CTF比赛中关于zip的总结

转载

FLy_鹏程万里 于 2018-04-16 15:37:20 发布 10019 收藏 61
分类专栏： 【CTF】# CTF基础

【CTF】同时被 2 个专栏收录
30 篇文章 26 订阅
订阅专栏


CTF基础
4 篇文章 4 订阅
订阅专栏

前言

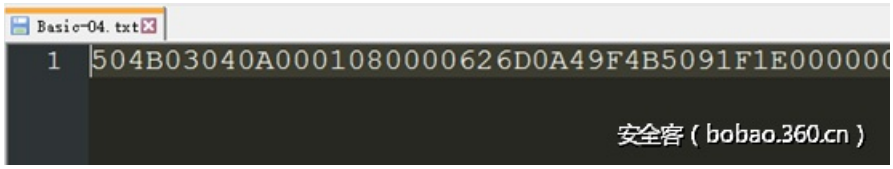
在CTF比赛的MISC和CRYPTO中，经常要和zip压缩包打交道，这里做一个zip方面的总结。

本文中用到的所有文件和工具都可在这个网盘中找到http://pan.baidu.com/s/1bWQxyA

目录

0x01. 通过进制转换隐藏信息

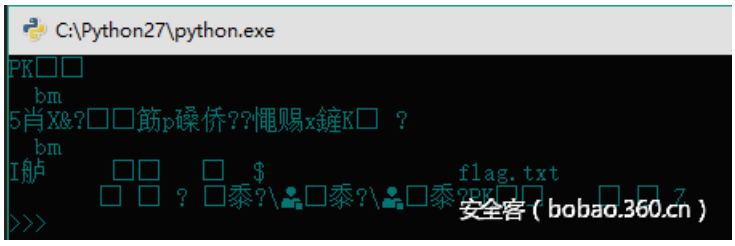
这种方法比较简单，直接拿一道题讲解（题目来自ISCC 2017 Basic-04）。题目给了一个txt文档如下图



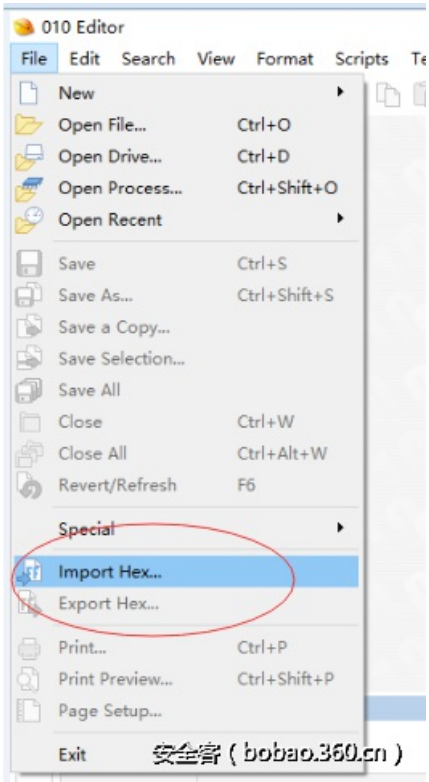
经过观察，所有数据都在16进制能表示的范围之内，因此先尝试使用十六进制编码解密，python脚本如下：

```
#coding:utf-8
with open('Basic-04.txt') as f:
    cipher = f.read()#读取 txt 内容
    plain = cipher.decode('hex')#16 进制编码解密
    print plain
```

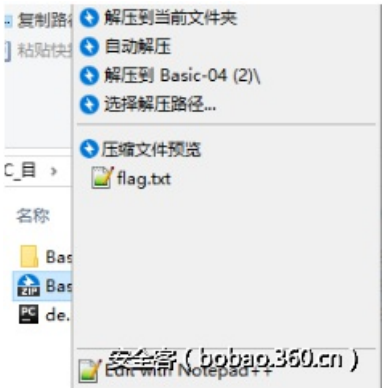

运行结果如下，虽然存在大量乱码，但还是能看到flag.txt，因此猜测txt中的这段字符是zip包的16进制表示（同时开头的PK也暗示这是一个zip包，PK是zip格式发明者Phil Katz的名称缩写，zip的前两个字母就用了PK）



