

Instrucciones para la práctica final del curso:

El objetivo es sencillo: ¡Captura la bandera!

Servidor: *tretornesp.com*

Puerto: *5000*

Permite hasta 5 conexiones simultáneas, por favor, no dejéis terminales conectadas sin utilizar pues bloquearéis a vuestros compañeros.

Preparación de la práctica:

Descarga el código desde:

<https://github.com/TretornESP/bec/blob/main/final/reto/server/pwnme/files/main.c>

<https://github.com/TretornESP/bec/blob/main/final/reto/exploit/exploit.py>

Contenidos:

- exploit.py: Esqueleto del exploit.
- main.c: código vulnerable que se ejecuta en el servidor.

Requisitos:

- Máquina Linux, preferiblemente con Ubuntu20.04.
- Gcc y gdb (recomendado ghidra).
- Python3 con pwntools (pip install pwntools)

Compilación:

```
gcc -no-pie -fno-stack-protector -z execstack -g main.c -o  
a.out
```

Pasos para completar la práctica:

- 1) Ejecutar nmap sobre la máquina con el siguiente comando:

```
nmap tretornesp.com -Pn -p 5000
```

y comprobar que el puerto esté abierto:

```
norte@XABIER-IGLESIAS:/mnt/c/Users/xabier.iglesias/Desktop$ nmap -Pn tretornesp.com -p 5000
Starting Nmap 7.80 ( https://nmap.org ) at 2022-04-13 14:48 CEST
Nmap scan report for tretornesp.com (82.223.81.88)
Host is up (0.024s latency).

PORT      STATE SERVICE
5000/tcp  open  upnp

Nmap done: 1 IP address (1 host up) scanned in 1.09 seconds
```

- 2) Conectarse al servidor mediante netcat y jugar un poco con él:

```
nc tretornesp.com 5000
```

```
norte@XABIER-IGLESIAS:/mnt/c/Users/xabier.iglesias/Desktop$ nc tretornesp.com 5000
Welcome to the BEC high security server
Enter a command:asd
received 4 bytes; echoing
asd
Enter a command:dsa
received 4 bytes; echoing
dsa
Enter a command:|
```

- 3) Revisar el código fuente del programa (o intentar la ingeniería inversa!).

4) Identificar la función vulnerable y probar que el programa crashea.

```
welcome to the nc-high security server
Enter a command:
received 0 bytes; echoing
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[Data]Buffer empieza en: 0x7fffffff0819
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
====Data====
root@HARSHEN-IGLESIAS:/mnt/c/Users/xabier.iglesias/Desktop#
```

5) Seleccionar el payload que deseamos, tenemos principalmente dos opciones:

- Ejecutar una Shell.
- Leer la flag.

Se recomienda usar msfvenom para generarlo.

Nota: No uséis payloads de conexión remota, la máquina solo tiene expuesto el puerto del server.

6) Explotar el programa local dentro de gdb.

7) Modificar el esqueleto del exploit proporcionado.

En concreto necesitaréis 4 cosas:

- Introducir un shellcode válido.
- Calcular el relleno hasta el desbordar el RIP salvado.
- Calcular la dirección de inicio de vuestro shellcode.
- Introducir el comando vulnerable.

Consejos:

- Vigila que el shellcode no sea más grande que el buffer!
- Cuidado con los badchars y nullbytes.
- Aslr está desactivado, además tenéis una ayudita para la dirección de inicio, revisad el output de la función vulnerable.
- Si no sois capaces de acertar la dirección del stack, podéis usar una nopsled.

```
1  from pwn import *
2  import sys
3  import time
4
5  if len(sys.argv) < 3:
6      print("Usage: exploit.py ip port")
7      sys.exit(0)
8
9  buf = b'' #Completa el payload!
10 buf += b"
11 buf += b"
12 buf += b"
13 buf += b"
14 buf += b"
15 buf += b'
16 buf += b'
17 buf += b'
18 buf += p64(0x7f) #Dirección del shellcode
19
20 print("Connecting to {} on port {}".format(sys.argv[1], sys.argv[2]))
21 conn = remote(sys.argv[1],int(sys.argv[2]))
22 conn.recvline()
23 conn.recvuntil(b':', drop=True)
24 conn.sendline(b'') # En que comando está la vulnerabilidad? modifica esta linea
25 time.sleep(1) #Esto es importante! si no los dos send se mezclan
26 conn.sendline(buf)
27 time.sleep(1)
28 conn.interactive() #Si tu exploit hace execve(/bin/bash), si es un shell remoto no hace falta
```

8) Ejecutar el exploit y comprobar que tenemos una Shell:

python3 ./exploit.py tretornesp.com 5000

```
norte@XABIER-IGLESIAS:/mnt/c/Users/xabier.iglesias/Desktop/bec/final/reto/exploit$ python3 ./exploit.py tretornesp.com 5000
Connecting to tretornesp.com on port 5000
[+] Opening connection to tretornesp.com on port 5000: Done
Connecting to tretornesp.com on port 5000
[+] Opening connection to tretornesp.com on port 5000: Done
[*] Switching to interactive mode
Connecting to tretornesp.com on port 5000
[+] Opening connection to tretornesp.com on port 5000: Done
[*] Switching to interactive mode
received 4 bytes; echoing

$ whoami
root
$
```

9) Obtener la flag:

La flag está almacenada en el directorio: */app*

`cat /app/flag.txt`

```
$ cat /app/flag.txt
BEC{          }$
```

10) Enviar la flag:

Finalmente, para obtener el certificado, solo tenéis que mandarme la flag y la hora en la que la habéis conseguido a la dirección:

xabier.iglesias.perez@udc.es

Dudas y problemas: Para cualquier duda o problema a la hora de completar la práctica, escribid a la dirección de arriba. Habrá una sesión dedicada a resolver dudas el día 19.

Plazo: Mes de Abril.