

CTF 个人赛解题报告

--by siesta (就是那个 Gatekeeper)

2022302181276 潘鸿远

1.sign in

emmm 这个不用讲了吧，去频道一看就有：)

2.rack your brain

看到是 txt 名称先把后缀改了，看到佛又说直接上网搜解密就好了（虽然一开始搞半天不知佛说和佛又说是不同的）

3.小紫

第一步用 winhex 把图片长宽改改就好，但是第二张 bmp 格式编码是反过来的，找长宽找老半天，那个提示误导性也挺强的（不满），找半天图片下面都没 flag，结果在最上面那么一点：（

4.wingman

遇到这种要交互的我都是先读一遍代码然后直接连上去看看，一开始就是破解一个 sha256，但是只是前四位，那就很简单了，sha256 明文由 0~9 10 个数字和 26 个大

小写字母组成，一共 62 个字符，只要四重循环暴力破解就好。

o

```
import hashlib, string, random

#mLHnrRCd
part_proof = '711JI7qyZ5UxDHN50uDqrMguJhcC'
sha = '26c2d150e528218f5592abeec856442e4fa25f25e80e9061760b341637a85409'
table = string.ascii_letters + string.digits
XXXX = ""
print(part_proof)
print(sha)

for i in range(0,10): XXXX+=str(i)
for i in range(65,91): XXXX+=str(chr(i))
for i in range(97,123): XXXX+=str(chr(i))    #生成包含62个字符的字符串

for i in range(0,61):
    for j in range(0,61):
        for k in range(0,61):
            for l in range(0,61):
                t=XXXX[i]+XXXX[j]+XXXX[k]+XXXX[l]    #穷举四个字符
                temp_proof = t + part_proof
                temp_sha = hashlib.sha256(temp_proof.encode()).hexdigest()
                if sha == temp_sha:
                    #io.recvuntil(b"[+] Give Me XXXX :")
                    print(t.encode())
                    #io.sendline(t.encode())
                    #io.sendline(t.encode())
                    break
```

然后就出现了那个 hint 提到的经纬度，应该就是输入正确的经纬度就能得到 flag 了，线索就只有那张图片了，先把图片反转，然后以 google 那个店名和 universal study，当时是找到美国去了（哭），但是后来注意到背景都是亚洲人模样，也找到了日文介绍这个地方，就确定在日本了。然后说这个地方在大阪，又说 she majors in law，现居地应该就是大学了，一个一个试就行了。（第一次做这种题属实把我震惊到了）

5.baby_typing_game

这题我忘的差不多了，我记得好像是直接复制粘贴？

现在也连不上端口了，直接上代码好了：)。

```
b=1
while b<1000: #这段就是重复复制粘贴，前面有一个一模一样的XXXX就不贴上来了
    #print(b, end=':')

    a = io.recvline()
    #print(a)
    io.recv()
    tmp = a.split(b': ')
    ans = tmp[1]
    io.sendline(ans)
    a = io.recvline()
    print(a)
    b=b+1
io.recvline()
a = io.recvline()
print(a)
io.recvline()
```

6.NO COPY

去那个开发者工具把禁用 JavaScript 打开就能 copy 了

7.get_my_number

先看下源代码：

```
scanf("%d", &n);

if (n > 1000)
{
    printf("Am I that old?");
    return 0;
}

if(n > sizeof(int)*1000)
    system("/bin/sh");
```

很明显只要满足 $n > 4000$ 就好了，但是这样前面就直接先

return 0 了，我当时想都没想就觉得这是个整数溢出，直接输了个 2147483648，结果蒙对了（笑），后来才知道这是因为 sizeof 返回无符号整数，只要输个负数让他溢出就可以躲过第一句的判断，下面就自己变成超大数了。（btw 我一直不知道 cat flag 只能在当前目录下用，浪费一个上午搞这个，再此特别鸣谢 nemo 学长~kiss~）

8.get_my_float

还是先看源代码：

```
for (int i = 0; i < 8; i++)
{
    gundam.ch[i] = getchar();
}

if (gundam.fa == 0.618)
{
    printf("Gundam Rising!!");
    system("cat /tmp/flag");
}
```

刚开始看蒙了，思路应该是改变联合体中一个变量的值让另一个也变化，于是上网搜了下联合体（退坑太久已经忘光力），发现两个变量共用一段起始地址，那这样就清楚了，只要输入一定的字符，让他在 double 变量读取的时候变成 0.618 就好了。

然后问题来了，我记得浮点数在内存中存储是有些不同的，然后又搜了一下，发现 double 类型是由 1 位符号位，11

位指数位和 52 位尾数位构成的，然后自己写了个程序去算 0.618 化成二进制（网上没有这么多位的（悲））

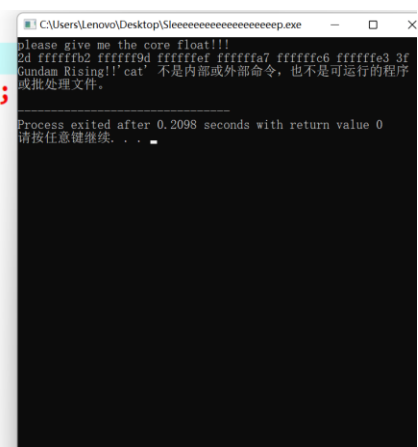
```
double a=0.618;
int n=0;
while(n!=52)
{
    n++;
    if (a*2.0>1.0)
    {
        cout << 1;
        a=a*2.0-1.0;
    }
    else cout << 0,a=a*2.0;
    //cout << ' ' << a << endl;
}
}/*得到结果:
```

