TSCTF-2017 AC 自动机 writeup

神秘的文件：

看到流量包看到有两个文件，

```
56802    21 Request: TYPE I
56802    21 Request: SYST
56800    21 Request: SYST
56802    21 Request: STOR tsctf.txt
56802    21 Request: STOR FLAG.zip
56802    21 Request: QUIT
```

找到 ftpdata:

```
0  SSDP       216   52158   1900 M-SEARCH * HTTP/1.1
   FTP-DATA   138   58425     20 FTP Data: 84 bytes
   FTP-DATA   259   57184     20 FTP Data: 205 bytes
   TCP         66   58425     20 58425 → 20 [SYN, ACK] Seq=0 Ack=
```

将第二个数据包的内容到处保存为 flag.zip

第一个数据包的内容是是第二个数据包的密码，

```
> Internet Protocol Version 4, Src: 192.168.222.139, Dst: 192.168.222.1
> Transmission Control Protocol, Src Port: 58425 (58425), Dst Port: 20 (20), Seq: 1, Ack: 1, Len: 84
  FTP Data (MD5(pass) = 24885fdab795c41166d6f0067782dc9f\r\n\r\npass = ['a','h','k','q','y','$','%'])
```

密码的 MD5 值为上面的 MD5 值，直接跑脚本得到解压密码为：

```
C:\Users\sunshine>pyt
ah%kyq$
```

解压得到 FLAG.txt：然后再进行 base64 解码得到 flag:

FLAG.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
TSCTF{dHNjdGZfcmV2b2x1dG1vb18yMDE3X01heQ==}
```

```
d\xfdnN\xc6K\xcc}\xad
>>> base64.b64decode('dHNjdGZfcmV2b2x1dG1vb18yMDE3X01heQ==')
'tsctf_revolution_2017_May'
>>>
```

EasyCrypto:

直接求加密的逆过程 ：先对 cyper_text 进行 base64 解码，然后进行与加密相同的格式转换，然后进行加密的逆过程得到 flag。

```
import struct
import base64

cypher_text = 'DgYiZFttExBafXJPPn8BNhI9cwEhaUMgPmg+IA=='

iv = struct.unpack("I", 'x1a0')[0]
print 'iv is ', hex(iv)

def crypto(data):
    return data ^ data >> 16

def decode(cypher, iv):

    datas = []
    flag = ''

    for i in range(1, 7):

        datas += [crypto(cypher[i]) ^ cypher[i-1] ]

    datas = [crypto(cypher[0]) ^ iv] + datas

    for i in range(7):
        flag += struct.pack("I", datas[i])
        tmp = struct.pack("I", datas[i])
        print str(hex(ord(tmp[0]))) + ' ' + str(hex(ord(tmp[1]))) + ' ' + str(hex(ord(tmp[2]))) + ' ' + str(hex(ord(tmp[3]))) + ' '
    print flag

flag = base64.b64decode(cypher_text)

datas = []
datas = struct.unpack("I" * (len(flag) / 4), flag)
print datas

decode(datas, iv)
```

```
C:\Users\sunshine>python C:\Users\sunshine\Desktop\easycryto.py
iv is  0x30613178
(1679951374, 269708635, 1332903258, 906067774, 24329490, 541288737, 540960830)
0x54 0x53 0x43 0x54
0x46 0x7b 0x31 0x74
0x73 0x5f 0x61 0x5f
0x65 0x34 0x73 0x79
0x5f 0x43 0x72 0x37
0x70 0x74 0x30 0x21
0x21 0x21 0x7d 0x0
TSCTF{1ts_a_e4sy_Cr7pt0!!!}
```

密码学穷举攻击：

分析发现实质就是一个 RSA，其中 c1,c2,n1,n2 都知道，c1=m1^e(mod n1),c2=m2^e(mod n1),e = m1^m2(mod n2),

因为 RSA 中 ed = 1(mod Q(n1))；

N1 = 0Xffffffff 的欧拉函数就等于它的素因子减一相乘，n1 的素因子很容易得到：

```
C:\Users\sunshine\Desktop\dsd.exe
4294967295
3
5
17
257
65537
```

所以通过枚举e ,然后计算出d ,就相当于得到了RSA的私钥 ,就可以解出m1,m2;

当 e = m1^m2(mod n2)这个条件满足时的 m1,m2 就为正确解。

密码穷举攻击.cpp [*] dsd.cpp

