

Hernández González Ricardo Omar

Examen

Lo primero que haremos será hacerle un **file** al archivo descargado para saber de qué se trata:

```
[23:47]~/Documents/DGTIC/AnalisisVulnerabilidades:master X file SHELLow
SHELLow: gzip compressed data, last modified: Fri Mar 31 19:11:39 2017, from Unix, original size 10240
```

Podemos observar que es un comprimido, por lo cual lo descomprimimos con **tar**:

```
[23:49]~/Documents/DGTIC/AnalisisVulnerabilidades:master X ➤ tar -xvzf SHELLOW
shell_mod2
```

Observamos que nos descomprime un archivo, lo ejecutamos para ver que pasa:

```
[23:49]~/Documents/DGfIC/AnalisisVulnerabilidades:master x ➤ ./shell_mod2
zsh: formato de ejecutable incorrecto: ./shell_mod2
```

No se puede ejecutar, por lo que procedemos a ver que contiene internamente. Al hacerle un **strings** vemos que tiene una parte de más:

```
[23:50]~/Documents/DGTIC/AnalisisVulnerabilidades:master X ➤ strings shell_mod2
ELQuitaestoparaquefuncioneelprograma
/lib64/ld-linux-x86-64.so.2
libc.so.6
puts
__libc_start_main
```

Por lo que podemos deducir que al ejecutable se le agregaron cosas, se las quitamos, esto lo podemos hacer de varias formas, en este caso lo haremos con un editor:

```
[23:52]~/Documents/DGTIC/AnalisisVulnerabilidades:master X vim shell_mod2
```

```
1 ^?ELFquitaestoparaquefuncioneelprograma^B^A^A^@^@
  @^[^@F^@^@E^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
  ^@^@^@^@^@^@B^@^@^@^@^@\^@^@^@^@^@\^@^@^@^@^@
  ^@^@^@<8c>^G^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
```

```
1 ^?EL^B^A^A^@^@^@^@^@^@^@^@^@^@B^@>^@^A^@^@^@P  
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@  
^@^@^@^@^@\^@^@^@^@^@^@^@^@A^@^@^@^@^@^@^@^@A^@^@  
^@^@^@A^@^@^@F^@^@^@<90>^G^@^@^@^@^@^@^@<90>^G`  
^@"^G`^@^@^@^@^@"^G`^@^@^@^@^@B^A^@^@^@^@^@E  
^@^@^@D^@^@^@^@^@^@^@^@D^@^@^@^@^@^@^@^@P^@t^d^P^@
```

Lo guardamos y lo volvemos a ejecutar:

```
(23:54)~/Documents/DGTIC/AnalisisVulnerabilidades:master ✖ ➦ ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
█
```

Ahora podemos observar que ya se ejecuta pero que está a la espera de algo, por lo que probamos en escribir texto y vemos que pasa:

```
{23:54}~/Documents/DGTIC/AnalisisVulnerabilidades:master X ➤ ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)

a
af
gprueba
fjakf
ajfjk
})
```

Entonces podemos pensar que la onda va por otro lado. Procedemos a ver la información del archivo con **rabin2**:

```
{23:57}~/Documents/DGTIC/AnalisisVulnerabilidades:master X ➤ rabin2 -I shell_mod2
arch      x86
binsz     5427
bintype   elf
bits      64
canary    false
class     ELF64
crypto    false
endian    little
havecode  true
intrp     /lib64/ld-linux-x86-64.so.2
lang      c
linenum   true
lsyms     true
machine   AMD x86-64 architecture
maxopsz   16
minopsz   1
nx        false
os        linux
pcalign   0
pic       false
relocs    true
relro     no
rpath     NONE
static    false
stripped  false
subsys    linux
va        true
```

Donde solo tendremos que estamos tratando con mnemónicos de 64 bits. Después veremos sus cadenas:

```
{0:03}~/Documents/DGTIC/AnalisisVulnerabilidades:master X ➤ rabin2 -zqq shell_mod2
87654-32109-87654-321DRO-WSSAP
SHELLow was here :P
^1n7
8}_b
6[J4
{B;H~
```

