
# 华山杯 WRITEUP

YHZX_2013 Afternoontea
boo0m bluecake
watchdog yinhuochong
hyrathon Rocky
pinko blueeternal
carter

2016/9/10

# Crypto

## 紧急报文

将 crypto.txt 内容用 base64 解谜，得到加解密所用矩阵

请将要加密或解密的内容复制到以下区域

```
 A D F G  X
 ----------------
A | p h q g  m
D | e a y n  o
F | f d x k  r
G | c v s z  w
X | b u t i/j l
```

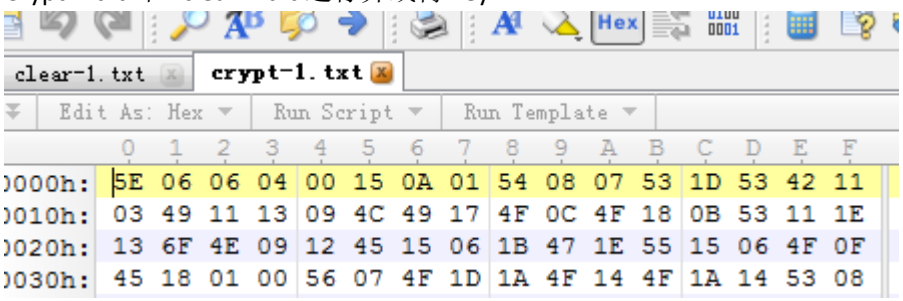
再对应题目所给的内容，解密即可：

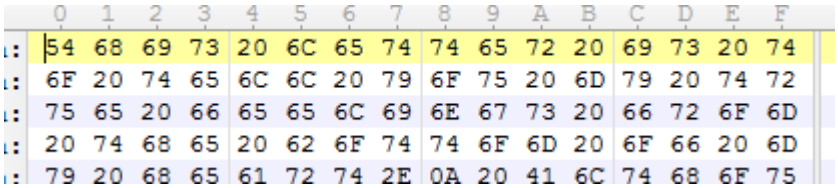
FA XX DD AG FF XG FD XG DD DG GA XF FA

f l a g x i d i a n c t f

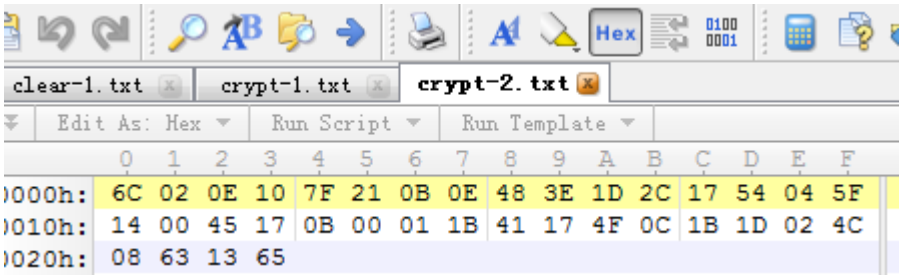
## Is it x or z ?

Crypt-1.txt 和 clear-1.txt 逐行异或得 key




然后 key 和 crypt-2 异或得 flag



# 分组加密模式检测

分组密码加密模式有：EBC、CBC、OFB、CFB 等，其中可以用来作为此处考点的多半是 EBC 模式。将 challenge.txt 文件内容 base64 解码，得到 205 个长度为 320 字节的密文。

考虑到 EBC 模式的特点，即相同的明文和相同的秘钥得到相同的密文，所以要在 205 组密文中，找到一组使用 EBC 模式加密的分组，这个分组的特点就是：在 320 字节中，按照 16 字节一组进行划分，可以划分为 20 个小块；在这 20 个小块中，至少有其中两个小块的内容相同。编程实现如下：

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

offset = 16
def cut(str1):
    res = []
    length = len(str1)
    num = length/offset
    for i in range(num):
        res.append(str1[offset*i:offset*(i+1)])
    return res

while 1:
    line = f.readline()
    line_cut = cut(line)

    for item in line_cut:
        index = line_cut.index(item)
        del line_cut[index]
        if item in line_cut:
            print item
            print line
            break
            break
    if not line:
        break

f.close()
```

找到使用 EBC 模式工作的分组

```
carter@ubuntu:~/Desktop/hsb/crypto/3$ python solve.py
08649af70dc06f4f
d880619740a8a19b7840a8a31c810a3d08649af70dc06f4fd5d2d69c744cd283e2dd052f6b641dbf
9d11b0348542bb5708649af70dc06f4fd5d2d69c744cd2839475c9dfdbc1d46597949d9c7e82bf5a
08649af70dc06f4fd5d2d69c744cd28397a93eab8d6aecd566489154789a6b0308649af70dc06f4f
d5d2d69c744cd283d403180c98c8f6db1f2a3f9c4040deb0ab51b29933f2c123c58386b06fba186a
```

## 修复一下这份邀请函的部分内容

直接从 crypt.txt 中读取到 flag

```
We hereby sincerely invite you and
your company representatives to
attend our IT conference .
In this conference there will be
many top managers of IT industry
 and many topics will be talked
during the conference,this is related
to the future of ITindustry.And the
main purpose of this conference is to
give you more ideas on IT business view
 here.At the same time to try implementing
 the agreement,which is under discussion for some time.
We are looking for your attending
flag xie can xie yu hen xing gao
your answer is right  so really!please submit above
```

# Android

## 错错错

题目很简单，先生成一段随机字符串，再生成一个 4 以内的数字用于指定 hash 算法，然后做一次仿射加密，最终的结果作为序列号密钥

## 寻找密码

本题实现了一个简单的 apk 壳，加壳后的 dex 文件结构为壳 dex+payload.apk+payload 长度(四字节)，其中，payload.apk 需要与 255 进行异或才能得出原始的 payload.apk 文件

payload.apk 中要求输入 username 和 password，username 与 She11_N6Rc 忽略大小写一致，username 使用 sha1 计算 hash 值后的前 16 个字节作为 password

## 神奇的 ZIP

首先拿到的 apk 坏了，拖到 linux 下 zip -ff testndk4_Signed4.zip 修复一下，第一个知识点完成.

题目是纯的 Java 加上存的 JNI, Java 下没有需要分析的逻辑.

首先需要绕过一处一定会退出的检查，修改 so 文件里的 move r0,0 为 move r1,1 强行返回 1. 或者直接动态调在此处下断点改寄存器过去.