```cpp
unsigned long long  phi = 2147483648ull;

ull c1 = ( 0x85ba527aull + 0x8393f16eull) >> 1 , c2 = (0x85ba527aull - 0x8393f16eull) >>

int main()
{
    for(unsigned long e =3;e < n2 ;e++)
    {
        if (e % 10000 == 3)
        {
            cout << "e = " << e << endl;
        }
        unsigned long long d = inv(e, phi);

        unsigned long long  m1 = pow_mod_n(c1, d, n1);

        unsigned long long m2 = pow_mod_n(c2, d, n1);

        if (e == (m2 ^ m1) % n2)
        {
            cout << m1 << ' ' << m2 <<endl;
            return 0;
        }
    }
}
```

```
e = 4050003
e = 4060003
e = 4070003
e = 4080003
e = 4090003
e = 4100003
4120554291 2611229368

--------------------------------
```

```
深思数盾 CTF 2017 CrackMe
Please input serial:f59aab339ba432b8
Congratulations, registration successful!


Process returned 0 (0x0)   execution time : 112.459 s
Press any key to continue.
```

Baby_android:

反编译后看到源码，就是输入以 TSCTF{开头}结尾的一串只包含 0-9a-f 的 32 个字符组成的字符串，通过计算使得计算后的字符串的等于已知的一个字符串：

```
public class MainActivity extends AppCompatActivity {
    public MainActivity() {
        super();
    }

    public boolean check(String flag) {
        boolean v7 = false;
        if((flag.startsWith("TSCTF{")) && (flag.endsWith("}"))) {
            String v3 = flag.substring(6, flag.length() - 1);
            if(v3.length() == 32 && (Pattern.compile("[0-9a-f]+").matcher(((CharSequence)v3)).matches())
                    ) {
                byte[] v4 = v3.getBytes();
                int v1 = 0;
                String v6 = "";
                while(v1 < v4.length) {
                    int v0 = v4[v1];
                    v6 = v0 > 102 || v0 < 97 ? v6 + (v0 - 48) : v6 + (v0 - 87);
                    ++v1;
                }

                if(!v6.equals("11928116108151591468529124390810231301615 13")) {
                    return v7;
                }

                v7 = true;
            }
        }

        return v7;
    }
```

通过分析 v6 只能由 10-15 和 0-9 之间的数组合起来，所以直接将 11928116108151591468529124390810231301615 13 分解得到 flag：

```
11928116108151591468529124390810231301615 13"

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

b928b6a8ff9e68529c43908a23d016fd
```

CODING-小明二进制：

递归，每一位判断能否减 1。最后计算个数。

用 pwntools 链接服务器。

CODING-泽哥的算术：

快速幂取模。

CODING-Las Vegas：

博弈题，kB+k 是必败局面。数据排除了 kB+k 的初始局面。

CODING-修路：

简单的并查集。

MISC-zipcrc：

首先 crc 碰撞出 k1,k2,k3。解压时 k1+k2+k3 不对 ,卡了好久最后爆破出来了。

其实解压密码是 k3+k2+k1。

之后拿到加密算法，加密算法其实也是解密算法。已知盐，密文，带入加密算法就得到明文。

在原文件中添加 decode 函数。

```
def decode(data, key=key, decode=base64.b64decode, salt_length=16):

    data = decode(data)

    salt = data[0:16]
    datas = data[16:]

    print crypt(datas, sha1(key + salt).digest())


print decode(ctMessage)
```

PWN-ascii：

逆向分析可知，这是栈溢出漏洞，checksec 发现无保护措施。但要求 shellcode 每一个字符都大于 32.百度纯 ascii shellcode，发现 msf 就可以构造。

https://www.offensive-security.com/metasploit-unleashed/alphanumeric-shellcode/

执行指令

msfvenom –a x86 –platform linux –p /linux/x86/shell/exec COM /bin/sh –e x86/alpha_mixed –f python

然后得到 payload，使用 pwntools 提交，进入交互模式即可。

REVERSE-take it easy：

Ida+od，搞清验证规则，拟过程是，

v8 -> ^= v37 –> 移位变换。

写了一个 python 的脚本，v8 和 v37 从栈上 dump 下来。跑出 flag