Podemos ver que se trata de algún tipo de serial, por lo cual nos da más información de qué tenemos que hacer, ahora solo falta el saber cómo está compuesto el serial y como se lo pasamos. Posteriormente hacemos el analisis con **strace**:

```
[0:05]~/Documents/DGTTIC/Analysis/vulnerabilidades:master X ⌂ strace ./shell_mod2
execve("./shell_mod2", ["/.shell_mod2"], 0x7ffe461cbde0 /* 49 vars */) = 0
brk(NULL) = 0x19c2000
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=191346, ...}) = 0
mmap(NULL, 191346, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f30e7238000
close(3) = 0
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\1\0\0\0\240\33\2\0\0\0\0\0\0... 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1800248, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f30e7236000
mmap(NULL, 3906272, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f30e6c89000
mprotect(0x7f30e6e3a000, 2093056, PROT_NONE) = 0
mmap(0x7f30e7039000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7f30e7039000
mmap(0x7f30e703f000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f30e703f000
close(3) = 0
arch_prctl(ARCH_SET_FS, 0x7f30e72374c0) = 0
mprotect(0x7f30e7039000, 16384, PROT_READ) = 0
mprotect(0x7f30e7267000, 4096, PROT_READ) = 0
munmap(0x7f30e7238000, 191346) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 2), ...}) = 0
brk(NULL) = 0x19c2000
brk(0x19e3000) = 0x19e3000
write(1, "Baia, baia ... si que has llegado"..., 40Baia, baia ... si que has llegado lejos
) = 40
write(1, "It's time to crackme Miss/Mr Rev"..., 49It's time to crackme Miss/Mr Reverse Engineer ;)
) = 49
socket(AF_INET, SOCK_STREAM, IPPROTO_IP) = 3
bind(3, {sa_family=AF_INET, sin_port=htons(39321)}, sin_addr=inet_addr("0.0.0.0")), 16) = -1 EADDRINUSE (Address already in use)
strace: [ Process PID=16375 runs in x86 mode. ]
syscall_0xffffffffffff32(0x3, 0, 0x10, 0x76655220724d2f73, 0x617263206f742065, 0x73694d20656d6b63) = -1 (errno 38)
syscall_0xffffffffffff7b(0x3, 0, 0x10, 0x76655220724d2f73, 0x617263206f742065, 0x73694d20656d6b63) = -1 (errno 38)
```

```
[23:54]~/Documents/DGTIC/AnalisisVulnerabilidades:master x ➤ nc localhost 39321
87654-32109-87654-321DRO-WSSAP
```

```
[0:08]~/Documents/DGTIC/AnalisisVulnerabilidades:master x ➤ ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
[1] 18733 segmentation fault ./shell_mod2
```

```
{0:13}~/Documents/DGTIC/AnálisisVulnerabilidades:master ✖ ▢ readelf -a shell_mod2
Encabezado ELF:
Mágico: 7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00
Clase: ELF64
Datos: complemento a 2, little endian
Versión: 1 (current)
OS/ABI: UNIX - System V
Versión ABI: 0
Tipo: EXEC (Fichero ejecutable)
Máquina: Advanced Micro Devices X86-64
Versión: 0x1
Dirección del punto de entrada: 0x400410
Inicio de encabezados de programa: 64 (bytes en el fichero)
Inicio de encabezados de sección: 5432 (bytes en el fichero)
Opciones: 0x0
Tamaño de este encabezado: 64 (bytes)
Tamaño de encabezados de programa: 56 (bytes)
Número de encabezados de programa: 8
Tamaño de encabezados de sección: 64 (bytes)
Número de encabezados de sección: 30
Índice de tabla de cadenas de sección de encabezado: 27
```


Más abajo, en la tabla de símbolos, observamos que hay una parte del código que se llama **shellcode**, esto también fue encontrado cuando hicimos **strings**, pero aquí encontramos más información acerca de ellos, vemos que es un objeto y por el nombre suponemos hace algo que queremos obtener con el serial.

