Table of Contents

DLL Injection 2

How DLL Injection Works 2

Common Techniques for DLL Injection 2

API Injection 3

How API Injection Works 3

Common Techniques for API Injection 3

Malware Analysis Tools 4

Categories of Malware Analysis Tools 4

Static Analysis Tools 4

Dynamic Analysis Tools 4

Network Analysis Tools 4

Memory Analysis Tools 5

Conclusion 5

DLL Injection

DLL Injection is a technique used by attackers to manipulate the execution of a running process. By injecting a Dynamic Link Library (DLL) into the memory space of another process, the attacker can execute arbitrary code within the context of that process. This can be used for malicious purposes, such as installing backdoors, stealing information, or manipulating the behavior of software.

How DLL Injection Works

1. **Identify the Target Process:** The attacker first selects a running process based on factors like privilege level or specific functionality that they want to exploit.
2. **Load the DLL:** The attacker forces the target process to load the malicious DLL using one of several methods:
    - **CreateRemoteThread:** This method involves opening the target process, allocating memory within it, writing the DLL path into that memory, and then creating a new thread that runs the LoadLibrary function to load the DLL.
    - **SetWindowsHookEx:** This technique sets a hook in the target process, causing it to load the attacker's DLL when specific events occur (e.g., keyboard input).
    - **NtCreateThreadEx:** Similar to CreateRemoteThread, this method uses a lower-level API to create a thread in the target process to load the DLL, often used to avoid detection.
3. **Execution:** Once the DLL is loaded, the malicious code within it is executed in the context of the target process. This can lead to various malicious activities like spying on the user, altering the behavior of the application, or further spreading the malware.

Common Techniques for DLL Injection

1. **CreateRemoteThread:** The attacker creates a remote thread in the target process to load the DLL, making it one of the most straightforward and commonly used methods.
2. **SetWindowsHookEx:** A hook is set in the target process, causing the DLL to be loaded when a specific system event occurs (like a keystroke), which is useful for keylogging or screen capturing.
3. **Process Hollowing:** The attacker starts a legitimate process and then replaces its memory with the malicious DLL, allowing them to run malicious code under the guise of a legitimate process.

API Injection

API Injection involves intercepting and modifying API calls made by a process. This technique allows attackers to alter the behavior of a program by changing the parameters or return values of API functions. This can be used to bypass security mechanisms, manipulate software behavior, or inject malicious code.

How API Injection Works

- **Intercept API Calls:** The attacker identifies specific API functions they want to intercept. This can be done by altering the Import Address Table (IAT), which contains pointers to the API functions used by a program, or by using techniques like inline hooking to modify the function's code in memory.
- **Modify API Behavior**: Once the API call is intercepted, the attacker can alter its behavior by changing the parameters passed to the function, modifying the return value, or injecting additional malicious code that runs alongside the legitimate function.
- **Execution:** The intercepted and modified API call executes with the attacker's alterations, allowing them to manipulate how the application interacts with the system. This can be used to bypass security measures, inject malicious operations, or alter program behavior.

Common Techniques for API Injection

- **IAT Hooking:** The attacker modifies the Import Address Table so that API calls are redirected to malicious code instead of the original API function. This technique is often used because the IAT provides a central point to intercept all calls to a specific API function.
- **Inline Hooking:** The attacker overwrites the initial instructions of an API function with a jump to their malicious code. After the malicious code executes, control may be returned to the original function or further altered.
- **Detours:** This involves using a library like Microsoft's Detours, which is typically used for legitimate purposes like debugging or extending functionality. Attackers misuse it to intercept and modify API calls by rerouting them through their code before passing control back to the original function.

Malware Analysis Tools

Malware analysis involves examining and understanding the behavior and structure of malicious software. Analysts use various tools to dissect malware, understand its functionality, and develop countermeasures.

Categories of Malware Analysis Tools

Static Analysis Tools

IDAPRO: A powerful disassembler used to analyze binary code without executing it. It provides insights into the structure and behavior of malware.

Ghidra: An open-source software reverse engineering (SRE) tool developed by the NSA. It helps analysts understand the low-level operations of malware.

PE Explorer: A tool that provides detailed information about Portable Executable (PE) files, helping to identify suspicious characteristics in binaries.

Dynamic Analysis Tools

Process Monitor (ProcMon): A tool that monitors and logs system activity in real-time, providing insights into file, registry, and process activities.

Process Explorer: Allows analysts to view detailed information about processes running on a system, including loaded DLLs and memory usage.

Sandboxie: A sandboxing tool that executes malware in a controlled environment, allowing analysts to observe behavior without risking system integrity.

Network Analysis Tools

Wireshark: A network protocol analyzer used to capture and analyze network traffic. It helps in identifying malicious network activity.

Fiddler: A web debugging proxy that logs all HTTP(S) traffic, useful for analyzing malware communication with command and control servers.

Tcpdump: A command-line packet analyzer that provides detailed network traffic information.

Memory Analysis Tools

Volatility: A memory forensics tool that analyzes memory dumps to uncover artifacts left by malware.

Rekall: Another memory forensic framework that helps in identifying malicious activity by analyzing memory dumps.

RAM Capturer: A tool used to capture the contents of the system's memory, which can then be analyzed for malware traces.

Conclusion

DLL Injection, API Injection, and the use of sophisticated malware analysis tools are critical aspects of modern cybersecurity. Understanding these techniques and tools is essential for defending against advanced threats and ensuring the integrity and security of software systems. The use of DLL and API injection by attackers highlights the need for robust security mechanisms and thorough analysis to detect and mitigate such threats effectively.