

## Reto Ingeniería Reversa – SHELLow

Lo primero que hice fue obtener información del archivo SHELLow:

```
dialid@ubuntu:~/Downloads$ file SHELLow
SHELLow: gzip compressed data, last modified: Fri Mar 31 19:11:39 2017, from Uni
x
```

Como es un archivo gzip procedí a ponerle dicha extensión y a descomprimirlo.

Luego hice un cat:

[illegible]

Y nuevamente obtuve información del archivo:

```
ellow
shellow: POSIX tar archive (GNU)
```

Quité la cadena “quita esto para que funcione el programa”.

Y procedí nuevamente a cambiar la extensión del archivo y a descomprimir, obteniendo así un archivo binario shell mod2.

Obtuve las cadenas con strings:

Por lo que pude observar que el archivo ejecutable fue escrito en C.

Al ejecutar el programa, noté que se quedaba esperando una entrada, por lo que procedí a revisar su había puertos abiertos, comando `netstat -ntap`.

```
tcp      0      0 0.0.0.0:39321      0.0.0.0:*          LISTEN
1204/./shell mod2
```

Efectivamente el binario abrió el puerto 39321.

```
dialid@ubuntu: ~/Downloads
/lib64/ld-linux-x86-64.so.2
libc.so.6
puts
__libc_start_main
__gmon_start__
GLIBC_2.2.5
UH-H
ffffff.
Baia, baH
ia ... sH
i que haH
s llegadH
o lejos
It's timH
e to craH
ckme MisH
s/Mr RevH
erse EngH
inner ;)H
[ ]A[A]A^A_
;*3$"
87654-32109-87654-32109-WSSAP
SHELLow was here :P
8}_b
```

Procedí a debuggear el binario con gdb:

```
dialid@ubuntu: ~/Downloads
0x4005b0 <main+170>      mov     rdi, rax
0x4005b3 <main+173>      call    0x4003e0 <puts@plt>
0x4005b8 <main+178>      lea     rax, [rbp-0x70]
0x4005bc <main+182>      mov     rdi, rax
0x4005bf <main+185>      call    0x4003e0 <puts@plt>
B+> 0x4005c4 <main+190>      mov     QWORD PTR [rbp-0x8], 0x600a40
0x4005cc <main+198>      mov     rdx, QWORD PTR [rbp-0x8]
0x4005d0 <main+202>      mov     eax, 0x0
0x4005d5 <main+207>      call    rdx
0x4005d7 <main+209>      leave
0x4005d8 <main+210>      ret
0x4005d9                nop     DWORD PTR [rax+0x0]
0x4005e0 < libc_csu init>    push    r15
native process 5255 In: main                                L??  PC: 0x4005c4
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
(gdb) r
Starting program: /home/dialid/Downloads/shell_mod2

Breakpoint 1, 0x0000000004005c4 in main ()
(gdb)
```

Desensamblando la función main y realizando varios breakpoints y pruebas encontré la siguiente función:

```
dialid@ubuntu: ~/Downloads
> 0x600a40 <shellcode>    mov     edx,0xcfee357c
0x600a45 <shellcode+5>    fcmovu st,st(4)
0x600a47 <shellcode+7>    fnstenv [rsp-0xc]
0x600a4b <shellcode+11>   pop     rsi
0x600a4c <shellcode+12>   xor     ecx,ecx
0x600a4e <shellcode+14>   mov     cl,0x3a
0x600a50 <shellcode+16>   sub     esi,0xffffffffc
0x600a53 <shellcode+19>   xor     DWORD PTR [rsi+0xf],edx
0x600a56 <shellcode+22>   add     edx,DWORD PTR [rsi+0x73]
0x600a59 <shellcode+25>   xlat    BYTE PTR ds:[rbx]
0x600a5a <shellcode+26>   sbb     edi,esi
0x600a5c <shellcode+28>   jge     0x600a3e
0x600a5e <shellcode+30>   add     bh,dh

native process 5255 In: shellcode          L??  PC: 0x600a40
0x00000000004005cc in main ()
(gdb) si
0x00000000004005d0 in main ()
(gdb) si
0x00000000004005d5 in main ()
(gdb) si
0x0000000000600a40 in shellcode ()
(gdb)
```

Encontré también que en dicha función hay un loop.

