# 记录

练习题目录为~/ctf/c3.

虚拟机默认账号是ctf，密码是 ctf。

Root密码是ctftoor.

命令如下：

cd ~/ctf/c3

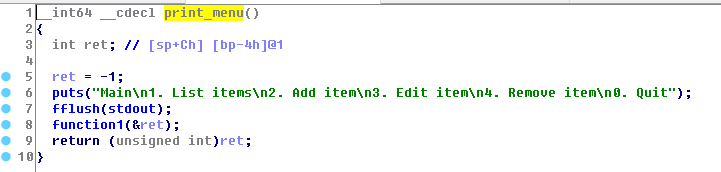
python c3.py

## 1.逆向

程序正常功能为：用户可以管理联系人（名称和电话号）。

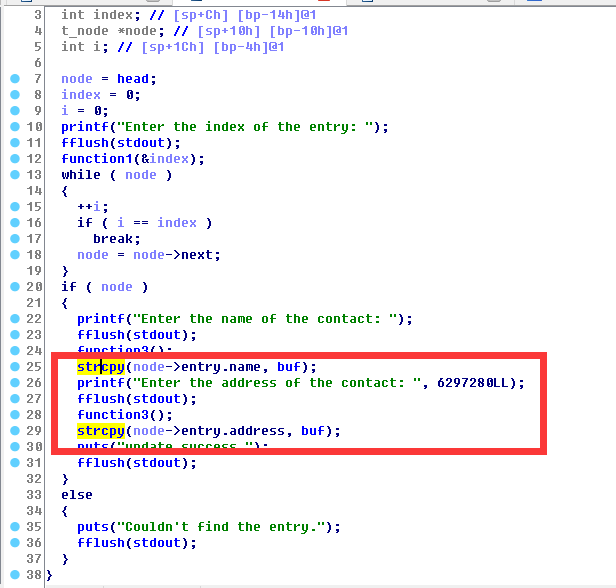
包括以下功能：

1. List items
2. Add item
3. Edit item
4. Remove item
5. Quit



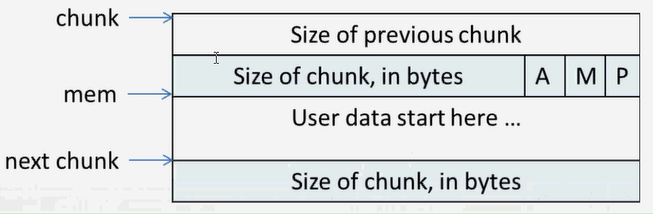
## 漏洞位置

漏洞存在于第3项功能Edit item，在对联系人进行编辑的时候，没有检查缓冲区大小，直接拷贝导致堆溢出。



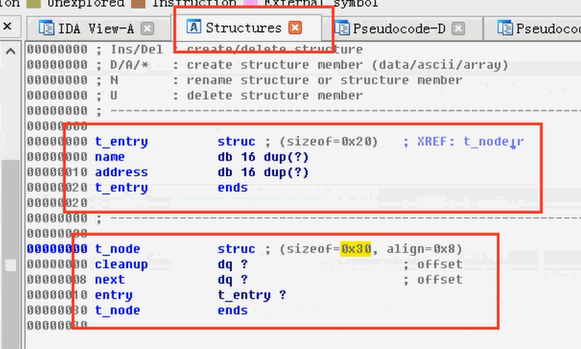
## 堆的结构

64bit程序堆块的结构为：



前16个字节为堆块的管理结构，之后为用户数据。

## 数据结构



Address字段溢出直接覆盖下一个堆块，目标是覆盖下一个堆块的函数指针cleanup。

|  |  |  |
| --- | --- | --- |
| 偏移 | 字段 |  |
| 0 | Cleanup指针 |  |
| 0x08 | Next指针 |  |
| 0x10 | Name指针 |  |
| 0x20 | Address指针 |  |
| 0x30 | 下一个堆块的块首结构 |  |
| 0x40 | 下一个堆块的cleanup函数指针 |  |
|  |  |  |

## 本地调试过程

### 5.1利用zio库，搭建本地调试环境。

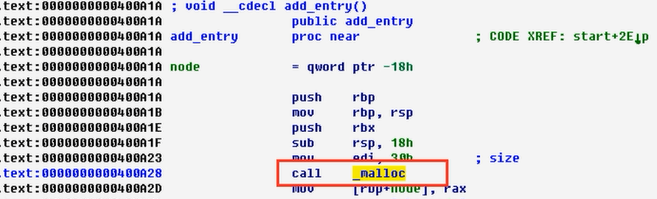




### 5.2 checksec发现没有nx保护，说明栈可以执行。

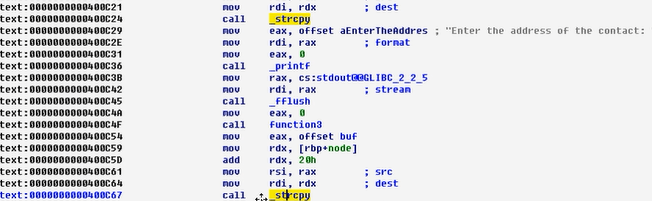


### 5.3 Add\_entry函数中申请堆块后下断点：

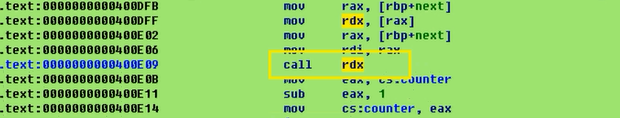


获得两个堆块的地址。

### 5.4 edit\_entry



### 5.5 Remove\_entry



断点：

b \*0x0000000000400A2D

b \*0x0000000000400C67

b \*0x0000000000400C6C

b \*0x0000000000400E09

## 远程环境模拟

accept-fork

每来一个连接，fork一个子进程

通过命令：

socat tcp-l:10000,reuseaddr,fork exec:./c3

## 思路总结：

首先使用add\_item 添加两个node，然后修改第一个node，从而填充第二个node的cleanup的地址指向buffer(0x6016C0 + 2)(buffer固定可控)，之后再调用remove\_entry删除第二个node，并且填入shellcode，因为这个函数中有调用到cleanup，从而就执行buffer中的shellcode。

Tips: buffer地址+2是因为输入remove\_entry(4)的时候index为2，程序输入了’2\x00’+shellcode给buffer所以地址+2.