```
super.onCreate(arg5);
        this.setContentView(2130903065);
        if(this.isExit()) {
            this.startActivity(new Intent(((Context)this),
MainActivity.class));
        }
        else {
            Timer v0 = new Timer();
            b v1 = new b(this);
            Toast.makeText(((Context)this), "抱歉，请先获得权限，再进入！！",
0).show();
            v0.schedule(((TimerTask)v1), 5000);
        }
```

题目有个问题对动态调试没有做任何限制,因此在最终需要比较的密码处下个断点看内存即可发现 flag.

```
int __fastcall Java_com_example_testndk4_MainActivity_encodePassword(int
a1)
{
  int v1; // r5@1
  const char *v2; // r7@1
  char *src; // ST04_4@1
  char *v4; // ST04_4@1
  char *v5; // r0@2
  const char *v6; // r1@2
  int result; // r0@4
  char v8; // [sp+8h] [bp-58h]@1
  char v9; // [sp+14h] [bp-4Ch]@1
  char s; // [sp+28h] [bp-38h]@1
  int v11; // [sp+44h] [bp-1Ch]@1

  v1 = a1;
  v11 = _stack_chk_guard;
  v2 = (const char *)Jstring2CStr();
  j_j_memcpy(&v9, "thinkingInAndroid", 0x12u);
  src = (char *)encodePS(&v9);
  j_j_memset(&s, 0, 0x1Au);
  j_j_strcpy(&s, src);
  v4 = (char *)encodePS(&s);
  j_j_memset(&v8, 0, 0xAu);
  if ( j_j_strcmp(v2, v4) )
  {
    v5 = &v8;
```

```
    v6 = "Sorry!";
  }
  else
  {
    v5 = &v8;
    v6 = "Sucess!";
  }
  j_j_strcpy(v5, v6);
  result = (*(int (__fastcall **)(int, char *))(*(_DWORD *)v1 + 668))(v1,
&v8);
  if ( v11 != _stack_chk_guard )
    j_j___stack_chk_fail(result);
  return result;
}
```

看源码或者 F5 都可以发现程序是对 thinkingInAndroid 字符串做了复杂的运算,但是用户输入没有参与进来,最后将用户输入与复杂预算的结果比较, 直接在那里下断点输出:

```
MOVS    R0, R7          ; s1
LDR     R1, [SP,#0x60+src] ; s2
--> set bp here :BL      j_j_strcmp
CMP     R0, #0
BNE     loc_1042
```

定位到内存,得到 flag:lxienietIeAehfyih

## 顺藤摸瓜

单纯的 jni 逻辑分析, java 代码只涉及一点.

```
int __fastcall Java_com_example_demo_MainActivity_check(int a1, int a2, int
a3)
{
  int v3; // r4@1
  int v4; // r6@1
  int v5; // r7@1
  void *v6; // ST08_4@1
  int v7; // r0@1
  int v8; // r7@1
  signed int v9; // r6@1
  void *v10; // r5@2
  signed int v11; // r0@4
  signed int i; // r3@4
  int result; // r0@7
  void *src; // [sp+8h] [bp-2F0h]@1
  char dest[56]; // [sp+14h] [bp-2E4h]@4
```

```
  char v16[200]; // [sp+4Ch] [bp-2ACh]@4
  char v17; // [sp+114h] [bp-1E4h]@4
  char s; // [sp+1DCh] [bp-11Ch]@4
  int v19; // [sp+2DCh] [bp-1Ch]@1

  v3 = a1;
  v4 = a3;
  v19 = _stack_chk_guard;
  v5 = (*(int (**)(void))(*(_DWORD *)a1 + 24))();
  v6 = (void *)(*(int (__fastcall **)(int, const char *))(*(_DWORD *)v3 +
668))(v3, "GB2312");
  v7 = (*(int (__fastcall **)(int, int, const char *, const char
*))(*(_DWORD *)v3 + 132))(
        v3,
        v5,
        "getBytes",
        "(Ljava/lang/String;)[B");
  v8 = (*(int (__fastcall **)(int, int, int, void *))(*(_DWORD *)v3 +
136))(v3, v4, v7, v6);
  v9 = (*(int (__fastcall **)(int, int))(*(_DWORD *)v3 + 684))(v3, v8);
  src = (void *)(*(int (__fastcall **)(int, int, _DWORD))(*(_DWORD *)v3 +
736))(v3, v8, 0);
  if ( v9 <= 0 )
  {
    v10 = 0;
  }
  else
  {
    v10 = j_j_malloc(v9 + 1);
    j_j_memcpy(v10, src, v9);
    *((_BYTE *)v10 + v9) = 0;
  }
  (*(void (__fastcall **)(int, int, void *, _DWORD))(*(_DWORD *)v3 +
768))(v3, v8, src, 0);
  j_j_memset(v16, 0, 0xC8u);
  j_j_memset(&v17, 0, 0xC8u);
  j_j_memset(&s, 0, 0x100u);
  j_j_memcpy(dest, &unk_2454, 0x38u);
  v11 = j_j_strlen((const char *)v10);
  for ( i = 0; i < v11; ++i )
    v16[i] = *((_BYTE *)v10 + i) + 97 - dest[4 * i];
  v16[v11 & (~v11 >> 31)] = 0;
  n1(v16, "nbrcdpassword", &v17);
  n2(&v17, &s);
```

```
  result = (unsigned int)j_j_strcmp(&s, "74058473948333034392948223 34") <=
0;
  if ( v19 != _stack_chk_guard )
    j_j___stack_chk_fail(result);
  return result;
}
```

先贴一下 F5，其中没有用的逻辑很多，基本就是如何如何从 java 拷贝到 c 申请内存然后到处挪.ida 改的变量名没保存.变量里边 dest 有点用从内存一遍读取了一段加密向量. 然而这些都没有卵用.....