El código de este binario fue ofuscado con msfvenom por lo que es un poco más difícil encontrar el camino correcto de la ejecución.

```
> 0x600a50 <shellcode+16> sub     esi,0xffffffffc
0x600a53 <shellcode+19> xor     DWORD PTR [rsi+0xf],edx
0x600a56 <shellcode+22> add     edx,DWORD PTR [rsi+0xf]
0x600a59 <shellcode+25> loop    0x600a50 <shellcode+16>
0x600a5b <shellcode+27> xor     esi,esi
0x600a5d <shellcode+29> mul     esi

native process 5255 In: shellcode          L??  PC: 0x600a50
```

Finalmente entramos a shellcode + 27 colocando un breakpoint que apunta directamente a esta instrucción.

```
dialid@ubuntu: ~/Downloads
Register group: general
rax      0x0      0
rbx      0x0      0
rcx      0x0      0
rdx      0x594c9666 1498191462
rsi      0x600b2d 6294317
rdi      0x1      1

0x600a56 <shellcode+22> add    edx,DWORD PTR [rsi+0xf]
0x600a59 <shellcode+25> loop   0x600a50 <shellcode+16>
B+> 0x600a5b <shellcode+27> xor    esi,esi
0x600a5d <shellcode+29> mul    esi
0x600a5f <shellcode+31> inc    esi
0x600a61 <shellcode+33> push   0x2

native process 5255 In: shellcode L?? PC: 0x600a5b
0x0000000000600a56 in shellcode ()
(gdb) si
0x0000000000600a59 in shellcode ()
(gdb) c
Continuing.

Breakpoint 4, 0x0000000000600a5b in shellcode ()
(gdb) 
```

Y comenzamos a observar las diferentes instrucciones que lo componen.

```
dialid@ubuntu: ~/Downloads

0x600a59 <shellcode+25> loop   0x600a50 <shellcode+16>
0x600a5b <shellcode+27> xor    esi,esi
0x600a5d <shellcode+29> mul    esi
0x600a5f <shellcode+31> inc    esi
0x600a61 <shellcode+33> push   0x2
0x600a63 <shellcode+35> pop    rdi
0x600a64 <shellcode+36> add    al,0x29
0x600a66 <shellcode+38> syscall
0x600a68 <shellcode+40> push   rax
0x600a69 <shellcode+41> pop    rdi
0x600a6a <shellcode+42> push   0x2
0x600a6c <shellcode+44> mov    WORD PTR [rsp+0x2],0x9999
0x600a73 <shellcode+51> push   rsp
```

La primera llamada al sistema que vemos en shellcode es <shellcode+38>, anteriormente vemos que se hace una operación add al,0x29. Para llamadas al sistema el valor 0x29 corresponde a socket.

```
dialid@ubuntu: ~/Downloads
0x600a73 <shellcode+51> push    rsp
0x600a74 <shellcode+52> pop     rsi
0x600a75 <shellcode+53> push    rdx
0x600a76 <shellcode+54> push    0x10
0x600a78 <shellcode+56> pop     rdx
0x600a79 <shellcode+57> push    0x31
0x600a7b <shellcode+59> pop     rax
0x600a7c <shellcode+60> syscall
0x600a7e <shellcode+62> pop     rsi
0x600a7f <shellcode+63> mov     al,0x32
0x600a81 <shellcode+65> syscall
0x600a83 <shellcode+67> mov     al,0x2b
0x600a85 <shellcode+69> syscall
```

Después tenemos las llamadas al sistema 0x31 que corresponde a bind y recibe como parámetro struct sockaddr \*, así como 0x32 que corresponde a listen y pone en escucha el puerto abierto anteriormente y 0x2b acepta la conexión del mismo socket.

En esa última llamada al sistema el programa se queda esperando hasta que recibe una cadena por el puerto a la escucha.

