

SHELLow

Análisis de Vulnerabilidades



Pacheco Franco Jesús Enrique

jesus.pacheco@bec.seguridad.unam.mx

15/Abril/2019

Objetivo

Encontrar la una cadena de entrada al programa que contenga mi nombre de tal manera que se me permita el acceso y se ejecute una Shell.

Desarrollo

Lo primero que hice fue descargarme el ejecutable del repositorio que se me indico y posteriormente mi primera decisión fue ejecutarlo para ver como se comportaba y me encuentro con lo siguiente.

```
root@sins:/home/exam# ./SHELLow
bash: ./SHELLow: cannot execute binary file: Exec format error
```

Entonces pensé ala mejor viene comprimido o algo por el estilo y lo siguiente que se me ocurrió fue hacer un file SHELLow.

```
root@sins:/home/exam# file SHELLow
SHELLow: gzip compressed data, last modified: Fri Mar 31 19:11:39 2017, from Unix, original size 10240
```

Y entonces observé que en efecto era un comprimido y lo descomprimí con el comando zcat SHELLow > descomprimido.

```
root@sins:/home/exam# zcat SHELLow > descomprimido
root@sins:/home/exam# file descomprimido
descomprimido: POSIX tar archive (GNU)
```

Pensé que seria una buena idea volver a hacer un file para ver que tipo de archivo teníamos y ahora nos encontramos con un tar entonces lo descomprimimos otra vez.

```
root@sins:/home/exam# tar -xvf descomprimido
shell_mod2
root@sins:/home/exam# file shell_mod2
shell_mod2: ELF, unknown class 113
```

Después de hacerle un file al archivo obtenido observamos que es un ELF entonces ya podemos ejecutarlo.

```
root@sins:/home/exam# ./shell_mod2
bash: ./shell_mod2: cannot execute binary file: Exec format error
```

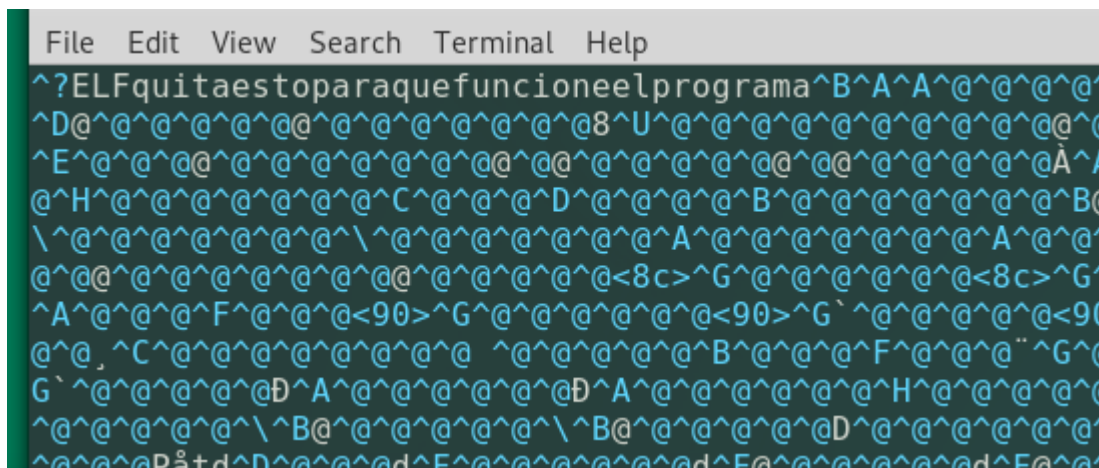
Al parecer algo no esta cuadrando, a pesar de ser un ELF no se ejecuta, entonces se me ocurrió hacerle un strings a ver qué pasaba.

```

root@sins:/home/exam# 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
fffff.

```

Entonces observo que hay un texto raro que dice que lo quite para que funcione y le hago caso. Abro vim y procedo a remover el texto.



