

## REPORTE DE EXAMEN

Inmediatamente descargado el archivo con “file” revisé el tipo de archivo y al ser un gzip lo descomprimí. De nuevo volví a revisar el archivo extraído esta vez llamado shell\_mod2 y esta vez como resultado era el binario, pero el archivo ELF no se reconocía. Al intentar ejecutarlo tampoco funcionó.

```
lilium@debian:~/Documents/ExamenVulnes$ file SHELLow
SHELLow: gzip compressed data, last modified: Fri Mar 31 19:11:39 2017, from Unix
lilium@debian:~/Documents/ExamenVulnes$ tar -xzf SHELLow
shell_mod2
lilium@debian:~/Documents/ExamenVulnes$ ls
shell_mod2  SHELLow
lilium@debian:~/Documents/ExamenVulnes$ file shell_mod2
shell_mod2: ELF, unknown class 113
```

Al analizarlo rápidamente con Strings se mostraban varias cadenas incluyendo una extra, más tarde supe que estaba contenida en el header de ELF y con Vim lo eliminé.

```
lilium@debian:~/Documents/ExamenVulnes$ strings shell_mod2
ELFquitaestoparaquefuncioneelprograma
/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
```

Una vez quitada esa cadena, al volver a hacer el file del binario se podía ver más información y de hecho ya se podía ejecutar.

```
lilium@debian:~/Documents/ExamenVulnes$ file shell_mod2
shell_mod2: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=4ab82577a85ffa8894e95109fb63bdd2f199903f, not stripped
```

Al ejecutarlo parecía mantenerse a la espera de algo y al poner una entrada no ocurría nada.

```
lilium@debian:~/Documents/ExamenVulnes$ ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
```

Al listar los puertos que se encontraban escuchando se mostraba el puerto 39321, abierto por el binario.

```
lilium@debian:~/Documents$ lsof -i -P | grep LISTEN
shell_mod 25520 lilium    3u  IPv4 530229      0t0  TCP *:39321 (LISTEN)
```

Al listar el contenido de la sección de .data aparecía una cadena escrita de forma extraña

“87654-32109-87654-321DRO-WSSAP”

```
lilium@debian:~/Documents/ExamenVulnes$ readelf --hex-dump=24 shell_mod2
Hex dump of section '.data':
0x006009c0 00000000 00000000 00000000 00000000 .....
0x006009d0 00000000 00000000 00000000 00000000 .....
0x006009e0 00000000 00000000 00000000 00000000 .....
0x006009f0 00000000 00000000 00000000 00000000 .....
0x00600a00 38373635 342d3332 3130392d 38373635 87654-32109-8765
0x00600a10 342d3332 3144524f 2d575353 41500000 4-321DRO-WSSAP..
0x00600a20 5348454c 4c6f7720 77617320 68657265 SHELLow was here
0x00600a30 203a5000 00000000 00000000 00000000 :P.....
0x00600a40 ba7c35ee cfdadc9 7424f45e 31c9b13a .|5.....t$.^1..
```

Esa cadena, junto con la cadena revertida fueron enviadas con netcat pero causó un error.

```
lilium@debian:~/Documents$ nc localhost 39321
87654-32109-87654-321DRO-WSSAP
lilium@debian:~/Documents$ nc localhost 39321
PASSW-ORD123-45678-90123-45678
lilium@debian:~/Documents$ 
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
Segmentation fault
lilium@debian:~/Documents/ExamenVulnes$ ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
Segmentation fault
```

Al seguir analizando el binario, con radare2 se mostraban las siguientes funciones

```
-[ functions ]-----
(a) add      (x)xrefs      (q)quit
(r) rename   (c)calls      (g)go
(d) delete   (v)variables  (?)help
* 0x00400410  41 entry0
  0x00400506 211 sym.main
  0x004003e0  32 sym.imp.puts
  0x00400440  41 sym.deregister_tm_clones
  0x00400480  53 sym.register_tm_clones
  0x004004c0  28 sym.__do_global_dtors_aux
  0x004004e0  35 sym.frame_dummy
  0x00400650   2 sym.__libc_csu_fini
  0x00400654   9 sym._fini
  0x004005e0 101 sym.__libc_csu_init
  0x004003a8  26 sym._init
  0x00400400  48 loc.imp.__gmon_start__
  0x004003f0  48 sym.imp.__libc_start_main
```

De igual manera se mostraba la estructura del main de la siguiente manera

En este se puede ver que manda a llamar 3 funciones en particular. 2 puts y con call rdx, lo cual implica que manda a llamar lo que se encuentra dentro de rdx en esa parte de la ejecución.