Posteriormente empieza a hacer las comparaciones referentes a la cadena que recibe a través del cliente socket en el puerto 39321.

La primera comparación sucede cuando verifica el salto de línea para entonces comparar dicha cadena.

```
dialid@ubuntu: ~/Downloads
Register group: general
rax      0x0      0
rbx      0x0      0
rcx      0x600a98 6294168
rdx      0x20     32
rsi      0x7fffffffdf0 140737488346608
rdi      0x4      4

0x600a94 <shellcode+84> xor     eax,eax
0x600a96 <shellcode+86> syscall
0x600a98 <shellcode+88> xor     rax,rax
> 0x600a9b <shellcode+91> mov     al,0x4a
0x600a9d <shellcode+93> sub     al,0x40
0x600a9f <shellcode+95> xor     rcx,rcx

native process 5255 In: shellcode
0x00000000000000a94 in shellcode ()
(gdb) si
0x00000000000000a96 in shellcode ()
(gdb) si
0x00000000000000a98 in shellcode ()
```

```
C program '\x' escapes are noted.
Oct  Dec  Hex  Char
000  0    00   NUL '\0' (null character)
001  1    01   SOH (start of heading)
002  2    02   STX (start of text)
003  3    03   ETX (end of text)
004  4    04   EOT (end of transmission)
005  5    05   ENQ (enquiry)
006  6    06   ACK (acknowledge)
007  7    07   BEL '\a' (bell)
010  8    08   BS  '\b' (backspace)
011  9    09   HT  '\t' (horizontal tab)
012 10   0A   LF  '\n' (new line)
013 11   0B   VT  '\v' (vertical tab)
014 12   0C   FF  '\f' (form feed)
015 13   0D   CR  '\r' (carriage ret)
016 14   0E   SO  (shift out)
017 15   0F   SI  (shift in)
020 16   10   DLE (data link escape)
021 17   11   DC1 (device control 1)
022 18   12   DC2 (device control 2)
```

[illegible]

rsp+rcx\*1 es la cadena que le hemos pasado y al almacena el salto de línea en este caso.

```
dialid@ubuntu: ~/Downloads
```

```
Register group: general
```

rax	0xa	10
rbx	0x0	0
rcx	0x4	4
rdx	0x20	32
rsi	0x7fffffffddfd0	140737488346608
rdi	0x4	4

```
> 0x600aa2 <shellcode+98>      cmp     BYTE PTR [rsp+rcx*1],al  
0x600aa5 <shellcode+101>    je       0x600aac <shellcode+108>  
0x600aa7 <shellcode+103>    inc       rcx  
> 0x600aaa <shellcode+106>   jmp       0x600aa2 <shellcode+98>  
0x600aac <shellcode+108>    cmp       rcx,0x1d  
0x600ab0 <shellcode+112>    jne       0x600b3f <shellcode+255>
```

```
native process 5255 In: shellcode                                L?? PC: 0x600aaa  
0x000000000000600aa2 in shellcode ()  
(gdb) si  
0x000000000000600aa5 in shellcode ()  
(gdb) si  
0x000000000000600aa7 in shellcode ()  
(gdb) si  
0x000000000000600aaa in shellcode ()  
(gdb)
```

Luego empieza a incrementar el contador rcx, para verificar el número de caracteres, así como las comparaciones sucesivas.

Compara si el tamaño de la cadena es igual a 29:



```

dialid@ubuntu: ~/Downloads
Register group: general
ax      0xa      10
bx      0x0      0
cx      0x1d     29
dx      0x20     32
si      0x7fffffffdd80  140737488346496
di      0x4      4

0x600aa7 <shellcode+103>    inc    rcx
0x600aaa <shellcode+106>    jmp    0x600aa2 <shellcode+98>
0x600aac <shellcode+108>    cmp    rcx,0x1d
0x600ab0 <shellcode+112>    jne    0x600b3f <shellcode+255>
0x600ab6 <shellcode+118>    xor    rcx,rcx
0x600ab9 <shellcode+121>    add    cl,0x5

Active process 5509 In: shellcode      L??  PC: 0x600aac
000000000000600aa5 in shellcode ()
fb) si
000000000000600aa7 in shellcode ()
fb) si
000000000000600aaa in shellcode ()
fb) si
000000000000600aa2 in shellcode ()
fb) si
000000000000600aa5 in shellcode ()
fb) si
000000000000600aac in shellcode ()
fb) █

