe the differences.
    - Virus total.
    - Reverse.it.
    - Hybrid Analysis.

Interesting Malware

These malware instances highlight various approaches to cyber threats, from network worms to sophisticated supply chain attacks. Each had a unique impact, changing how we view and respond to cyber threats.

1. Conficker

- First detected in 2008, this worm exploited Windows OS vulnerabilities, forming a large botnet.
- It spread via network shares and removable media, affecting millions of computers globally.

2. Morris Worm

- Released in 1988 by Robert Tappan Morris, this was one of the first worms distributed via the internet.
- It aimed to measure internet size but caused widespread disruption due to a bug, affecting approximately 10% of the internet.

3. Zeus Malware

- Known for banking credential theft, Zeus (or Zbot) was first identified in 2007.
- It used keylogging and form-grabbing tactics and could spread through phishing emails and drive-by downloads.

4. Stuxnet

- A highly sophisticated worm discovered in 2010, targeting Iran's nuclear facilities.
- It exploited multiple zero-day vulnerabilities, causing physical damage to centrifuges and marking one of the first known cyberattacks targeting critical infrastructure.

5. WannaCry

- This 2017 ransomware attack leveraged the EternalBlue exploit to spread through Windows systems.
- It encrypted user data and demanded payment, causing significant disruptions globally, especially in healthcare and other critical services.

6. CookieMiner

- A cryptocurrency-focused malware targeting macOS users, CookieMiner exploited saved credentials, web cookies, and cryptocurrency wallets.
- Detected in 2019, it was aimed at mining cryptocurrency and exfiltrating sensitive data, notably in the crypto community.

7. Sunburst (SolarWinds)

- Identified in 2020, this malware was embedded in updates of SolarWinds' Orion software.
- It led to a supply chain attack affecting numerous high-profile government and private sector organizations, making it one of the most far-reaching cyber espionage cases.

Malware Features

Here's a breakdown of these malware features and techniques commonly used to enhance their effectiveness and persistence.

1. Remote Code Execution (RCE)

- Malware often employs **RCE methods like buffer overflow, SQL injection, or exploitation of unpatched vulnerabilities, allowing attackers to execute arbitrary code on the target system remotely**.

2. Domain-Flux

- Domain-flux **dynamically generates domain names for command and control (C2) servers**. This makes it harder for defenders to block or take down the malware's infrastructure, as it continually shifts domain names.

3. Fast-Flux

- A technique where **IP addresses associated with malicious domains are frequently changed**, sometimes within seconds, to evade detection and make takedown efforts more difficult.

4. Covert Command and Control (C2) Channels

- Malware often uses **hidden or encrypted channels for C2 communication, including DNS tunneling, HTTP, HTTPS, or even social media channels to avoid detection**.

5. Evasion Techniques

- Malware includes **anti-analysis techniques to evade detection and analysis**, like anti-sandbox methods, virtual machine detection, and timing delays to bypass automated analysis tools.

6. Process Hollowing

- A technique where malware **injects malicious code into a legitimate process, effectively "hollowing" it out**. This enables malware to run within a trusted process, making detection by antivirus software more challenging.

7. Mutexes

- Mutexes are used to **ensure only one instance of the malware runs at a time on a system**. Mutexes can also be used as markers to avoid re-infecting a compromised host.

8. Multi-vector and Polymorphic Attacks

- **Multi-vector attacks leverage various attack types (e.g., phishing, exploit kits) to increase infection chances**. **Polymorphic malware continually changes its code or appearance to avoid signature-based detection**.

9. RAT (Remote Access Trojan) Features

- RATs allow attackers to **gain persistent access and control over infected systems**. Typical features include keylogging, screen capturing, webcam access, file exfiltration, and remote shell access.

Summary

These techniques and features demonstrate the **sophisticated tactics malware can use to evade detection, establish persistence, and maximize the impact of an infection**. They require multifaceted defense mechanisms to counteract.

Decompiling and Reversing

Decompiling and reversing engineering are **techniques used to analyze and understand compiled software**. These methods allow researchers (and attackers) to examine how a program works, identify vulnerabilities, and sometimes bypass protections. In response, many developers employ obfuscation to make reverse engineering more difficult. Here's an overview of decompiling, code obfuscation, unique strings, and popular tools like IDA Pro and Ghidra.