```
for x in range(len(fi[0])):
    print 'xor:' + str(hex(ord(fi[0][x]))) + '^' +  str(hex(ord(fi[1][x])))
    tmp += chr(ord(fi[0][x]) ^ ord(fi[1][x]) )

print 'tmp:%s' % tmp

for x in range(len(tmp)):

    print hex(ord(tmp[x]))

print '-------------------------'
dic = {}
for i in range(256):

    dic[(i >> 2) + ((i&3) << 6)] = i


flag = ''
flag_hex = ''

for x in range(len(tmp)):

    flag += chr(dic[ord(tmp[x])])
    print hex(dic[ord(tmp[x])])

print 'flag: %s' % flag

print flag_hex

print dic
```

异或 v8 和 v37 -> 构造逆向的映射 –> 映射出 flag。

WEB-web

配置文件写入漏洞，没思路，根据以前的经验，果断去 p 师傅博客找一找，找到一篇文章。

直接用\’;phpinfo();// ,发现能用。之后输入\’;`cat ../flag.php`// 拿到 flag。

WEB-Simple-Shop

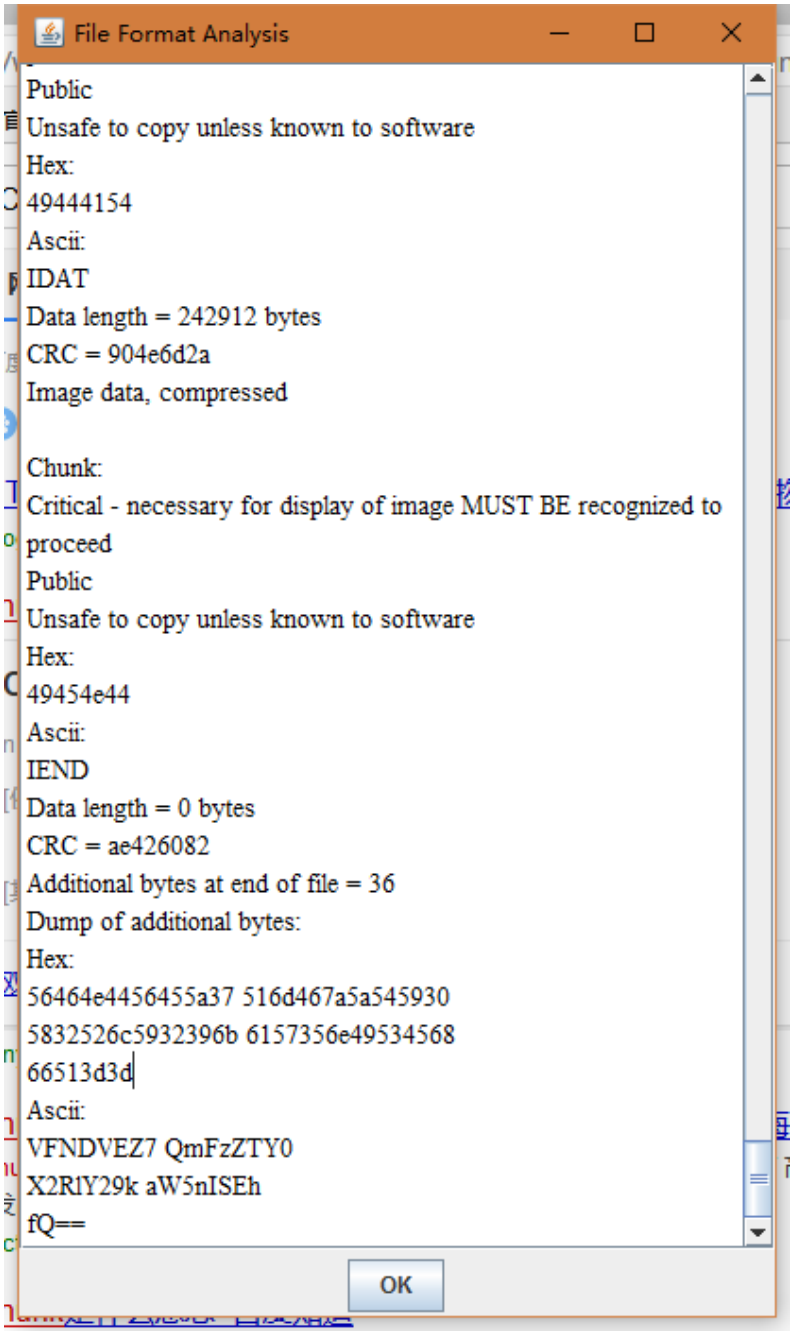
发现重置密码会回复指定用户的 sid，而买东西需要用到这个 sid，于是就能用 TOM 的账户买东西了。

签到