```
# 我们白白做了这么多工作
#!/bin/env python
#coding:utf-8
import string

iv1 =
[0x3F,0x4D,0x6C,0x5B,0x54,0x5B,0x6C,0x5B,0x54,0x46,0x38,0x46,0x3F,0x1C]
iv2 = "nbrcdpassword"
#iv2_alpha = [ord(x) - ord('a') for x in iv2]


final = "74058473948333034392948223 34"
#byte1 = [int(final[x:x+2:][::-1]) + 72 - 97 for x in range(0, len(final),
2)]
byte1 = map(int, list(final))
print byte1

#print string.printable
for i in range(14):
        for k in
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_":
                k1 = ord(k) - iv1[i] + 0x61 - ord('a')
                k2 = (k1 + ord(iv2[i % len(iv2)]) - ord('a'))%26 + 97 - 72
                if k2 % 10 == byte1[2*i] and (k2/10)%10 == byte1[2*i+1]:
                        print k,
        print
```

因为我们可以从 java 逻辑中发现 jni 做不做只影响判断，对输出没有影响.....

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.requestWindowFeature(1);
        this.setContentView(2130903065);
        this.D_text = this.findViewById(2131296319);
        String v0 = this.D_text.getText().toString();
```

```
        String v3 = "";
        int v2;
        for(v2 = 0; v2 < v0.length(); ++v2) {
            try {
                v0.charAt(v2);
                v3 = String.valueOf(v3) + (((char)(v0.charAt(v2) - 8)));
            }
            catch(Exception v1) {
                v3 = String.valueOf(v3) + v0.charAt(v2);
            }
        }

        this.D_text.setText(((CharSequence)v3));
    }
}
```

上面的代码，当 jni 执行了或者被绕过之后，逻辑是对一段 R 中的烫烫烫进行了移位. 砜夼圸烁 B 夼皑卟跷:吒 + 8 = 西电的校址会面 = flag

# Misc

## Try Everything

用 binwalk 提取了一堆文件

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 107 | 116 | 125 | 134 | 143 | 152 | 161 | 23 | 32 | 41 | 50 | 6 | 69 | 78 | 87 | 96 |
| 1 | 108 | 117 | 126 | 135 | 144 | 153 | 162 | 24 | 33 | 42 | 51 | 60 | 7 | 79 | 88 | 97 |
| 10 | 109 | 118 | 127 | 136 | 145 | 154 | 163 | 25 | 34 | 43 | 52 | 61 | 70 | 8 | 89 | 98 |
| 100 | 11 | 119 | 128 | 137 | 146 | 155 | 17 | 26 | 35 | 44 | 53 | 62 | 71 | 80 | 9 | 99 |
| 101 | 110 | 12 | 129 | 138 | 147 | 156 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | 90 | output |
| 102 | 111 | 120 | 13 | 139 | 148 | 157 | 19 | 28 | 37 | 46 | 55 | 64 | 73 | 82 | 91 | sol.py |
| 103 | 112 | 121 | 130 | 14 | 149 | 158 | 2 | 29 | 38 | 47 | 56 | 65 | 74 | 83 | 92 | |
| 104 | 113 | 122 | 131 | 140 | 15 | 159 | 20 | 3 | 39 | 48 | 57 | 66 | 75 | 84 | 93 | |
| 105 | 114 | 123 | 132 | 141 | 150 | 16 | 21 | 30 | 4 | 49 | 58 | 67 | 76 | 85 | 94 | |
| 106 | 115 | 124 | 133 | 142 | 151 | 160 | 22 | 31 | 40 | 5 | 59 | 68 | 77 | 86 | 95 | |

将每个文件的首字母提取连接，得到如下结果：

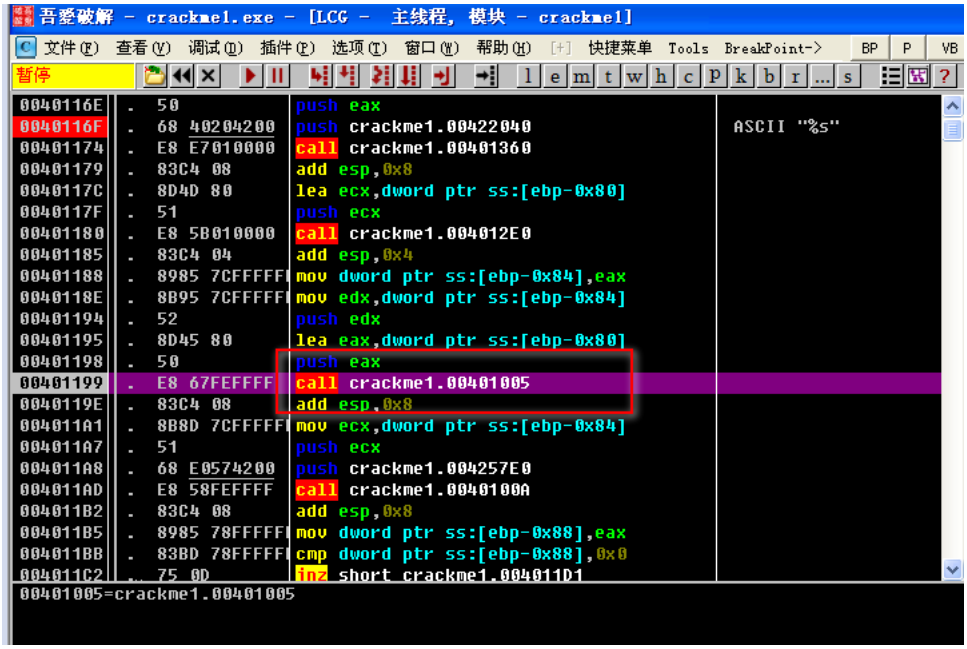
```
Since the eavesdropper have obtained our communication key, therefore we have ad
opted a new communication program.This is our new key:flag_Xd{hsh_ctf:4Ea9F16bA8
b@c}carter@ubuntu:~/Desktop/hsb/misc/1/_test.gz.extracted/_test.extracted$
```