```

Compara si en la posición 5 hay un – que es 0x2d:

Realiza lo mismo para las siguientes posiciones con guiones.

```

dialid@ubuntu: ~/Downloads
Register group: general
ax      0xa      10
bx      0x0      0
cx      0x5      5
dx      0x20     32
si      0x7fffffffdd80  140737488346496
di      0x4      4

0x600ab6 <shellcode+118>    xor    rcx,rcx
0x600ab9 <shellcode+121>    add    cl,0x5
0x600abc <shellcode+124>    cmp    BYTE PTR [rsp+rcx*1],0x2d
0x600ac0 <shellcode+128>    jne    0x600b3f <shellcode+255>
0x600ac2 <shellcode+130>    add    cl,0x6
0x600ac5 <shellcode+133>    cmp    cl,0x11

```

Compara que la suma de todos los caracteres de la cadena de como resultado 2272 en decimal o 0x8e0.

```
dialid@ubuntu: ~/Downloads
Register group: general
rax      0x8e0    2272
rbx      0x4c     76
rcx      0x0      0
rdx      0x20     32
rsi      0x7fffffffdd80   140737488346496
rdi      0x4      4

0x600ae3 <shellcode+163> add rax,rbx
0x600ae6 <shellcode+166> cmp rax,0x8e0
0x600aec <shellcode+172> jne 0x600b3f <shellcode+255>
0x600aee <shellcode+174> lea rdx,[rsp+0xc]
0x600af3 <shellcode+179> xor rcx,rcx
0x600af6 <shellcode+182> mov cl,0x5

Active process 5509 In: shellcode L?? PC: 0x600aee
db) si
0000000000000000add in shellcode ()
db) si
0000000000000000ae0 in shellcode ()
db) si
0000000000000000ae3 in shellcode ()
db) si
0000000000000000ae6 in shellcode ()
db) si
0000000000000000aec in shellcode ()
db) si
0000000000000000aee in shellcode ()
db)
```

Realiza una función test y va decrementando el contador ecx:

La otra condición que debe cumplir el serial es que los valores de los caracteres 15-18 sean consecutivos.



```

dialid@ubuntu: ~/Downloads
Register group: general
rax      0x0      0
rbx      0x4c     76
rcx      0x2      2
rdx      0x7fffffffdd8c  140737488346508
rsi      0x7fffffffdd80  140737488346496
rdi      0x4      4

0x600af6 <shellcode+182>      mov     cl,0x5
0x600af8 <shellcode+184>      xor     rax,rax
> 0x600afb <shellcode+187>     test    BYTE PTR [rdx+rcx*1],0x41
0x600aff <shellcode+191>      jne     0x600b0a <shellcode+202>
0x600b01 <shellcode+193>      test    BYTE PTR [rdx+rcx*1],0x61
0x600b05 <shellcode+197>      jne     0x600b0a <shellcode+202>

native process 5509 In: shellcode      L??  PC: 0x600afb
(gdb) si
0x0000000000600afb in shellcode ()
(gdb) si
0x0000000000600aff in shellcode ()
(gdb) si
0x0000000000600b01 in shellcode ()
(gdb) si
0x0000000000600b05 in shellcode ()
(gdb) si
0x0000000000600b0a in shellcode ()
(gdb) si
0x0000000000600afb in shellcode ()
(gdb) █