Symbol table '.symtab' contains 68 entries:

Num:	Valor	Tam	Tipo	Unión	Vis	Nombre	Ind
0:	0000000000000000	0	NOTYPE	LOCAL	DEFAULT	UND	
1:	00000000000400200	0	SECTION	LOCAL	DEFAULT		1
2:	0000000000040021c	0	SECTION	LOCAL	DEFAULT		2
3:	0000000000040023c	0	SECTION	LOCAL	DEFAULT		3
4:	00000000000400260	0	SECTION	LOCAL	DEFAULT		4
5:	00000000000400280	0	SECTION	LOCAL	DEFAULT		5
6:	000000000004002e0	0	SECTION	LOCAL	DEFAULT		6
7:	0000000000040031e	0	SECTION	LOCAL	DEFAULT		7
8:	00000000000400328	0	SECTION	LOCAL	DEFAULT		8
9:	00000000000400348	0	SECTION	LOCAL	DEFAULT		9
55:	000000000006009c8	0	OBJECT	GLOBAL	HIDDEN	24 __dso_handle	
56:	00000000000400660	4	OBJECT	GLOBAL	DEFAULT	15 __IO_stdin_used	
57:	00000000000600a00	31	OBJECT	GLOBAL	DEFAULT	24 msg2	
58:	000000000004005e0	101	FUNC	GLOBAL	DEFAULT	13 __libc_csu_init	
59:	00000000000600b48	0	NOTYPE	GLOBAL	DEFAULT	25 _end	
60:	00000000000400410	0	FUNC	GLOBAL	DEFAULT	13 _start	
61:	00000000000600a40	257	OBJECT	GLOBAL	DEFAULT	24 shellcode	
62:	00000000000600b41	0	NOTYPE	GLOBAL	DEFAULT	25 __bss_start	
63:	00000000000400506	211	FUNC	GLOBAL	DEFAULT	13 main	
64:	00000000000000000	0	NOTYPE	WEAK	DEFAULT	UND __Jv_RegisterClasses	
65:	00000000000600b48	0	OBJECT	GLOBAL	HIDDEN	24 __TMC_END__	
66:	00000000000000000	0	NOTYPE	WEAK	DEFAULT	UND __ITM_registerTMCloneTable	
67:	000000000004003a8	0	FUNC	GLOBAL	DEFAULT	11 _init	

Por último, después de toda la información obtenida anteriormente, procederemos a debugear el programa, para esto usaremos gdb, ponemos un break en el main y lo ejecutamos:

```
{0:20}~/Documents/DGTIC/AnálisisVulnerabilidades:master ✗ ➔ gdb shell_mod2 -q
Reading symbols from shell_mod2...(no debugging symbols found)...done.
(gdb) b main
Breakpoint 1 at 0x40050a
(gdb) r
Starting program: /home/richard_ohg/Documents/DGTIC/AnálisisVulnerabilidades/shell_mod2

Breakpoint 1, 0x0000000000040050a in main ()
(gdb) █
```

Ponemos el **layout asm** y **layout regs** para ver los registros y las instrucciones, ahí observaremos que llega al punto donde hace un call a un registro y se pasa a la zona del shellcode:

```
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
```

```
> 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,0xfffffffffc
0x600a53 <shellcode+19> xor     DWORD PTR [rsi+0xf],edx
```

Ahí es donde veremos las comparaciones que necesitamos para el serial. Podemos observar, mediante el análisis del código ensamblador, que el total de caracteres debe ser 29, que debe llevar entre cada 5 caracteres un guion medio '-' y por último, que la suma de todos los caracteres debe ser 2272.

Conclusión:

Con esto, podemos decir que el ejecutable abre un socket con puerto 39321. Que tiene que recibir la cadena desde el socket. Que la cadena es un serial con características específicas. El serial debe contener 29 caracteres, debe tener un guion entre cada 5 caracteres y la suma de dichos caracteres debe ser 2272.

Una entrada válida es esta:

```
{1:03}~/Documents/DGTIC/AnalisisVulnerabilidades:master X ➔ ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)

{1:05}~/Documents/DGTIC/AnalisisVulnerabilidades:master X ➔ nc localhost 39321
gRICA-RDOHE-RNAND-EZx|a-cNbHr
id
uid=1000(richard_ohg) gid=1000(richard_ohg) groups=1000(richard_ohg),7(lp),27(sudo),100(users),1
8(ubridge)
pwd
/home/richard_ohg/Documents/DGTIC/AnalisisVulnerabilidades
```

y con esto obtendremos una shell.