将 s 改成 S 即是 flag

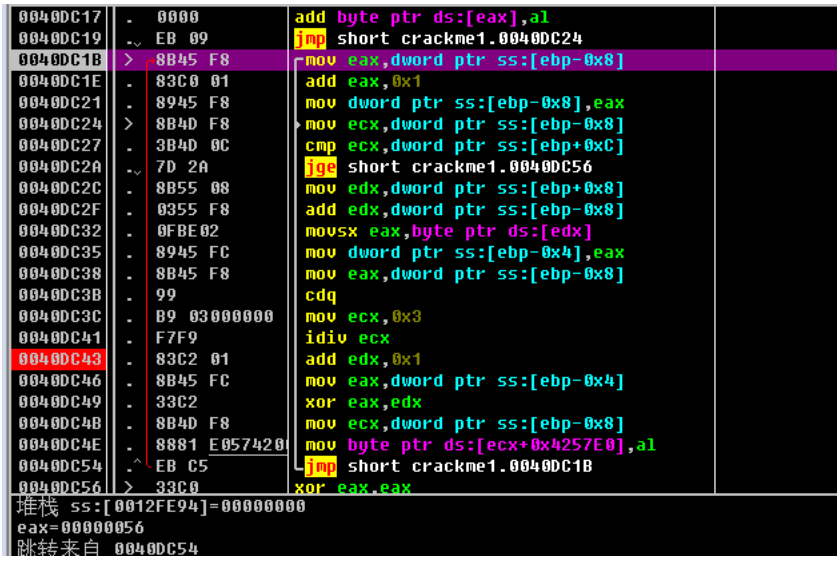
# Reverse

## warming up

关键函数在 sub_00401005 处，由于 IDA 里面加了混淆导致 F5 无法反编译，所以用 OD 动态调试，F7 跟进该函数：




单步到如下位置：




此处是对输入的字符串作异或操作，分别与 0x1、0x2、0x3 进行异或，用 IDA 找到 sub_4010B0 函数，发现将结果与字符串 VgobmndVlBVE 进行比较，所以得出 flag：

```
a="VgobmndVlBVE"
flag = ""
for i in range(len(a)):
```

```
        flag += chr(ord(a[i])^(i%3+1))
print flag
```

# 到手的钥匙

Admin xdamin 并没有卵用，关键函数

```
LABEL_6:
      v15 = 0;
      goto LABEL_8;
    }
  }
  v15 = -v0 | 1;
LABEL_8:
  v11 = v15;
  if ( v15 )
  {
    print((int)aYouHaveOneMore, v4[0]);
    sub_401340();
LABEL_24:
```

所以输 3247,5569 会蹦出密钥

```
__int32 sub_401340()
{
  __int32 v0; // ST0C_4@1
  __int32 result; // eax@1
  char v2; // [sp+0h] [bp-14h]@1
  signed int k; // [sp+8h] [bp-Ch]@7
  int i; // [sp+Ch] [bp-8h]@3
  int j; // [sp+10h] [bp-4h]@5

  v0 = sub_402DA7(user_416300);
  result = sub_402DA7(pass_41630C);
  v2 = result;
  if ( v0 == 3247 && result == 5569 )
  {
    for ( i = 0; i < 6; ++i )
    {
      for ( j = 0; j < 8; ++j )
      {
        for ( k = 0; k < 30; ++k )
        {
          if ( j >= 7 )
          {
            sub_401090(j, i);
            break;
          }
          print((int)a_, v2);
        }
      }
      result = i + 1;
    }
  }
  return result;
}
```

```
Input the correct user name:3247
Input the correct password:5569
You have one more shot...
.......................................................................
.......................................................................
.....................................................685b...............
.......................................................................
.......................................................................
.........................428b.........................................
.......................................................................
.......................................................................79
db.....................................................................
.......................................................................
.....................................................bccb...............
.......................................................................
.......................................................................
...........................4b1e........................................
.......................................................................
.......................................................................
baa4_
```

最后，flag 格式能提前说明白嘛？坑了好久

## 忘记用户名

关键 check，输入的字符 ascii 逐个减 0-6 与 key 做比较

```
    v5 = "user name must be at least five.\n";
LABEL_9:
    sub_401940(std::cout, v5);
    return 0;
  }
  if ( input_len )
  {
    if ( input_len > 0 )
    {
      do
      {
        if ( *((_BYTE *)&v9 + i) != i + *(&input + i) - input_len )
          break;
        ++i;
      }
      while ( i < input_len );
    }
    if ( i == input_len )
    {
      v5 = "good job!\n";
```

关键 key

```
  memset(&Dst, 0, 0x63u);
  v9 = 0x766F4C49;
  v10 = 0x445865;
  memset(&v11, 0, 0x5Cu);
```

## 探囊取物

告诉我这是 re？抠出来的关键数据

clear-1.txt | crypt-1.txt | crypt-2.txt | **reeeeeee3.bin**

Edit As: Hex | Run Script | Run Template

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 0123456789ABCDEF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 010h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 020h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 030h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 040h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 050h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 060h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 070h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 080h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 090h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 0A0h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 0B0h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 0C0h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000000000000000 |
| 0D0h: | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 0000000110000011 |
| 0E0h: | 30 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 30 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 0011111100111111 |
| 0F0h: | 31 | 31 | 30 | 30 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 1100011111100000 |
| 100h: | 31 | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 30 | 1110000111111110 |
| 110h: | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 1100000110111111 |
| 120h: | 31 | 31 | 30 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 30 | 30 | 30 | 30 | 1100111111110000 |
| 130h: | 30 | 31 | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 0111000011000011 |
| 140h: | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 30 | 30 | 0011000001101100 |
| 150h: | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 0011011000001101 |
| 160h: | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 30 | 1000011000110110 |
| 170h: | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 0000011000011000 |
| 180h: | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 0011011000001101 |
| 190h: | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 1000001100011011 |
| 1A0h: | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 31 | 31 | 30 | 0000110011000110 |
| 1B0h: | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 0000110110000000 |
| 1C0h: | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 1100000110110000 |
| 1D0h: | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 0000110001100000 |
| 1E0h: | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 1100001100000110 |
| 1F0h: | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 1100000110110000 |
| 200h: | 30 | 31 | 31 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 0110011000110000 |
| 210h: | 31 | 31 | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 1111000011111111 |
| 220h: | 31 | 30 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 30 | 30 | 31 | 31 | 31 | 31 | 31 | 1001111110011111 |
| 230h: | 31 | 31 | 31 | 30 | 30 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 30 | 30 | 31 | 31 | 1110001111110011 |
| 240h: | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 0000011000011000 |
| 250h: | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 31 | 31 | 31 | 0110000011011111 |
| 260h: | 31 | 31 | 31 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 1110011000001101 |
| 270h: | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 1000001100001100 |
| 280h: | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 0001100000110000 |
| 290h: | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 0001101100000110 |
| 2A0h: | 30 | 30 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 0000001101111111 |
| 2B0h: | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 30 | 30 | 31 | 31 | 30 | 30 | 1100001100001100 |