Los puts solo son para mostrar las líneas que aparecen en cuanto se ejecuta el programa, por lo cual, la llamada importante es la que hace a rdx.

```
[0x400506] ;[b]
;-- main:
(fcn) sym.main 211
sym.main ();
; var int local_70h @ rbp-0x70
; var int local_68h @ rbp-0x68
; var int local_60h @ rbp-0x60
; var int local_58h @ rbp-0x58
; var int local_50h @ rbp-0x50
; var int local_48h @ rbp-0x48
; var int local_40h @ rbp-0x40
; var int local_30h @ rbp-0x30
; var int local_28h @ rbp-0x28
; var int local_20h @ rbp-0x20
; var int local_18h @ rbp-0x18
; var int local_10h @ rbp-0x10
; var int local_8h @ rbp-0x8
; DATA XREF from 0x0040042d (entry0)
push rbp
mov rbp, rsp
sub rsp, 0x70 ; 'p'
movabs rax, 0x6162202c61696142
mov qword [rbp - local_30h], rax
movabs rax, 0x73202e2e2e206169
mov qword [rbp - local_28h], rax
movabs rax, 0x6168206575712069
mov qword [rbp - local_20h], rax
movabs rax, 0x646167656c6c2073
mov qword [rbp - local_18h], rax
movabs rax, 0x736f6a656c206f
mov qword [rbp - local_10h], rax
movabs rax, 0x6d69742073277449
mov qword [rbp - local_70h], rax
movabs rax, 0x617263206f742065
mov qword [rbp - local_68h], rax
movabs rax, 0x73694d20656d6b63
mov qword [rbp - local_60h], rax
movabs rax, 0x76655220724d2f73
mov qword [rbp - local_58h], rax
movabs rax, 0x676e452065737265
mov qword [rbp - local_50h], rax
movabs rax, 0x293b2072656e6e69
mov qword [rbp - local_48h], rax
mov byte [rbp - local_40h], 0
lea rax, qword [rbp - local_30h]
mov rdi, rax
call sym.imp.puts ;[a]; int puts(const char *s);
lea rax, qword [rbp - local_70h]
mov rdi, rax
call sym.imp.puts ;[a]; int puts(const char *s);
mov qword [rbp - local_8h], obj.shellcode
mov rdx, qword [rbp - local_8h]
mov eax, 0
call rdx
leave
ret
```

Después con gdb puse un breake en main y en la llamada a rdx para poder ver qué era lo que realizaba.

```

0x4005bc <main+182>    mov     rdi, rax
0x4005bf <main+185>    call   0x4003e0 <puts@plt>
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
0x4005d5 <main+207>    call   rdx

Native process 26021 In: main
(gdb) run
Starting program: /home/lilium/Documents/ExamenVulnes/shell_mod2
(gdb) b *0x4005d5
Breakpoint 2 at 0x4005d5
(gdb) c
Continuing.
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)

Breakpoint 2, 0x00000000004005d5 in main ()
(gdb)

```

Una vez adentro, aparecían varias llamadas al sistema, pero en particular hubo una que dejaba a gdb sin poder ejecutarse hasta que se realizara una conexión a través de netcat.

```

rcx      0x0      0
rax      0x2b     43
rcx      0x600a83 6294147
rsi      0x0      0
r10      0x848    2120
r8       0x7ffff7fd8440 140737353974848
r10      0x7ffff7dd3b58 140737351859032

0x4005bc <main+182>    mov     rdi, rax
0x4005bf <main+185>    call   0x4003e0 <puts@plt>
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
0x600a87 <shellcode+71> push    rax
0x600a88 <shellcode+72> pop     rdi
native 0x600a89 <shellcode+73> xor     rdx, rdx
(gdb) run
Starting program: /home/lilium/Documents/ExamenVulnes/shell_
0x0000000000600a79 in shellcode ()
0x0000000000600a7b in shellcode ()
0x0000000000600a7c in shellcode ()
0x0000000000600a7e in shellcode ()
0x0000000000600a7f in shellcode ()
0x0000000000600a81 in shellcode ()
0x0000000000600a83 in shellcode ()
0x0000000000600a85 in shellcode ()