导入到16进制编辑器中，这里用010editor做演示



导入后选择 Save As（快捷键 ctrl + shift + s），给新文件命名时加上后缀.zip，保存后发现zip文件是正常的，因此证明思路正确，此题的后续过程请继续阅读这篇文章

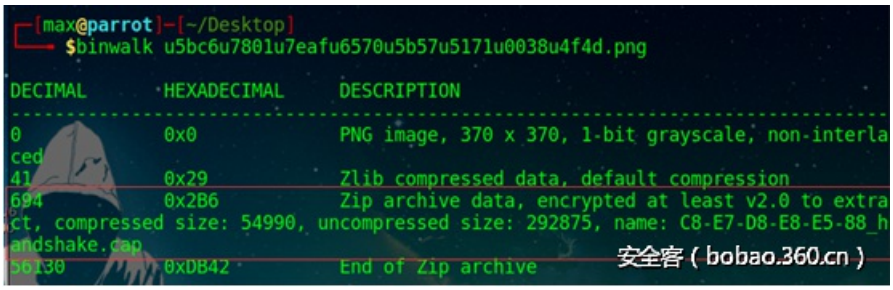


另：除了16进制的编码转换，有时还会遇到2进制编码的转换，思路相同，不再复述

0x02. 在图片中隐藏压缩包（图种）

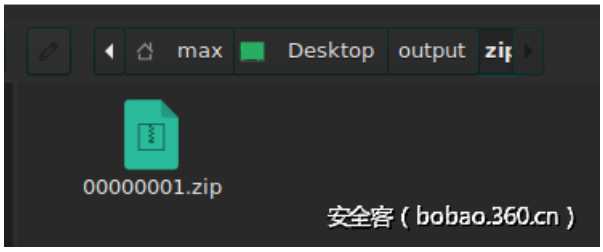
这种方法大概是zip中最常见的，多用于在一张图片中隐藏一个压缩包，这种方法的原理是：以jpg格式的图片为例，一个完整的 JPG 文件由 FF D8 开头，FF D9结尾，图片浏览器会忽略 FF D9 以后的内容，因此可以在 JPG 文件中加入其他文件。

也以一道题为例为例（ISCC 2017 Basic-07），对于这种隐写最简单的方法是使用Kali下的binwalk进行检测，binwalk 图片名 如下，检测出图片中存在压缩包

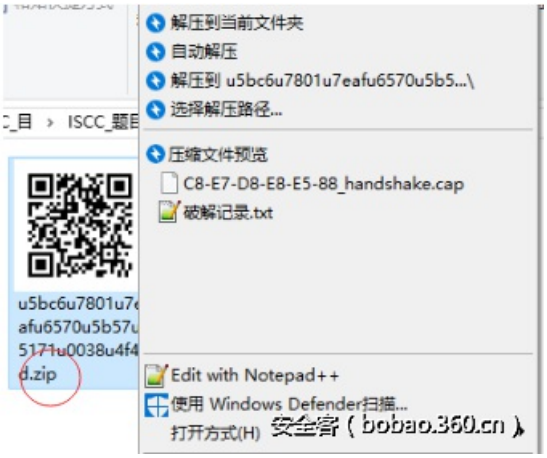


分离这个压缩包也有至少两种方法：

1. 利用Linux下的foremost工具， foremost 图片名 如下，foremost默认的输出文件夹为output，在这个文件夹中可以找到分离出的zip（推荐使用这种方法，因为foremost还能分离出其他隐藏的文件）



2. 更简单粗暴的方法是直接把图片的后缀改为.zip，然后解压即可（这种方法虽然简单快速，但如果隐写了多个文件时可能会失败）



另：本题后续步骤为构造字典，爆破握手包

0x03. 伪加密

Zip伪加密与zip的文件格式有关（zip的格式详解请翻到本文的最后0x07部分），zip中有一位是标记文件是否加密的，如果更改一个未加密zip包的加密标记位，那么在打开压缩包时就会提示该文件是加密的。