10011111111100011110001101010011111101111100111011011001000101100


接着疑惑的事情就来了，我把它转化为 16 进制 0x2cb29defa7c6e33f 赋值给字符数组后输出确实是 0.618，交上去却错了，后来我直接用 if 判断也错了，但显示确实是 0.618，我当时人都傻了，然后去问 nemo 学长，他说有小错，但我算半天都算不出来，最后直接让 fa=0.618 反向输出字符串再赋值就对了，一个大无语。（所以至今不知道我为什么错了（哭））

```
setvbuf(stdin, 0, 2, 0);
gundam.fa = 0.618;
printf("please give me the core float!!!\n");
for (int i = 0; i < 8; i++)
{
    //gundam.ch[i] = getchar();
    //printf("%d ",int(gundam.ch[i]));
    cout << hex << long(gundam.ch[i]) << ' ';
}

if (gundam.fa == 0.618)
{
    printf("Gundam Rising!!");
    system("cat /tmp/flag");
}
```



```
int main()
{
    float_char a;
    a.ch[7]=0x3f;
    a.ch[6]=0xe3;
    a.ch[5]=0xc6;
    a.ch[4]=0xa7;
    a.ch[3]=0xef;
    a.ch[2]=0x9d;
    a.ch[1]=0xb2;
    a.ch[0]=0x2d;
    if (a.fa!=0.618) cout
    cout << a.fa << endl;
}
```



9.hello_net

当时看一下子就有人做出来了，想都不想直接拖进 ida 就拿到 flag 了（唯一一个铜牌 233）

10.maze

先查壳，脱完壳扔进 ida 里得到这么一串东西

```
strcpy(v20, "#####");
strcpy(&v20[11], "**#####");
strcpy(v21, "#####");
strcpy(&v21[11], "**#####");
strcpy(v22, "#####");
strcpy(&v22[11], "**#####");
strcpy(v23, "#####");
strcpy(&v23[11], "#####");
strcpy(v24, "#####");
strcpy(&v24[11], "#####");
```

结合题目看的话，这应该是一个迷宫了，然后走出迷宫就好了，看里面的操作应该就是用 wasd 移动。

```
if ( v14 == 9 && v15 == 9 )  
{  
    v11 = std::operator<<<std::char_traits<char>>>(  
        &std::cout,  
        "Congratulations! Your flag is mayctf{sha256(input)}.",  
        v10);  
    std::ostream::operator<<<(v11, std::endl<char,std::char_traits<char>>);  
}
```

这应该就是走到 (9, 9) 就 ok 了。(不知道为什么我的 ida 显示的迷宫缺少了一点, 后来用了别人的才 ok 了)

11.easy_md5

还是先看一眼源代码:

```
if op == b'1':
    self.magic = os.urandom(64)
    ans = hashlib.md5(self.magic).hexdigest().encode()
    self.send(b"calculate Md5 of it: ")
    self.send(self.magic)
    self.send(b"your answer is: ")
    #self.send(ans)
    tmp = self.register()
    if tmp == ans:
        self.send(b"KANG!!!QIANG!!!")
        self.blood -= random.randint(1, 999999)
    else :
        self.send(b"WRONG!!!")
```

很明显就是来算随机数的 MD5，算对一次扣随机血量。

然后就出现了之前在频道里出现的 1w 次循环，和 mia 聊了之后才发现扣完血不会给 flag，要自己手动退出（麻了），然后又改半天，虽然还是有些 bug 不能判断血量，后来我不管了干脆直接循环个 4000 次把它扣到负血手动退出：


```

sum = 0
while sum<=4000:
    sum = sum + 1
    p.recv()
    p.send(b'1')
    p.recvline()
    a = p.recv()
    tmp = a.split(b' \nyour')
    ans = hashlib.md5(tmp[0]).hexdigest().encode()
    p.send(ans) #算md5并发送
    #p.recv()
    a = p.recvline()
    #print(sum, ':', a)
    p.recv()
    p.send(b'2')
    p.recvline()
    blood = p.recvline().strip().decode()
    print(blood)
    p.recvline()
    ''' if (int(blood)<=0): 就是这里一直出bug
        p.recv() 我明明一行一行打都没问题（委屈）
        p.send(b'3')|
        p.recvline()
        p.recvline()
        p.recvline()
        p.recvline()
        flag = p.recvline()
        print(flag)
        break'''
p.interactive()

```

-----止步于此-----

接下来就是这次比赛的一些感悟了，我之前学过一段时间信息学竞赛，但是对电脑什么的了解仅限于普通人水平，本来是不想打的，但是在舍友的劝说下还是来当炮灰了，打一周比赛是真的累（哭），但还是收获颇丰：

1. 学会了一点 python (写了一周的 py 感觉就是以后都不想写了)
2. Pwntool yyds! 这个写交互真的超方便的好不好 (重点错)
3. Nemo 学长人超好啊啊啊啊啊, nemo 学长我真的好喜欢你啊, 为了你, 我要学 pwn! 斯哈斯哈! (尖叫) (扭曲) (阴暗的爬行) (爬行) (扭动) (阴暗地蠕动) (翻滚) (激烈地爬动) (扭曲) (痉挛) (嘶吼) (蠕动) (阴森的低吼) (爬行) (分裂) (走上岸) (扭动) (痉挛) (蠕动) (扭曲的行走) (不分对象攻击)
4. 等我能出题了我也要夹带私货 (