Probamos a ejecutar de nuevo y entonces...

```

File Edit View Search Terminal Help
root@sins:/home/exam# ./shell_mod2
Baia, baia ... si que has llegado lejos
It's time to crackme Miss/Mr Reverse Enginner ;)

```

Bingo ya podemos empezar :/

Lo siguiente que hice fue probar a escribir alguna entrada y dar enter para ver si me marcaba algún error o algo por el estilo, pero por más caracteres que escribía y por mas enters que daba no hacia nada, entonces mejor me pasé a analizarlo con gdb.

```

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

```

Entonces en gdb me encontré con esto. Se estaba realizando una llamada al sistema y después de navegar por internet un poco encontré que el 0x29 correspondía con la llamada socket entonces supuse que se creaba un socket al ejecutar el programa. Para comprobarlo puse con watch el comando ss -tla para ver que socket se crean cada 0.1 segundos y bingo.

```

File Edit View Search Terminal Help
Every 0.1s: ss -tla                                sins: Mon Apr 15 20:54:12 2019
State      Recv-Q    Send-Q      Local Address:Port    Peer Address:Port
LISTEN     0          128         127.0.0.1:1928        0.0.0.0:*
LISTEN     0          0           0.0.0.0:39321        0.0.0.0:*

```

A partir de aquí en adelante fue ir leyendo el código e ir averiguando como tenia que ser la cadena de entrada para lograr poder obtener la Shell, en general si se encontraba que la cadena no cumplía con el patrón se terminaba el programa.

Se encontraron tres condiciones que se tenían que satisfacer para que el programa pudiera funcionar.

1. La cadena de entrada debe ser de longitud 29 decimal 0x1D hexadecimal.
2. La cadena debe tener 3 guiones medios en las posiciones 5, 11 y 17 contando a partir de 0 de izquierda a derecha.
3. Si se suma el numero decimal ascii correspondiente de cada carácter de la cadena el total de la suma debe de ser 2272 decimal, 8e0 hexadecimal.

Conexión con NetCat

```

File Edit View Search Terminal Help
root@sins:/home/exam# nc localhost 39321

```

```

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>

```

En la pila debe de haber un valor oa que también esta en el registro al (circulo azul) y aparte rcx se va incrementando cada que se lee un carácter de los que ingresamos por lo que podemos inferir que la longitud de la cadena debe de ser 0x1d o 29 decimal (circulo verde).

```

xor     rcx,rcx
add     cl,0x5
cmp     BYTE PTR [rsp+rcx*1],0x2d
jne     0x600b3f <shellcode+255>
add     cl,0x6
cmp     cl,0x11
jbe     0x600abc <shellcode+124>

```

Sección del código donde se valida la existencia de los guiones medios '-' el valor hexadecimal 0x2d corresponde con dicho carácter.

```

0x600ad8 <shellcode+152>   add     rax,rbx
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
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
0x600af8 <shellcode+184>   xor     rax,rax

```

Validación de que la suma de los caracteres coincida con 0x8e0 (2272 en decimal).

Esas tres condiciones son suficientes y necesarias para que se nos permita el acceso, una vez con esto en mente podemos proceder a formar nuestra cadena personalizada y probar.

Después de sumar los valores ASCII de varias combinaciones de caracteres como demente llegue a la siguiente cadena que me permitió el acceso al Shell.

jESUS-eNRIQ-UEpAC-HECOfRANCoj

Y ahora solo resta probar dicha cadena para comprobar que funciona.

The image shows three terminal windows from a Kali Linux machine (root@sins: ~).

The top window is running `ss -tla` to monitor network connections. It shows a listener on `127.0.0.1:1928` and `0.0.0.0:39321`. An established connection is visible on `127.0.0.1:39321` from `127.0.0.1:58938`.

The bottom-left window shows the execution of `./shell_mod2` in the `/home/exam` directory. The output is a message in Spanish: "Baia, baia ... si que has llegado lejos It's time to crackme Miss/Mr Reverse Enginner ;)", followed by a prompt.

The bottom-right window shows a netcat client connection to `localhost 39321`. The client sends the payload `jESUS-eNRIQ-UEpAC-HECOfRANCoj`. The server responds with the user `id` (`uid=0(root) gid=0(root) groups=0(root)`), the current directory `/home/exam`, and a list of files: `ls` shows `SHELLow`, `bueno`, `descomprimido`, `h`, `shell_mod2`, `shellcode`, `shellcode.asm`, `sock.py`, and `sumaHex.py`.