签到就不说了，按提示来就是了

Logo

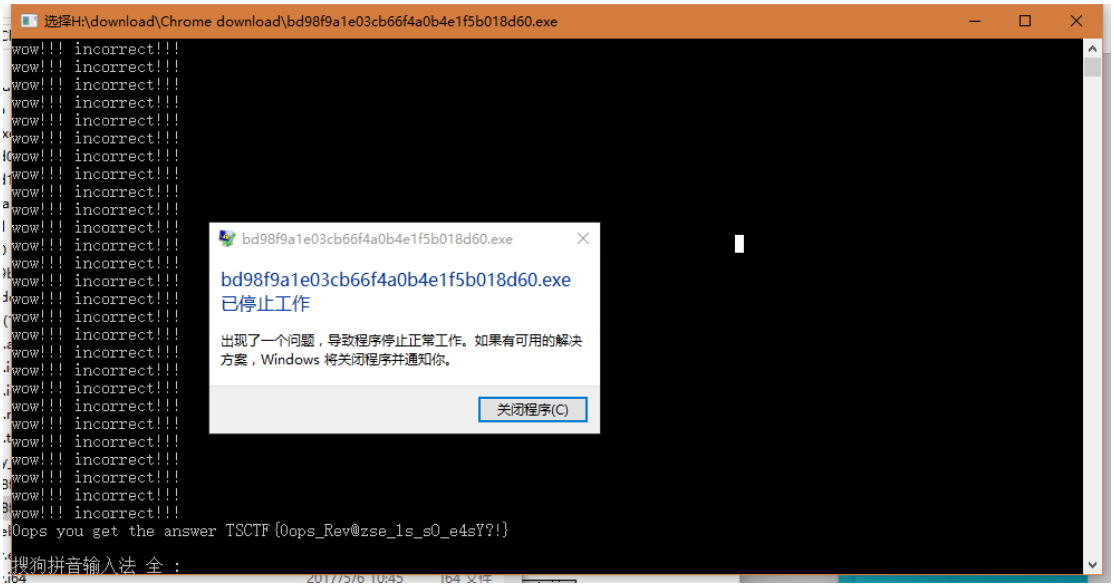
将图片放到 Stegsolve 分析一下 然后发现隐藏信息 base64 解密



VFNDVEZ7QmFzZTY0X2RlY29kaW5nISEhfQ==
TSCTF{Base64_decoding!!!}

Checkin：

随便输入一段长字符串，就溢出了，快速拿到 flag，都不用扔到 ollydbg 里看呢 !!!∑(ﾟДﾟノ)ノ



四维码：

Gif 打开随便一扫 2 字母，然后扫每一帧得到一个指向 Twitter 的网址，是一个三维码，扫描出来可以得到一段字符串，base64 解码以后是 key=qrcode 然后就卡住了 第一波 hint 放出来以后，Google 搜图得到了一个网址，进去发现是一个隐藏工具，还给的 GitHub 的地址，读了一下，然后用 qrcode 作为 password 和 macpassword 解开得到一个 secret.txt 和一个二维码 二维码处理一下可以扫出来一串 01 字符串。然后就卡这里 you 卡好久，因为用工具 macpassword 显示 HMAC :FALSE 就一直纠结 macpassword 是不是错了，随意更换 macpassword 得到的图片和 txt 不变，一直在纠结。然后 01 字符串转为 ASCII 码提交也不对，一直想不出来，后来思考了一下，四维变三维，三维变二维，是不是也是二维变一维（条形码），队友开始写代码，我前天晚上刚刚

做过 256*256 个 01 字符串转为二维码，遂将 01 字符串复制到记事本不换行，然后复制粘贴了多行。字体调到使内容一个屏幕可以看到的合适的大小，0 用空格替换掉 1 用加黑的竖线替换掉。然后万能的淘宝一扫就得到了结果。