López Fernández Servando Miguel

Evaluación Final Análisis de Vulnerabilidades

El primer paso que realice fue otorgar permisos de lectura, escritura y ejecución a SHELLow.

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# chmod 744 SHELLow
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# ls -la
total 12
drwxr-xr-x  2 root root 4096 Apr 15 21:36 .
drwxr-xr-x 10 root root 4096 Apr 15 21:34 ..
-rwxr--r--  1 root root 2987 Apr 15 21:35 SHELLow
```

A continuación ejecute el dgb tratando de conocer un poco sobre las funciones del binario, pero obtuve el siguiente error.

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# gdb SHELLow -q
"/root/Documents/AnalisisDeVulnes/Evaluacion/SHELLow": not in executable format: file format not recognized
(gdb)
```

Ejecute un file para conocer el tipo de archivo y agregue la extensión correspondiente para tener más control.

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# file SHELLow
SHELLow: gzip compressed data, last modified: Fri Mar 31 19:11:39 2017, from Unix, original size 10240
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# mv SHELLow SHELLow.gz
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion#
```

Descomprimi el archivo y revise el tipo de archivo.

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# gzip -d SHELLow.gz
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# ls
SHELLow
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# file
Usage: file [-bcCdEhikLlNnprsvzZ0] [--apple] [--extension] [--mime-encoding]
            [--mime-type] [-e <testname>] [-F <separator>]  [-f <namefile>]
            [-m <magicfiles>] [-P <parameter=value>] <file> ...
       file -C [-m <magicfiles>]
       file [--help]
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# file SHELLow
SHELLow: POSIX tar archive (GNU)
```

Volví a revisar el tipo de archivo y realice otra descompresión.

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# file SHELLow
SHELLow: POSIX tar archive (GNU)
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# mv SHELLow SHELLow.tar
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# tar -xvf SHELLow.tar
shell_mod2
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# ls
shell_mod2  SHELLow.tar
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# file shell_mod2
shell_mod2: ELF, unknown class 113
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion#
```

Imagine que las trampas habían terminado pero estaba en un error.

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# gdb ./shell_mod2 -q
"/root/Documents/AnalisisDeVulnes/Evaluacion/./shell_mod2": not in executable format: file format not recognized
(gdb)
```

Recordé que linux determina el tipo de archivo por la cabecera de este y no por la extensión, por lo tanto ejecute hexdump para intentar analizar si existía algún problema con el archivo y observe lo siguiente.

```
0000000 177   E   L   F   q   u   i   t   a   e   s   t   o   p   a   r
0000010   a   q   u   e   f   u   n   c   i   o   n   e   e   l   p   r
0000020   o   g   r   a   m   a 002 001 001  \0  \0  \0  \0  \0  \0  \0
0000030  \0  \0 002  \0   >  \0 001  \0  \0  \0 020 004   @  \0  \0  \0
0000040  \0  \0   @  \0  \0  \0  \0  \0  \0  \0   8 025  \0  \0  \0  \0
0000050  \0  \0  \0  \0  \0  \0   @  \0   8  \0  \b  \0   @  \0 036  \0
0000060 033  \0 006  \0  \0  \0 005  \0  \0  \0   @  \0  \0  \0  \0  \0
```

Por lo tanto elimine los los bytes que contenían la frase “quita esto para que funcione el programa”

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# head -c 4 shell_mod2 > shell_mod
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# hexdump -c shell_mod  | less
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# hexdump -c shell_mod  | less
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# dd if=shell_mod2 of=shell_mod11 ibs=38 skip=1
193+1 records in
14+1 records out
7348 bytes (7.3 kB, 7.2 KiB) copied, 0.000853627 s, 8.6 MB/s
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# cat shell_mod shell_mod11 > shell
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# hexdump -c shell | less
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion#
```

Y comencé a descubrir el funcionamiento del binario.

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# gdb ./shell -q
Reading symbols from ./shell...(no debugging symbols found)...done.
(gdb) i functions
All defined functions:

Non-debugging symbols:
0x00000000004003a8  _init
0x00000000004003e0  puts@plt
0x00000000004003f0  __libc_start_main@plt
0x0000000000400400  __gmon_start__@plt
0x0000000000400410  _start
0x0000000000400440  deregister_tm_clones
0x0000000000400480  register_tm_clones
0x00000000004004c0  __do_global_dtors_aux
0x00000000004004e0  frame_dummy
0x0000000000400506  main
0x00000000004005e0  __libc_csu_init
0x0000000000400650  __libc_csu_fini
0x0000000000400654  _fini
(gdb) b main
Breakpoint 1 at 0x40050a
(gdb) run
Starting program: /root/Documents/AnalisisDeVulnes/Evaluacion/shell

Breakpoint 1, 0x000000000040050a in main ()
(gdb) layout regs
```

Al correr una vez el programa me percate que en un punto se detenía y era justo despues de realizar una syscall por lo tanto, corrí nuevamente el programa y revise todas las syscalls.