1177=107*11，画个图

也是醉了。

## Help me

IDA 反编译 main 函数，可以看到如下代码：

```
34  v0 = 305419896;
35  dword_40CF78 = -2;
36  if ( v3 )
37  {
38    do
39    {
40      if ( dword_40CF70 == 1 )
41      {
42        v5 = dword_40CF74;
43        if ( dword_40CF74 == -1 )
44        {
45          if ( dword_40CF78 != -2 )
46            goto LABEL_12;
47          String[v4] ^= 9u;
48          v6 = String[v4];
49          v9 = 0;
50          OutputString = 0;
51          sub_401200("%2X", v6);
52          OutputDebugStringA(&OutputString);
53          ++v4;
54          continue;
55        }
56      }
57      else
58      {
```

对 String 字符串进行异或 9 的操作，并将其以 16 进制输出，结果即为 flag。

String: rev3rs3_ana1ys1s

flag: 7b6c7f3a7b7a3a5668676838707a387a

## 捉迷藏

用 IDA 分析：

IDA View-A | Pseudocode-A | Hex View-1 | Structures

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax@3
  char v5; // [sp+8h] [bp-D8h]@1
  char v6; // [sp+9h] [bp-D7h]@1
  char Str; // [sp+70h] [bp-70h]@1
  char Dst; // [sp+71h] [bp-6Fh]@1

  Str = 0;
  memset(&Dst, 0, 0x63u);
  v5 = 0;
  memset(&v6, 0, 0x63u);
  sub_401F20(std::cout, "input the correct name:\n");
  sub_402160(std::cin, &Str);
  if ( strlen(&Str) == 7 )
  {
    sub_401F20(std::cout, "input the correct passwd:\n");
    sub_402160(std::cin, &v5);
    if ( strlen(&v5) == 14 )
    {
      v3 = strcmp(&Str, "FindKey");
      if ( v3 )
        v3 = -(v3 < 0) | 1;
      if ( !v3 )
      {
        if ( sub_401C00((int)&v5, 14) )
          sub_401D00(&Str, &v5);
        printf_s("well_done!");
      }
    }
  }
  return 0;
}
```

00000FB2 main:23

可以看到输入的用户名为：FindKey

密码是 14 位，关键函数为 sub_401C00，进入看看

```
IDA View-A   Pseudocode-A   Hex View-1   Structures   Enums

int __fastcall sub_401C00(int a1, int a2)
{
  int v2; // esi@1
  char *v3; // edi@1
  const char *v4; // eax@1
  int v5; // eax@3
  int v6; // ebx@5
  char *Src; // [sp+4h] [bp-F0h]@1
  unsigned int v9; // [sp+18h] [bp-DCh]@1
  char v10; // [sp+1Ch] [bp-D8h]@1
  char v11; // [sp+1Dh] [bp-D7h]@1
  char v12[16]; // [sp+84h] [bp-70h]@1
  int v13; // [sp+94h] [bp-60h]@1
  char Dst; // [sp+98h] [bp-5Ch]@1

  v13 = 0x49585A;
  v2 = a2;
  v3 = (char *)a1;
  _mm_storeu_si128((__m128i *)v12, _mm_loadu_si128((const __m128i *)&xmmword_403338));
  memset(&Dst, 0, 0x50u);
  v10 = 0;
  memset(&v11, 0, 0x63u);
  sub_4012D0((int)&Src, v3, v2);
  v4 = (const char *)&Src;
  if ( v9 >= 0x10 )
    v4 = Src;
  strcpy_s(&v10, 0x64u, v4);
  strcat_s(v12, 0x64u, "=");
  v5 = strcmp(&v10, v12);
  if ( v5 )
    v5 = -(v5 < 0) | 1;
  v6 = v5 == 0;
  if ( v9 >= 0x10 )
    operator delete(Src);

00001000 sub_401C00:7
```

其中 V12 即位 xmmword_403338:

```
.rdata:00403337                 align 4
.rdata:00403338 xmmword_403338  xmmword 305648637439325179563362 5A353254h
.rdata:00403338                                         ; DATA XREF: sub_401C00+1
.rdata:00403348 dword_403348    dd 49585Ah              ; DATA XREF: sub_401C00+1
.rdata:0040334C ; char Src[2]
```

函数 sub_4012D0 作用是求 v3 的 base64 编码，将其与 v12 比较，所以对 v12 进行 base64 解码即可得到密码：

```
import base64
v12 = "49585A305648637439325179563362 5A353254"
v12 = v12.decode("hex")
v12 = v12[::-1]
print v12
#v12=T25Zb3VyQ29tcHV0ZXI
password = base64.b64decode(v12+"=")
print password
#password=OnYourComputer
```

然后查看 sub_401D00 函数：

IDA View-A | Pseudocode-A | Hex View-1 | Structure

```
int __fastcall sub_401D00(void *Str, void *a2)
{
  void *v2; // edi@1
  void *v3; // esi@1
  FILE *File; // [sp+8h] [bp-10Ch]@1
  CHAR Buffer; // [sp+Ch] [bp-108h]@1
  char Dst; // [sp+Dh] [bp-107h]@1

