CS315 · Computer Security

Lab 12

胡玉斌 / 11712121

December 14, 2020

Task 1:

Finding out the addresses of libc functions

We can find that the address of `system()` is `0xb7da4da0` and that for `exit()` is `0xb7d989d0`

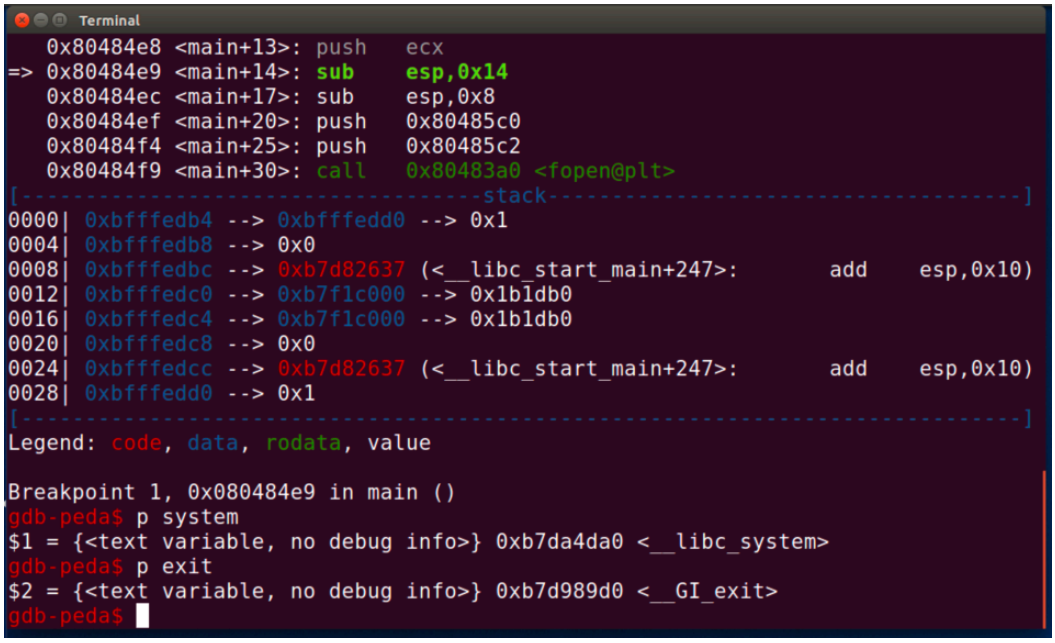


Figure 1: address of lib function

Task 2:

Putting the shell string in the memory

Setting the config of bash environment, we can find the shell variable is `0xbfffffec`