```

Una vez realizada la conexión ya permitía seguir ejecutando.

```
lilium@debian:~/Documents$ nc localhost 39321

0x600a85 <shellcode+69> syscall
> 0x600a87 <shellcode+71> push    rax
0x600a88 <shellcode+72> pop     rdi
native 0x600a89 <shellcode+73> xor     rdx,rdx
(gdb) run
Starting program: /home/lilium/Documents/ExamenVulnes/shell_mod2
0x0000000000600a7b in shellcode ()
0x0000000000600a7c in shellcode ()
0x0000000000600a7e in shellcode ()
0x0000000000600a7f in shellcode ()
0x0000000000600a81 in shellcode ()
0x0000000000600a83 in shellcode ()
0x0000000000600a85 in shellcode ()
0x0000000000600a87 in shellcode ()
(gdb) 
```

De nuevo en un punto esperaba la entrada de algo y en ese momento envié la cadena que me aparecía para ver cómo la estaba analizando.

```
lilium@debian:~/Documents$ nc localhost 39321
PASSW-ORD123-45678-90123-45678
```

Una vez dentro vi que los registros cambiaron y había uno que apuntaba directamente a donde se encontraba la cadena que recién había enviado.

```
rsi      0x7fffffff0f0  140737488347376  rdi      0x4
rbp      0x7fffffff170  0x7fffffff170   rsp      0x7fffffff
r8       0x7ffff7fd8440 140737353974848  r9       0x0
r10      0x7ffff7dd3b58 140737351859032  r11      0x346
r12      0x400410 4195344  r13      0x7fffffff
r14      0x0 0  r15      0x0

0x600a91 <shellcode+81> push    rsp
0x600a92 <shellcode+82> push    rsp
0x600a93 <shellcode+83> pop     rsi
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

native process 26021 In: shellcode
0x7fffffff0f0: PASSW-ORD123-45678-90123-45678 nackme Miss/Mr Reverse Enginner ;)
0x7fffffff131: ""
0x7fffffff132: ""
0x7fffffff133: ""
```

Siguiendo en la ejecución me aparecía un loop que se encontraba contando la cantidad de caracteres que habían sido recibidos, una vez terminado, hacía una comparación de la cantidad de caracteres enviados (En este caso 30) con 29.

```

Register group: general
rax      0xa      10      rbx
rcx      0x1e     30      rdx
rsi      0x7fffffff0f0  140737488347376  rdi
rbp      0x7fffffff170  0x7fffffff170  rsp
r8       0x7ffff7fd8440  140737353974848  r9
r10      0x7ffff7dd3b58  140737351859032  r11
r12      0x400410  4195344      r13
r14      0x0      0      r15

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>
0x600ab6 <shellcode+118>     xor     rcx,rcx
0x600ab9 <shellcode+121>     add     cl,0x5

```

Al no cumplir con esta condición enviaba a shellcode+255 y terminaba la ejecución con "segmentation default", lo cual indicaba que solo permitía 29 caracteres de entrada.

```

0x600b3a <shellcode+250>     pop     rdi
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
0x600b47                     add     BYTE PTR [rax],al

native process 26021 In: shellcode
0x000000000000600aa2 in shellcode ()
0x000000000000600aa5 in shellcode ()
0x000000000000600aac in shellcode ()
(gdb) si
0x000000000000600ab0 in shellcode ()
0x000000000000600b3f in shellcode ()
(gdb) si
Program received signal SIGSEGV, Segmentation fault.
0x000000000000600b3f in shellcode ()
(gdb)