```

Cuando alguna de las condiciones es que no se cumple el programa nos lleva al tope del programa <shellcode+255> lo cual nos retorna un segmentation fault.

```

dialid@ubuntu: ~/Downloads
Register group: general
rax      0xa      10
rbx      0x0      0
rcx      0x79     121
rdx      0x20     32
rsi      0x7fffffffddf0  140737488346608
rdi      0xffffffffffffda  -38

0x600b3b <shellcode+251>      mov     al,0x3b
0x600b3d <shellcode+253>      syscall
> 0x600b3f <shellcode+255>     or      al,BYTE PTR [rax]
0x600b41 <completed.6661>     add     BYTE PTR [rax],al
0x600b43                          add     BYTE PTR [rax],al
0x600b45                          add     BYTE PTR [rax],al

native process 5400 In: shellcode      L??  PC: 0x600b3f
(gdb) b *0x600a5b
Breakpoint 1 at 0x600a5b
(gdb) r
Starting program: /home/dialid/Downloads/shell_mod2
Baia, baia ... si que has llegado lejos
Program received signal SIGSEGV, Segmentation fault.
0x0000000000600b3f in shellcode ()
(gdb) █

```

```
dialid@ubuntu: ~/Downloads

  2 3 4 5 6 7      30 40 50 60 70 80 90 100 110 120
  -----
0:  0 @ P ` p      0:  ( 2 < F P Z d n x
1:  ! 1 A Q a q      1:  ) 3 = G Q [ e o y
2:  " 2 B R b r      2:  * 4 > H R \ f p z
3:  # 3 C S c s      3:  ! + 5 ? I S ] g q {
4:  $ 4 D T d t      4:  " , 6 @ J T ^ h r |
5:  % 5 E U e u      5:  # - 7 A K U _ i s }
6:  & 6 F V f v      6:  $ . 8 B L V ` j t ~
7:  ' 7 G W g w      7:  % / 9 C M W a k u DEL
8:  ( 8 H X h x      8:  & 0 : D N X b l v
9:  ) 9 I Y i y      9:  ' 1 ; E O Y c m w
A:  * : J Z j z
B:  + ; K [ k {
C:  , < L l |
D:  - = M ] m }
E:  . > N ^ n ~
F:  / ? 0 _ o DEL

NOTES
History
An ascii manual page appeared in Version 7 of AT&T UNIX.
Manual page ascii(7) line 86/135 79% (press h for help or q to quit)
```

Con las condiciones anteriores y con ayuda de la tabla ascii encontré el siguiente serial que cumple las condiciones:

LEZLY-DIALI-DC?>=-ERONRO\}}}}

x4c/x45/x5a/x4c/x59 x2D x44/x49/x41/x4c/x49 x2D x44/x43/x40/x3f/x3e/x3d x2D  
x45/x52/x4f/x4e/x52/x4f/x5c/x7d/x7d/x7d/x7d

Ejecutamos y obtenemos una shell del lado del cliente:

```
dialid@ubuntu:~/Documents/vulne$ nc localhost 39321
LEZLY-DIALI-DC?>=-ERONRO\}}}}
ls
SHELLow
allinone
allinone.c
shell_mod2
shellow
shellow2
shellow3.tar
str_allinone
strings_aio
sublime_text_3_build_3211_x64(1).tar.bz2
sublime_text_3_build_3211_x64.tar.bz2
whoami
dialid
```

## Pseudocódigo del programa

- |  |                         |
|--|-------------------------|
| 1. Imprimir cadenas de crack me.   | Printf()                |
| 2. Crear socket.   | Socket()                |
| 3. Hacer bind socket.  | Bind socket()           |
| 4. Poner el puerto a la escucha.   | Listen()                |
| 5. Aceptar las conexiones.   | Accept()                |
| 6. Revisar si existe un salto de línea y almacenar la cadena   | if(\n) cadena=          |
| 7. Comparar si cadena tiene longitud de 29.  | If(len(cadena))==29     |
| 8. Comparar si cadena tiene guiones cada 5 caracteres.   | If(cadena[i]%5)='0x2d') |
| 9. Comparar si la suma de los valores numéricos de los caracteres es 2272.                           | If(suma==2272)          |
| 10. Comparar si los caracteres en las posiciones 15-17 son consecutivos.                             |                         |
| 11. Si todas las condiciones anteriores se cumplen, crea una shell /bin/dash en el lado del cliente. | Shellcode()             |
| 12. Si no se cumplen, devuelve un segmentation fault.  | Exit()                  |