  Buffer = 0;
  v2 = a2;
  v3 = Str;
  memset(&Dst, 0, 0x103u);
  GetTempPathA(0x104u, &Buffer);
  strcat_s(&Buffer, 0x104u, "flag.jpg");
  fopen_s(&File, &Buffer, "w+");
  fwrite(v3, strlen((const char *)v3), 1u, File);
  fwrite(v2, strlen((const char *)v2), 1u, File);
  fwrite("ArvinShow", 9u, 1u, File);
  return fclose(File);
}
```

得到 flag:FindKeyOnYourComputerArvinShow

# 移动迷宫

这道题其实就是对输入进行转换，变成走迷宫的路径，能走出迷宫的输入就是 flag:

```
E017                    db  31h ; 1
E018 ; char byte_40E018[]
E018 byte_40E018        db 2Ah                  ; DATA XREF: _main+150↑r
E019                    db '**********####******#**#*****##*##********#*********#*#####***###'
E019                    db '***#*********#*********#********##',0
E07D                    align 10h
E080 off_40E080         dd offset unk_40FD20    ; DATA XREF: sub_4017DA↑o
E080                                            ; ___initstdio+52↑o ...
E084                    align 8
E088                    dd offset unk_40FD20
```

迷宫地图如上，别忘记了还有前面一个\x2A，总共 100 个字符，地图为 10*10:

```
**********
*####*****
*#**#*****
##*##*****
***#******
***#*#####
***###***#
*********#
*********#
********##
```

```
  sub_401000((int)&v6[6 * i], (int)(&v8 + 6 *
while ( v16 < 24 )
{
  switch ( *(&v8 + v16) )
  {
    case 1:
      --v7;
      break;
    case 2:
      ++v7;
      break;
    case 3:
      --v17;
      break;
    case 4:
      ++v17;
      break;
    default:
      break;
  }
  if ( *(&byte_40E018[10 * v7] + v17) != 35 )
```

所以 1 代表向上走，2 代表向下走，3 代表向左走，4 代表向右走，所以可以得出路径：411444223222441444422223

sub_401000 函数中对输入作如下操作，将路径分为 6 位一组，byte_40E000 数组分为 4 位一组，byte_40E000 为 0A1Ba2b34C5Dc6d78E9Fe0f1，如路径第一个为 411444，byte_40E000 第一组为 0A1B，所以输入的第一位就是 B，路径第二个为 1，byte_40E000 第二组为 a2b3，所以输入的第二位就是 a，以此类推

IDA View-A | Pseudocode-A | Hex View-1 | Structures

```
int __cdecl sub_401000(int a1, int a2)
{
  int result; // eax@7
  signed int v3; // [sp+0h] [bp-8h]@1
  signed int i; // [sp+4h] [bp-4h]@1

  v3 = 0;
  for ( i = 0; i < 6; ++i )
  {
    while ( v3 < 4 )
    {
      if ( *(_BYTE *)(i + a1) == *(&byte_40E000[4 * i] + v3) )
      {
        *(_BYTE *)(i + a2) = v3 + 1;
        break;
      }
      ++v3;
    }
    v3 = 0;
    result = i + 1;
  }
  return result;
}
```

脚本如下：

```
a="***********##########*#**#*****##*##********#*********#*#######***###***#****
*****#*********#********##"
print len(a)
for i in range(10):
    print a[i*10:i*10+10]
A="RUURRRDDLDDDRRURRRRDDDDL"
```

```
print len(A)
#1:U,2:D,3:L,4:R
flag = ""
for i in A:
    if i == "R":
        flag += '4'
    elif i == "L":
        flag += '3'
    elif i == "U":
        flag += '1'
    else:
        flag += '2'
print flag
aa=flag
bb="0A1Ba2b34C5Dc6d78E9Fe0f1"
flag=""
for i in range(len(aa)):
    num = i%6
    if aa[i]=='1':
        flag += bb[num*4]
    elif aa[i]=='2':
        flag += bb[num*4+1]
    elif aa[i]=='3':
        flag += bb[num*4+2]
    else:
        flag += bb[num*4+3]
print flag
#flag_Xd{hSh_ctf:Ba47F1A256E0B347F1B2C6Ef}
```

flag: Ba47F1A256E0B347F1B2C6Ef

## Do something

关键函数为 sub_401380，这题应该算数学题，输入为 16 字节的字符串，将每个字符减去 96 得到 16 个整数，对这 16 个整数的约束条件在函数 sub_401000 中：

```
IDA View-A    Pseudocode-A    Strings window    Hex Vie
int __cdecl sub_401000(char *Src)
{
  char Dst; // [sp+0h] [bp-14h]@1
  int v3; // [sp+1h] [bp-13h]@1
  int v4; // [sp+5h] [bp-Fh]@1
  int v5; // [sp+9h] [bp-Bh]@1
  int v6; // [sp+Dh] [bp-7h]@1
  __int16 v7; // [sp+11h] [bp-3h]@1
  char v8; // [sp+13h] [bp-1h]@1

  Dst = byte_415282;
  v3 = 0;
  v4 = 0;
  v5 = 0;
  v6 = 0;
  v7 = 0;
  v8 = 0;
  strcpy_s(&Dst, 0x11u, Src);
  sub_401320(Dst, $BYTE3(v4));
  sub_401320(Dst, (char)v5);
  sub_401320((char)v3, $BYTE1(v5));
  sub_401320($BYTE1(v3), $BYTE3(v3));
  sub_401320($BYTE2(v3), (char)v4);
  sub_401320($BYTE2(v5), 5);
  sub_401320($BYTE2(v4), 3 * $BYTE2(v5));
  sub_401350($BYTE3(v5), 5 * $BYTE1(v6));
  sub_401320((char)v6, 2 * $BYTE3(v5));
  sub_401350($BYTE2(v3), 3 * $BYTE3(v5));
  sub_401350(Dst, $BYTE2(v3));
  sub_401350(21, Dst);
  sub_401320(Dst, $BYTE1(v4) + $BYTE3(v5));
  sub_401320($BYTE1(v4), 2 * $BYTE2(v6));
  sub_401350($BYTE1(v3), 4 * $BYTE1(v6));
  sub_401350($BYTE1(v4), $BYTE1(v3));