```

De nuevo envié una cadena esta vez con 29 caracteres y me permitió continuar con la ejecución, esta vez realizando una comparación del tipo de carácter, donde checaba que hubieran guiones en posiciones específicas de la cadena (cada 5 caracteres)

```

rcx      0x5      5      rdx      0x20     32
rsi      0x7fffffff0f0  140737488347376  rdi      0x4      4
r8       0x7ffff7fd8440  140737353974848  rsp      0x7fffffff0f0
r10      0x7ffff7dd3b58  140737351859032  r11      0x302    770
r14      0x0      0      r11      0x346    838

0x4005d5 <main+207>      call    rdx
0x600a7c <shellcode+60> syscall
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
> 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
native proc8ss 26172 In:136> jbe     0x600abc <shellcode+124>
(gdb) run
0x0000000000600a78 in shellcode ()
0x7ffffffffffe136: ""
0x7ffffffffffe137: ""
0x7ffffffffffe138: "-\006@"
---Type <return> to continue, or q <return> to quit---
Quit
(gdb) x/s 0x7ffffffffffe0f5
0x7ffffffffffe0f5: "-AAAAA-AAAA-AAAA-AAAA\nrackme Miss/Mr Reverse Enginner ;)"
(gdb) x/x 0x7ffffffffffe0f5
0x7ffffffffffe0f5: 0x2d

```

Una vez pasada esa parte, de nuevo me aparecía una comparación, pero esta vez sumaba el valor decimal del ascii de cada carácter que se enviaba y se podía notar que debía ser un total de 8e0, o en decimal 2272.

```

rcx      0x0      0      rdx
rax      0x70d     1805   rbx
rcx      0x0      0      rdx
rsi      0x7fffffff0f0  140737488347376  rdi
r10      0x848     2120   rsp
r8       0x7ffff7fd8440  140737353974848  r9
r10      0x7ffff7dd3b58  140737351859032  r11

0x400506 <main>      push    rbp
0x600ad8 <shellcode+152> add     rax,rbx
0x600adb <shellcode+155> loop    0x600ad2 <shellcode+146>
B+ 0x600add <shellcode+157> xor     rbx,rbx
0x600ae0 <shellcode+160> mov     bl,BYTE PTR [rsp+rcx*1]
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]

```

En la siguiente ejecución proporcioné una cadena que cumplía las condiciones anteriores y además sumaba 2272 en ascii, que era la siguiente

SSSSS-SSSSS-SSSSS-SSSSd



A partir de esa, entró a la última condición que verificaba que se encontrara una "A" en al menos las ultimas 2 secciones que eran separadas por los guiones de la cadena.

```

rcx      0x0      0      rdx
rax      0x0      0      rbx
rcx      0x3      3      rdx
rsi      0x7fffffff0f0 140737488347376 rdi
r10      0x848    2120   rsp
r8       0x7ffff7fd8440 140737353974848 r9
r10      0x7ffff7dd3b58 140737351859032 r11
0x4005c4 <main+190>    mov     QWORD PTR [rbp-0x8],0x600a40
0x4005cc <main+198>    mov     rdx,QWORD PTR [rbp-0x8]
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+189>
0x600b01 <shellcode+193> test    BYTE PTR [rdx+rcx*1],0x61
0x600b05 <shellcode+197> jne     0x600b0a <shellcode+189>
native600b07 <shellcode+199> inc     rax
Starting program: /home/lilium/Documents/ExamenVulnes/shell_mod2
shellcode
0x0000000000600afb in shellcode ()
(gdb) x/s 0x7ffff7ffe100
0x7ffff7ffe100: "S SSSSS-SSSSd'nackme Miss/Mr Reverse Enginner ;)"
(gdb) si
0x0000000000600aff in shellcode ()
0x0000000000600b0a in shellcode ()
0x0000000000600afb in shellcode ()
(gdb)

```

En particular, aquí tuve un problema pues a veces permitía seguir a la cadena y a veces no, le pasara A o a, o no le pasara nada.

Tomando en cuenta todas las condiciones anteriores armé un serial que ocupara mi nombre:

diana-tadeo-AAAAA-AAAA-AUUUX

Al probarlo me devolvió una shell.

```

lilium@debian: ~/Documents
File Edit View Search Terminal Help
lilium@debian:~/Documents/ExamenVulnes$ ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)

lilium@debian: ~/Documents
File Edit View Search Terminal Help
lilium@debian:~/Documents$ nc localhost 39321
diana-tadeo-AAAAA-AAAA-AUUUX
uname
Linux
whoami
lilium
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin

```