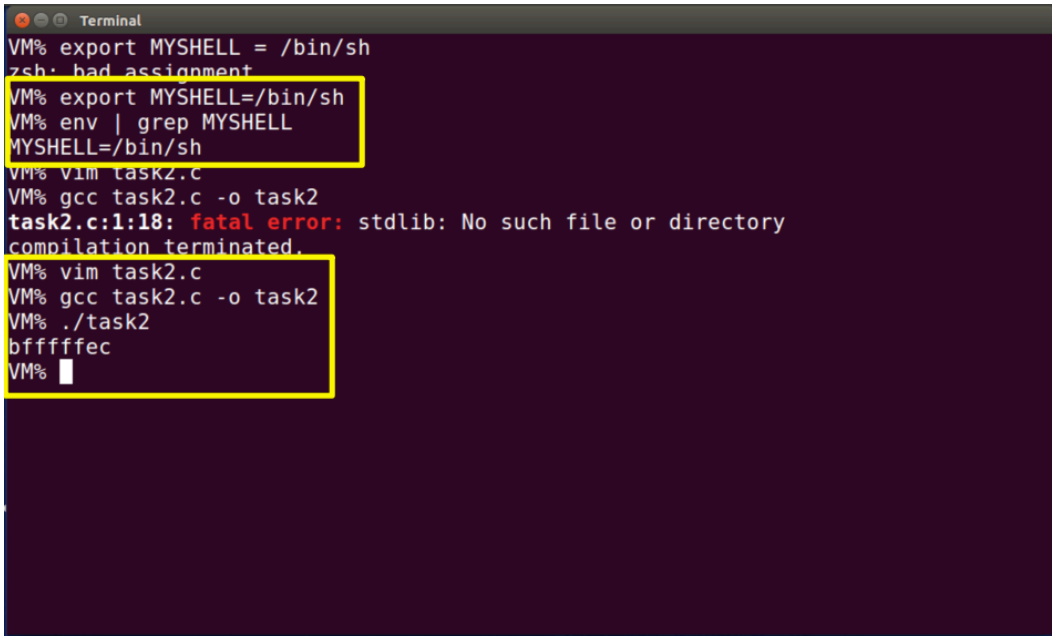


Figure 2: address of shell

Task 3:

Exploiting the Buffer-Overflow Vulnerability

First, we generate the `badfile`

```
VM% echo $(printf "\xff\x01\x02\x03\x04\x05\x06\x07\x08\x09\x10\x11\x12\x13\x14\
x15\x16\x17\x18\x19\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x30\x31\x32\x33\x34\
x35\x36\x37\x38\x39") > badfile
```

Figure 3: badfile

When we put the address which we find from task1 and task2 to the `exploit.c`, we got error that it points to `bin/sh`, so we modify `0xbfffffec` to `0xbfffffeb`

```
VM% gcc exploit.c -o exploit
VM% ./exploit
VM% ./retlib
zsh:1: no such file or directory: bin/sh
```

Figure 4: error address

success attack

Figure 5: code

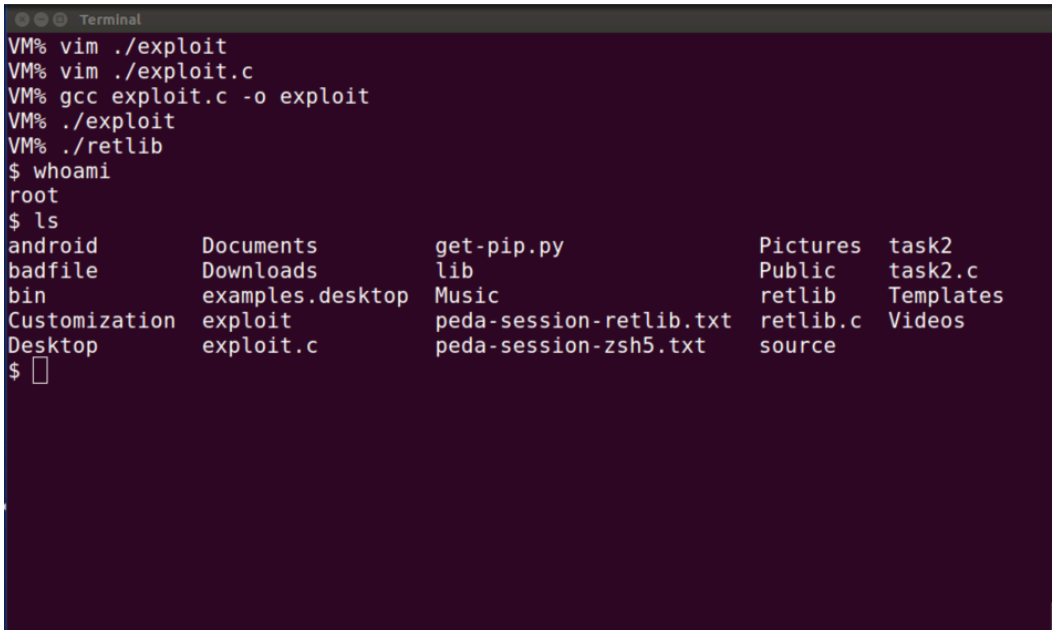


Figure 6: root shell

Attack variation 1:

Return address of `system()` is not valid, so we will get `segmentation fault`

Figure 7: code

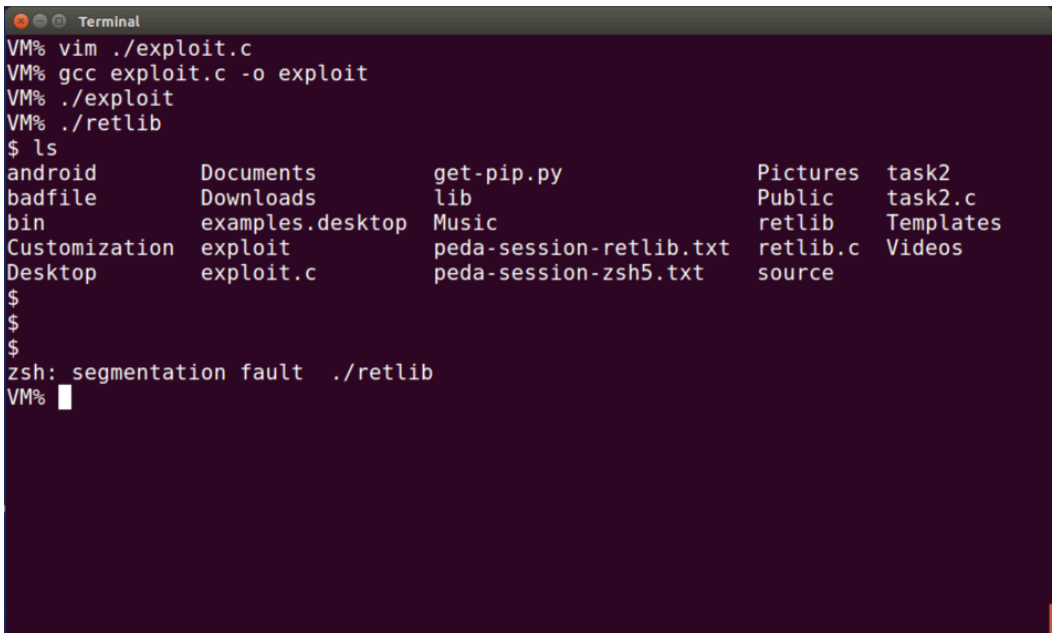


Figure 8: segmentation fault

Attack variation 2:

When we rename the `retlib` to `newreturn`, we cannot exploit successfully. The reason is that the address of `MYSHELL` is changed with the name of the program is changed.