Decompiling and Reversing Engineering

- Definition: Decompiling or reverse engineering is **the process of analyzing compiled code to understand its structure, behavior, and functionality**. This technique is often used to find vulnerabilities, analyze malware, or bypass software protections.
- Process
    - **Disassembly**: Translates machine code into assembly language, which is easier to interpret.
    - **Decompilation**: Converts the binary code back into a higher-level language (like C or Python) to study the program's logic and flow.
- Applications
    - **Malware Analysis**: Security researchers reverse engineer malware to understand its behavior, identify its origin, and create effective defenses.
    - **Vulnerability Discovery**: Identifying flaws in software for exploitation or patching purposes.
    - **Software Cracking**: Attackers reverse engineer applications to remove license checks or other protections.

Code Obfuscation

- Definition: Code obfuscation is a **technique developers use to make code more challenging to read and reverse engineer**. By altering code structure or renaming variables and functions, obfuscation makes it harder to interpret the program's purpose.
- Common Obfuscation Techniques
    - **Variable/Function Renaming**: Renaming variables and functions to meaningless labels (e.g., A1B2C3) makes code more confusing.
    - **Control Flow Alteration**: Modifying the code's flow to include unnecessary operations or convoluted loops, making it harder to follow.
    - **Encryption of Strings**: Encrypting or encoding strings within the code so that they aren't readable without decryption during execution.
- Security Implications: Obfuscation is **commonly used in malware to evade detection and analysis**, as well as in legitimate software to protect intellectual property. However, advanced reverse engineering tools can sometimes deobfuscate code or bypass obfuscation.

Unique Strings for Identification

- Definition: Unique strings in code, such as specific error messages, URLs, or other recognizable data, can be used to identify specific versions or variations of the software.
- Use in Reversing
    - Malware analysts often **look for unique strings in malicious software to trace its origin, behavior, or command-and-control (C2) infrastructure**.

    - Strings can serve as indicators of compromise (IOCs) in threat intelligence, helping identify similar malware samples across different systems.
- Security Implications: Unique strings make it easier for analysts to recognize patterns in code, connect variants of malware, or identify specific functionality within software. Obfuscation often targets these strings to make them harder to use for identification.

Popular Reversing Tools: IDA Pro and Ghidra

- **IDA Pro**
    - Definition: IDA Pro (Interactive Disassembler) is a professional-grade disassembler widely used for reverse engineering.
    - Features
        - Converts machine code into assembly language.
        - Includes powerful visualization tools, such as call graphs and function graphs, to help analyze complex code structures.
        - Supports plug-ins to extend its functionality, including decompilers for certain architectures.
    - Security Implications: IDA Pro is a powerful tool, particularly effective for analyzing complex malware and commercial software. However, it is expensive, making it less accessible to hobbyists.
- Ghidra
    - Definition: Ghidra is an **open-source reverse engineering tool developed by the NSA**, available for free.
    - Features
        - Similar to IDA Pro, it disassembles and decompiles code into higher-level languages.
        - Includes collaboration features that allow multiple analysts to work on the same project.
        - Offers plug-in support and has an active community contributing to its development.
    - Security Implications: Ghidra democratizes access to high-quality reverse engineering tools, making it popular with malware analysts, researchers, and enthusiasts. It provides advanced features and flexibility at no cost, although it lacks some features of IDA Pro.

Summary

- **Decompiling and Reversing** enable the analysis of compiled code, allowing researchers to uncover program logic, vulnerabilities, and malicious functions.
- **Code Obfuscation** complicates reverse engineering by making the code structure harder to understand, protecting intellectual property or malicious functionality.
- **Unique Strings** in code can aid in identifying specific malware samples or versions of software, though obfuscation often targets these strings.
- **IDA Pro and Ghidra are popular tools for reversing**, with IDA Pro offering advanced features for professionals and Ghidra providing a robust, free alternative for the community.

Understanding decompiling, obfuscation, and reversing tools like IDA Pro and Ghidra is essential in fields such as malware analysis, vulnerability research, and intellectual property protection. These tools and techniques empower researchers and defenders, but they also serve as essential skills for attackers seeking to understand and exploit software weaknesses.

Static and Dynamic Analysis

Static and dynamic analysis are **two primary techniques used in malware analysis and reverse engineering to study the behavior of software**, particularly malicious software, without or with execution, respectively. Here's a breakdown of their differences, along with tools commonly used for analysis, such as VirusTotal, Reverse.it, and Hybrid Analysis.