00000400 sub_401000:1
```

sub_401320(a1,a2)绕过的条件是 a1 等于 a2，

sub_401350(a1,a2)绕过的条件是 a1>a2。

所以可以得到对这 16 个整数的约束条件，算出满足条件的整数如下：

20,8,9,d,9,d,14,15,20,20,8,5,6,12,1,7

所以得到 flag:

```
d = 19
data = [20,8,9,d,9,d,14,15,20,20,8,5,6,12,1,7]
print len(data)
flag = ""
for i in data:
    flag += chr(96+i)
print flag
```

flag: thisisnottheflag

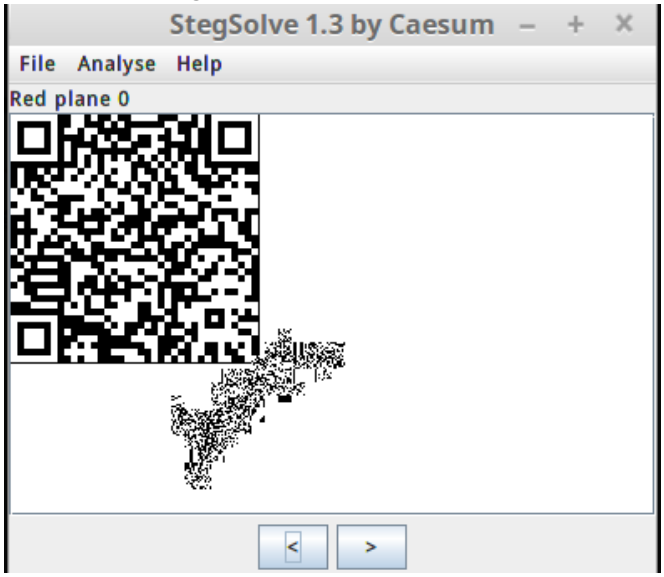
```
please input user name:thisisnottheflag
you can get flag at reverse.xdsec.cc/galfehttonsisiht.php
```

访问该网页得到 flag:
flag_Xd{hSh_ctf:KaliI5600d}

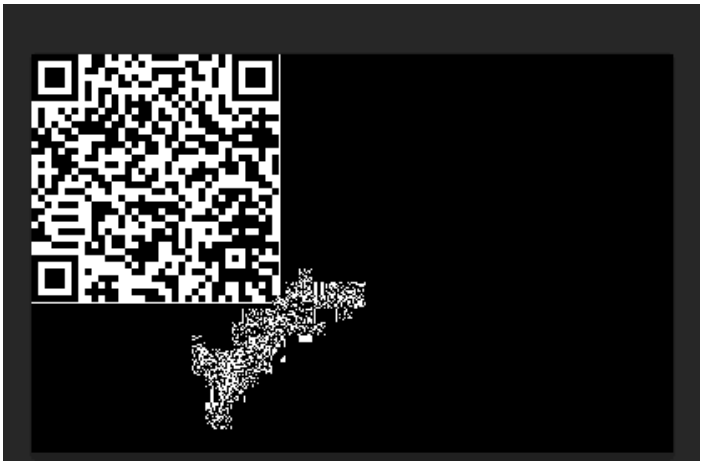
# Forensics

## 蒲公英的约定

直接用 Stegsolve 可以查看隐藏了二维码




然后 ps 转色一下



扫一下就可以到一传字符串,MZWGCZ27LBSHW2CTNBPWG5DGHJKTE4ZQL5RW63ZRPU======
然后 base32 解码后得到 flag

## 什么鬼

binwalk 可以识别出里面藏了 zip
提取出来后需要密码，密码提示是 4 位，直接爆破

解开后是一个二维码

补全定位符，扫码得 flag

## 客官，听点小曲儿？

密码 cheers，日了狗了，huashan 当密码居然也能解开

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Sat, 10 Sep 2016 13:39:16 GMT
Content-Type: text/html
Content-Length: 121
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.19
key: cheers
Vary: Accept-Encoding

<html>
<header>
<title>Music</title></header>
<body>
<a href=song.mp3><center>Listen</center></a>
</body>
</html>
```

MP3Stego.exe

```
C:\Windows\system32\cmd.exe

E:\CTF\2016华山杯\misc\MP3Stego_1_1_18\MP3Stego>Decode.exe -X -P cheers song.mp3

MP3StegoEncoder 1.1.17
See README file for copyright info
Input file = 'song.mp3'  output file = 'song.mp3.pcm'
Will attempt to extract hidden information. Output: song.mp3.txt
the bit stream file song.mp3 is a BINARY file
HDR: s=FFF, id=1, l=3, ep=off, br=9, sf=0, pd=1, pr=0, m=0, js=0, c=0, o=0, e=0
alg.=MPEG-1, layer=III, tot bitrate=128, sfrq=44.1
mode=stereo, sblim=32, jsbd=32, ch=2
[Frame 9747]Avg slots/frame = 417.917; b/smp = 2.90; br = 127.987 kbps
Decoding of "song.mp3" is finished
The decoded PCM output file name is "song.mp3.pcm"
```

得到 fdc3_#l{tsf#ahfte}gS:en_hmgcX_poe

栅栏密码，最后

```
fdc3_#
l{tsf#
ahfte}
gS:en
_hmgc
X_poe
```

# Web

## 打不过

这个题一开始以为是万能密码登录，结果不是。返回包里 str 的值解 base64 然后再解 md5 得到 flag

## 系统管理

右键查看源码得到一段注释
<!-- $test=$_POST['username']; $test=md5($test); if($test=='0') -->
意思就是找个 md5 值弱等于 0 的，记得去年西电就出过这道题。网上一搜选个值 aabg7XSs。
提交之后得到提示 user.php，访问 user.php 的得到源码
$unserialize_str = $_POST['password']; $data_unserialize = unserialize($unserialize_str); if($data_unserialize['user'] == '???' && $data_unserialize['pass']=='???') { print_r($flag); }
一个简单的反序列化。给出 payload
username=aabg7XSs&password=a:2:{s:4:"user";s:3:"???";s:4:"pass";s:3:"???";}