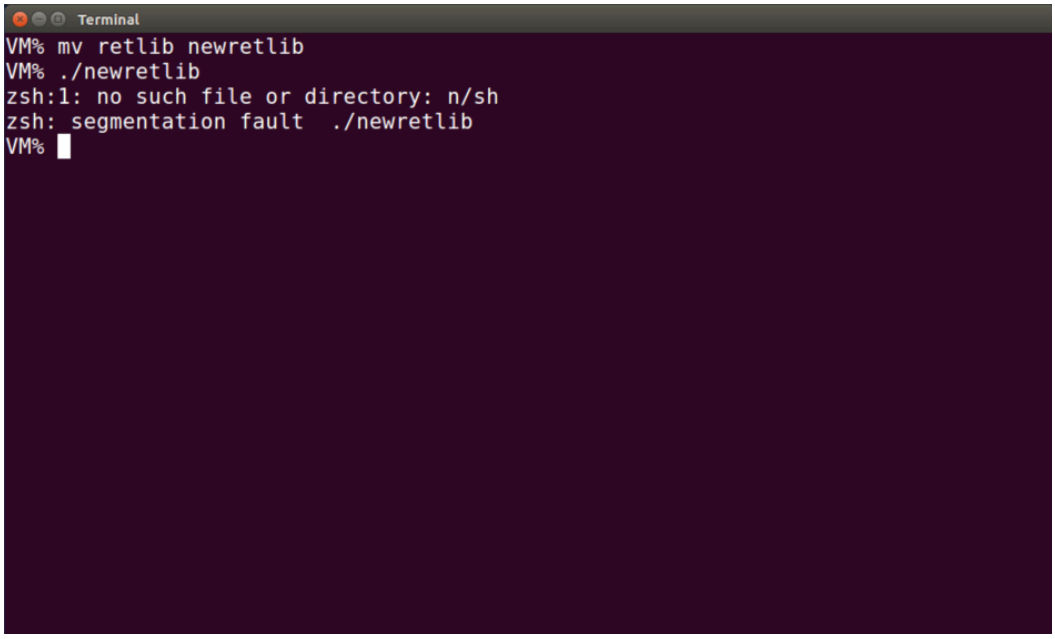


Figure 9: segmentation fault

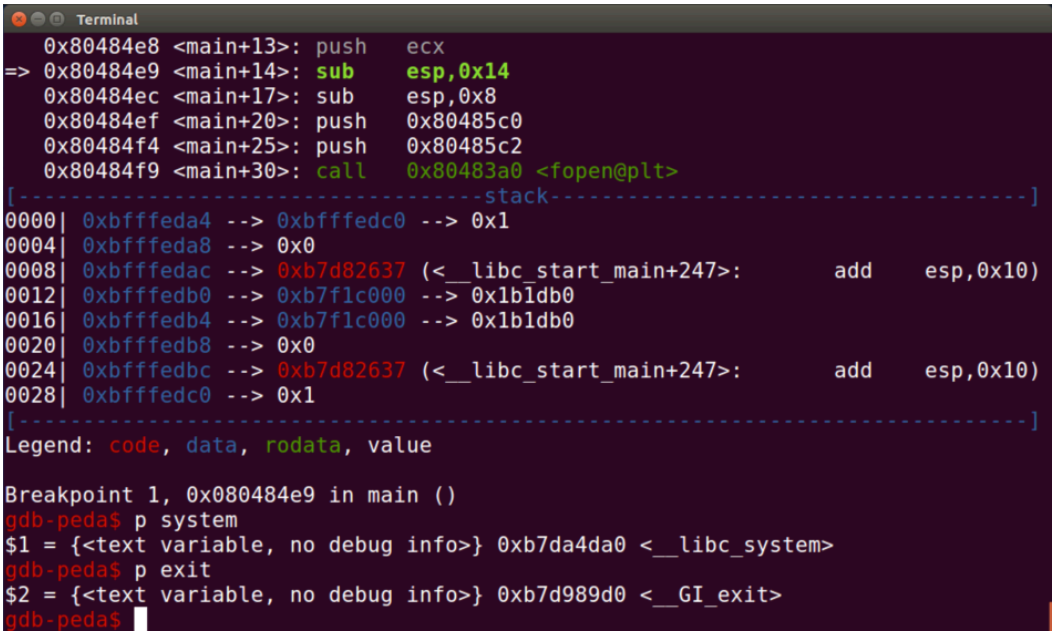
Task 4:

Turning on Address Randomization

We cannot exploit successfully when we turn on the randomization. And we got `segmentation fault` when we repeat task1&2.

The address of `MYSHELL` is changed randomly. `system()` and `exit()` are also changed.

In `exploit.c`, `X, Y, Z` are correct, but `buf[X], buf[Y], buf[Z]` are incorrect.


Terminal
0x80484e8 <main+13>: push ecx
=> 0x80484e9 <main+14>: sub esp,0x14
0x80484ec <main+17>: sub esp,0x8
0x80484ef <main+20>: push 0x80485c0
0x80484f4 <main+25>: push 0x80485c2
0x80484f9 <main+30>: call 0x80483a0 <fopen@plt>
stack
0000| 0xbfffeda4 --> 0xbfffedc0 --> 0x1
0004| 0xbfffeda8 --> 0x0
0008| 0xbfffedac --> 0xb7d82637 (<__libc_start_main+247>: add esp,0x10)
0012| 0xbfffedb0 --> 0xb7f1c000 --> 0x1b1db0
0016| 0xbfffedb4 --> 0xb7f1c000 --> 0x1b1db0
0020| 0xbfffedb8 --> 0x0
0024| 0xbfffedbc --> 0xb7d82637 (<__libc_start_main+247>: add esp,0x10)
0028| 0xbfffedc0 --> 0x1
Legend: code, data, rodata, value
Breakpoint 1, 0x080484e9 in main ()
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7da4da0 <__libc_system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0xb7d989d0 <__GI_exit>
gdb-peda$