- syscall 41 o 0x29 - sys_socket
- syscall 49 o 0x31 - sys_bind
- syscall 51 o 0x32 - sys_listen
- syscall 43 o 0x2b - sys_accept

Envíe la salida de los puertos a un archivo antes y al ejecutar el script, observé las diferencias para encontrar el puerto.

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# lsof -i -P -n > antes.txt
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# lsof -i -P -n > despues.txt
```

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# ss -tla > antes.txt
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# ss -tla > despues.txt
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# cmp antes.txt despues.txt
antes.txt despues.txt differ: byte 10, line 1
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# cat antes.txt
State      Recv-Q  Send-Q    Local Address:Port              Peer Address:Port
LISTEN     0       128           127.0.0.1:postgresql             0.0.0.0:*
LISTEN     0       128           127.0.0.1:5433                   0.0.0.0:*
TIME-WAIT  0       0           192.168.0.8:58612          168.197.123.161:http
LISTEN     0       128               [::1]:postgresql               [::]:*
LISTEN     0       128               [::1]:5433                     [::]:*
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# cat despues.txt
State    Recv-Q   Send-Q     Local Address:Port             Peer Address:Port
LISTEN   0        128            127.0.0.1:postgresql           0.0.0.0:*
LISTEN   0        0                0.0.0.0:39321                0.0.0.0:*
LISTEN   0        128            127.0.0.1:5433                 0.0.0.0:*
LISTEN   0        128                [::1]:postgresql             [::]:*
LISTEN   0        128                [::1]:5433                   [::]:*
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion#
```

Envíe la cadena que parece un serial para descubrir si funciona.

```
ckme MisH
s/Mr RevH
erse EngH
inner ;)H
[]A\A]A^A_
;*3$"
87654-32109-87654-321DRO-WSSAP
SHELLow was here :P
8}_b
 6[J4
{B;H~
```

Una vez enviada la cadena con netcat observamos que la carga en la pila y realiza una comparación.

```
   0x600a9f <shellcode+95>         xor    rcx,rcx
 > 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
native process 3723 In: shellcode
0x0000000000600a93 in shellcode ()
0x0000000000600a94 in shellcode ()
0x0000000000600a96 in shellcode ()
(gdb) si
0x0000000000600a98 in shellcode ()
0x0000000000600a9b in shellcode ()
0x0000000000600a9d in shellcode ()
0x0000000000600a9f in shellcode ()
0x0000000000600aa2 in shellcode ()
(gdb) x/5x $rsp
0x7fffffffe0e0: 0x35363738      0x32332d34      0x2d393031      0x35363738
0x7fffffffe0f0: 0x32332d34
(gdb) x/3s $rsp
0x7fffffffe0e0: "87654-32109-87654-321DRO-WSSAP\nackme Miss/Mr Reverse Enginner ;)"
0x7fffffffe121: ""
0x7fffffffe122: ""
```

El primer filtro es un contador de caracteres mientras el carácter sea diferente de \n

```
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>
```

el contador anterior permite comprobar la longitud del serial, para continuar debe tener 0x1d(29) caracteres.

```
 > 0x600aac <shellcode+108>        cmp    rcx,0x1d
   0x600ab0 <shellcode+112>        jne    0x600b3f <shellcode+255>
```

Después comprueba que a cada 5 caracteres exista un guión y un carácter menor a 0x11(puede ser un salto de línea) para continuar .

```
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
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
0x600ac8 <shellcode+136>        jbe    0x600abc <shellcode+124>
```

La suma de los caracteres de la clave debe dar 0x830 para continuar.

```
0x600aca <shellcode+138>        xor    rcx,rcx
0x600acd <shellcode+141>        mov    cl,0x1c
0x600acf <shellcode+143>        xor    rax,rax
0x600ad2 <shellcode+146>        xor    rbx,rbx
0x600ad5 <shellcode+149>        mov    bl,BYTE PTR [rsp+rcx*1]
0x600ad8 <shellcode+152>        add    rax,rbx
0x600adb <shellcode+155>        loop   0x600ad2 <shellcode+146>
```

```
0x600adb <shellcode+155>        loop   0x600ad2 <shellcode+146>
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
```

Estas condiciones permiten obtener la shell, de esta manera se comprueba que la cadena

“FLAG -{SeRv-AnDo -MiGuE-l5SM}”

es un serial correcto.

```
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 235, in <
    test()
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 231, in t
    BaseHTTPServer.test(HandlerClass, ServerClass)
  File "/usr/lib/python2.7/BaseHTTPServer.py", line 610, in tes
    httpd.serve_forever()
  File "/usr/lib/python2.7/SocketServer.py", line 231, in serve
    poll_interval)
  File "/usr/lib/python2.7/SocketServer.py", line 150, in _eint
    return func(*args)
KeyboardInterrupt
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# ./shell
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)
```

```
root@kali:~/Documents/AnalisisDeVulnes/Evaluacion# nc localhost 39321
FLAG -{SeRv-AnDo -MiGuE-l5SM}

ls
SHELLow.tar
antes.txt
despues.txt
shell
shell_mod
shell_mod11
shell_mod2
```