## 简单的 js

打开网页，打开 firebug 发现右边可以看到执行完的 a 的值，然后把这个值输入到 input 框中，就可以得到 flag



# 弹弹弹

这个题目测就过滤了 script 标签 <img src=1 onerror=alert(1)>成功弹窗得到 flag

# 233

右键看源码，是一大段 js 变形的代码。扔到 console 里面，去掉最后的括号，用 toString 得到这段代码的字符串

function anonymous() {

alert("十攡數畣整爟煥敵瑳∨瞃祳ぴ⒈点ᶊ┤ >")

}

alert 里面的字符串转十六进制得到 253c652065787563657472207165657574732228406579733074336722742529 3e

再转回字符串得到

%<e exucetr qeeuts"(@eys0t3g"t%)>

每两位字符交换顺序得到最后的 flag

# 无间道

文件上传，题目代码似乎有误，无论传什么都提示只能上传图片，用“PHP 很烦人”题目中的文件读取可以看到源码

```
    if($ex===php){
        echo "flag_Xd{hSh_ctf:asu@3sud9:!}";
        unlink("upload/" . $_FILES["file"]["name"]);
    }
}
```

# php 很烦人

右键源码：

```
if(isset($user)&&(file_get_contents($user,'r')=="the user is admin")){
    echo "hello admin!<br>";
    include($file); //class.php
}else{
    echo "you are not admin ! ";
}
```

User=php://input，POST 数据 the user is admin 可过第一步

$file 变量可以使用 php://filter 读取任意文件源码

读取 class.php 和 index.php

$file = php://filter/read=convert.base64-encode/resource=class.php

```
<?php

class Read{//f1a9.php
    public $file;
    public function __toString(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
        }
        return "__toString was called!";
    }
}
?>
```

Index.php 中没有在注释中的代码

```
if(preg_match("/f1a9/",$file)){
    exit();
}else{
    include($file); //class.php
    $pass = unserialize($pass);
    echo $pass;
}
```

将$pass 变量反序列化并输出，并过滤了 f1a9，在$pass 中构造$file=f1a9.php 的 Read 类实例即可读取 flag。

## More try

用户登录界面，登录 POST 的表单中的 role 变量为两次 base64 后的字符串，无过滤可注入。写个 base64 中专脚本，sqlmap 解决

```
Database: websec
Table: the_key
[1 entry]
+-------------------------------+
| key                           |
+-------------------------------+
| flag_Xd{hSh_ctf:sql_succeed!} |
+-------------------------------+
```

## 三秒钟记忆

http://huashan.xdsec.cn/pic/login(main/logout/pic)能得到这些源码，其中最重要的就是 login 的源码。源码很长就不贴了，大意是有 3 个功能，注册、登录、重置密码。其中在注册和登录处都有 mysql_real_escape_string 函数做过滤。但是在重置密码处，这一句代码会造成二次注入

mysql_query(sprintf("update users set password='%s', resetinfo='%s' where username='%s'", $passnew,$ip,$res->username));

username 在取出来又拼接进 update 语句时没有做过滤导致 sql 注入。而能够判断是否能够注入的办法只能是重置密码后再登录看 update 语句有没有重置成功，因此只能是盲注。这里吐槽一下重置密码的响应时间，基本上 1 分钟左右才能跑一条 payload。正在被这无语的响应速度折磨时，有队友谷歌发现这居然是个原题：http://tasteless.eu/post/2014/04/plaidctf-2014-whatscat-writeup/。直接把里面的脚本抓来稍微改改就开跑。

## 疯狂的 js

这题懒得看代码了，只要知道结果 filter 函数的返 回结果是 XDCTFHS 就行了。

在浏览器的 console 里输入以下代码，选一组结果提交得 flag

```
var FLAG="XDCTFHS";

function var_dump(o){
	var str = [];
	var pre = '';
	typeof o == 'function' && '';
	typeof o == 'string' ? console.log(o) : console.log(each.call(o));
	function each(o){
		for (var p in this) {
			typeof this[p] == 'object' && (pre += '[' + p + '].', each.call(this[p])) || str.push(pre + '[' + p + ']->[' + this[p] + '] /r/n')
		}
		return str.join('');
	}
}

function cal(z) {
for(var xxx = 5; xxx < 50; xxx+=5) {
	var yyy = (function stopb(b) { return b < 0 ? function(n){return Infinity;} : function stop(n) {
	return (function(s){return s && (s<b?s:Infinity)})(cal.sbox[n]) || (function(q){return q == Infinity ? q : (cal.sbox[n] = q)})(1+Math.min.apply(null, [n/2, 3*n+1, n/(function twos(n) {
		return n%2 || 2*twos(n/2);
	})(n)].map(function xia(x,i,a) {
		return i >= a.length ? undefined : x%1 || !x || x == n ? (function(){a.splice(i,1); return xia(a[i],i,a);})() : x;
	}).filter(function(x){return x;}).map(stopb(b-1))));

}; })(xxx)(z);
	if(yyy != Infinity) return yyy;
}
}
cal.sbox = {2:1, 1:1, 4:1};

function filter() {
	var args = [].slice.apply(arguments).sort().filter(function(x,i,a){return a.indexOf(x) == i;});
	if(args.length != 5) return "数够参数了吗？ ";
	var flag = false; args.map(function(x){flag |= x >= 999;});
	if(flag) return "有点大了哦";

	var m = args.map(cal);
   //var_dump(m);
	if(m.filter(function(x,i){return m[1]+4*i==x;}).length < 2) return "no";
	if(m.filter(function(x,i){return m[1]+3*i==x;}).length < 1) return "no";
```

```
        if(m.filter(function(x,i){return x == args[i];}).length < 2) return "nono";
        if(m.filter(function(x,i){return x > m[i-1];}).length > 2) return "bala";
        if(m.filter(function(x,i){return x < m[i-1];}).length > 1) return "balana~";

        return FLAG;
}

for(var a=1;a<100;a++){
    for(var b=1;b<100;b++){
        for(var c=1;c<100;c++){
            for(var d=1;d<100;d++){
                for(var e=1;e<100;e++){
                    if (FLAG == filter(a,b,c,d,e)){
                        console.log(a,b,c,d,e);
                        break;
                    }
                }
            }
        }
    }
}
```