1. Differences Between Static and Dynamic Analysis

- Static Analysis
  - Definition: Static analysis involves **examining the code, structure, and properties of a file without actually running it**. This method is useful for understanding code logic, identifying embedded strings, and finding potential vulnerabilities.
  - Process
    - Analyzing binary code, disassembled code, or decompiled code.
    - Searching for hardcoded strings, URLs, IP addresses, or indicators of compromise (IOCs).
    - Reviewing imported libraries, API calls, and potential file dependencies.
  - Advantages
    - **Safer**, as the malware isn't executed, reducing the risk of infection.
    - Allows for a preliminary understanding of the malware's purpose and structure.
  - Limitations
    - Can be **difficult if the code is heavily obfuscated or packed**.
    - Limited insight into runtime behavior, such as network connections or changes to the file system.
- Dynamic Analysis
  - Definition: Dynamic analysis involves **running the software in a controlled environment (such as a sandbox or virtual machine) to observe its behavior in real time**. Analysts use this method to see what the program does during execution.
  - Process
    - Running the software **in a sandbox to monitor its interactions with the operating system, file system, and network**.
    - Observing the creation of files, registry changes, network requests, and any potential persistence mechanisms.
    - Capturing and analyzing data on the software's runtime behavior.
  - Advantages
    - **Provides concrete data** on the malware's actions, such as network communication, file changes, and process creation.
    - Useful for analyzing malware that only reveals its behavior when executed.
  - Limitations
    - **Requires a secure, isolated environment to prevent spreading malware**.
    - Sophisticated malware may detect the virtual environment or sandbox and alter its behavior, reducing visibility into its full functionality.

2. Tools for Static and Dynamic Analysis

- **VirusTotal**

    - Definition: VirusTotal is a popular **web-based tool that aggregates antivirus engine scans and other analysis tools to detect malware**.
    - Features
        - Allows users to upload files, URLs, or IP addresses for analysis.
        - Provides scan results from dozens of antivirus engines, identifying malware signatures or known malicious indicators.
        - Offers basic static analysis, with additional details such as file hashes, metadata, and identified IOCs.
    - Usage: VirusTotal is frequently used for an initial scan to determine if a file is already known as malware and to gain a quick overview of potential threats.
    - Security Implications: VirusTotal is accessible to anyone, including attackers, who might use it to test if their malware evades detection. Therefore, be cautious about uploading sensitive or proprietary files.
- **Reverse.it (formerly known as Hatching Triage)**
    - Definition: Reverse.it is an **automated malware analysis platform that combines both static and dynamic analysis** to examine files, particularly for security research.
    - Features
        - Provides dynamic sandbox analysis, capturing runtime behavior such as network traffic, file changes, and registry modifications.
        - Supports detailed reporting with data on file structure, dependencies, and observed behaviors.
        - Can perform analysis on various file types, including executables, documents, and scripts.
    - Usage: Reverse.it is particularly useful for in-depth analysis of unknown files and for identifying malware behaviors in a controlled environment.
    - Security Implications: Reverse.it's reports help security teams and researchers understand potential threats before further action or remediation.
- **Hybrid Analysis**
    - Definition: Hybrid Analysis is an **automated malware analysis tool by CrowdStrike, combining both static and dynamic analysis for comprehensive malware examination**.
    - Features
        - Provides dynamic sandboxing to observe runtime behavior, such as network activity, file operations, and process creation.
        - Offers extensive static analysis data, including file structure, metadata, and embedded resources.
        - Presents reports with IOCs and scoring for threat levels, aiding in triaging threats based on severity.
    - Usage: Hybrid Analysis is used to analyze files suspected of malware, gaining insights into their structure and behavior, and it's commonly used for both initial and in-depth examination.
    - Security Implications: Hybrid Analysis offers a balanced approach to malware analysis, with detailed reports that aid in threat detection and response.

Summary

- **Static Analysis examines code without running it**, providing insights into its structure and potential functions. It's **safer** and useful for identifying IOCs but limited against obfuscated or packed code.

- **Dynamic Analysis observes the program in execution**, capturing runtime behaviors like file modifications and network requests. It **provides concrete behavioral data** but requires a controlled environment.
- **VirusTotal**: Useful for quick detection and multi-engine scanning of files, offering initial static analysis results.
- **Reverse.it**: Provides a combination of static and dynamic analysis with in-depth sandboxing capabilities, capturing runtime behaviors.
- **Hybrid Analysis**: Blends static and dynamic analysis, delivering comprehensive reports with threat indicators and actionable data.

These tools, especially when combined, give security teams a robust set of insights into potential malware, helping them identify, understand, and mitigate threats. Using both static and dynamic methods in tandem often yields the most thorough analysis, as they provide complementary views of a program's behavior and structure.