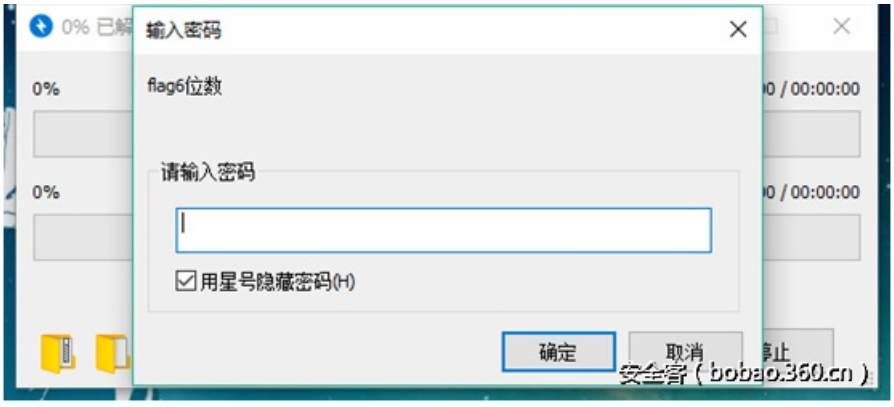
对于伪加密有以下几种方法：

1. 在Mac OS及部分Linux（如Kali）系统中，可以直接打开伪加密的zip压缩包

2. 使用检测伪加密的ZipCenOp.jar，解密后如果能成功打开zip包，则是伪加密，否则说明思路错误

3. 使用16进制编辑器改回加密标记位

以HBCTF的一道题讲解这几种方法：



如上，尝试解压压缩包时提示有密码，根据题干：比爆破更好的方法推测为伪加密，用三种方法来解此题：

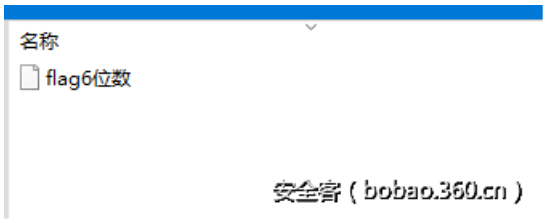
1. 用除windows外的系统直接打开压缩包

在Mac OS和部分Linux系统（如Kali）中，右键解压可直接打开伪加密的zip压缩包，笔者暂未明确何种Linux能打开伪加密压缩包，如有传授，不胜感激！

2. 使用ZipCenOp.jar（需java环境） 使用方法

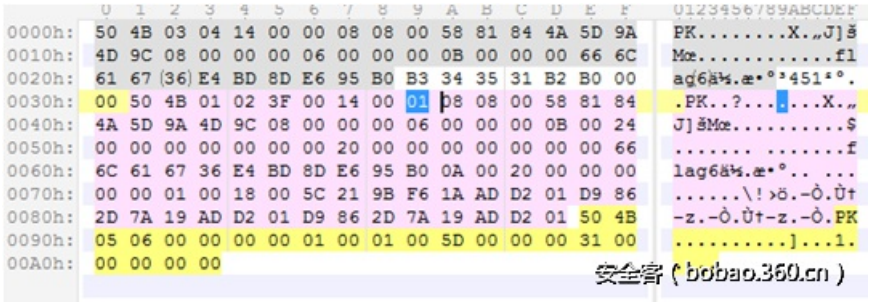
```
java -jar ZipCenOp.jar r xxx.zip
```

经ZipCenOp.jar解密后的压缩包可直接打开



推荐使用这种方法，最便捷

3. 用16进制编辑器修改加密标记位



如上图，修改加密标记位为00，保存，即可打开压缩包（关于zip文件的结构，请翻到本文最末0x07部分）

0x04. 爆破/字典/掩码攻击

把这三种归位一类是因为这三种方法在本质上都是逐个尝试，只不过待选密码的集合不同

1. 爆破：顾名思义，逐个尝试选定集合中可以组成的所有密码，知道遇到正确密码

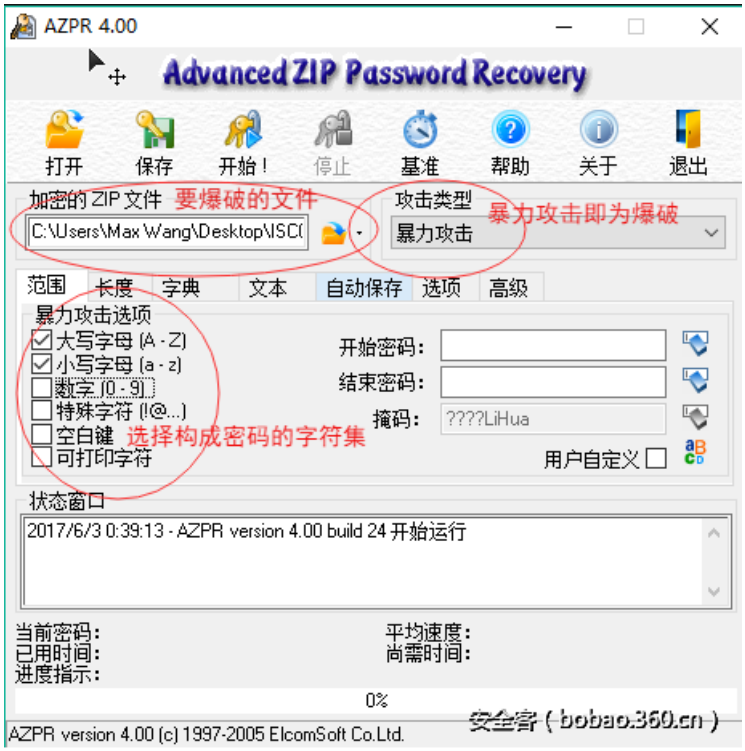
2. 字典：字典攻击的效率比爆破稍高，因为字典中存储了常用的密码，因此就避免了爆破时把时间浪费在脸滚键盘类的密码上

3. 掩码攻击：如果已知密码的某几位，如已知6位密码的第3位是a，那么可以构造 ??a??? 进行掩码攻击，掩码攻击的原理相当于构造了第3位为a的字典，因此掩码攻击的效率也比爆破高出不少

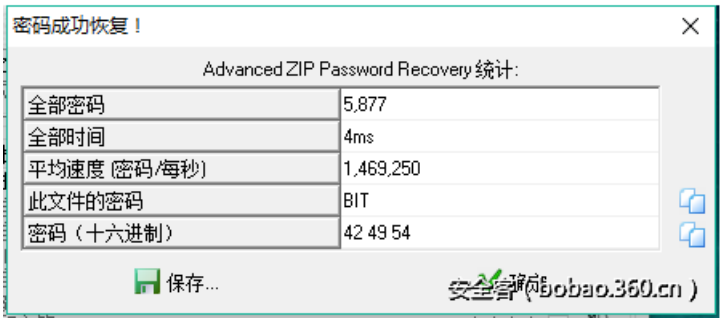
对这一类的zip问题，推荐windows下的神器AZPR

举例如下：

1. 对爆破，以ISCC 2017 Basic-08为例，选定暴力攻击、字符集和长度后进行爆破

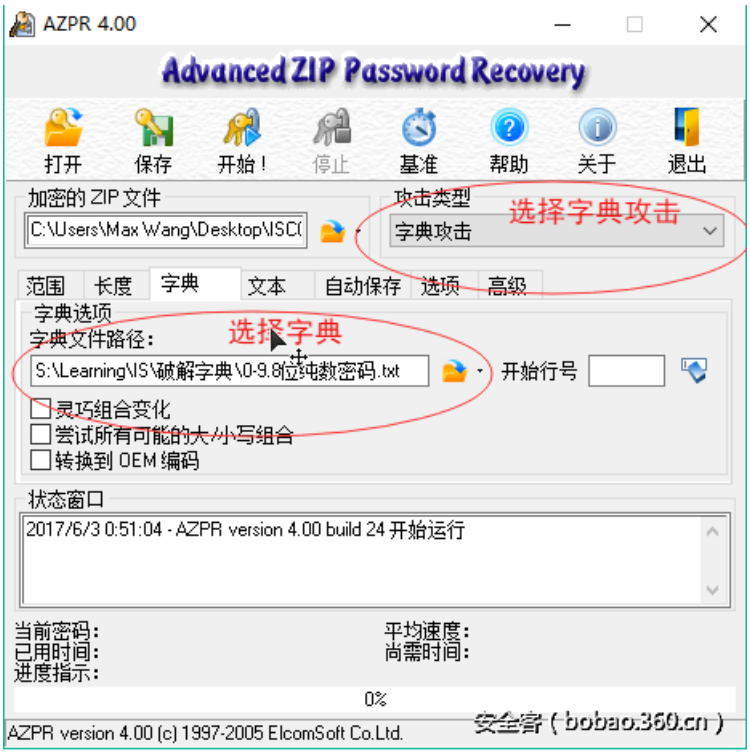


点击开始，进行爆破，如下图，在4ms内就找到了密码为BIT

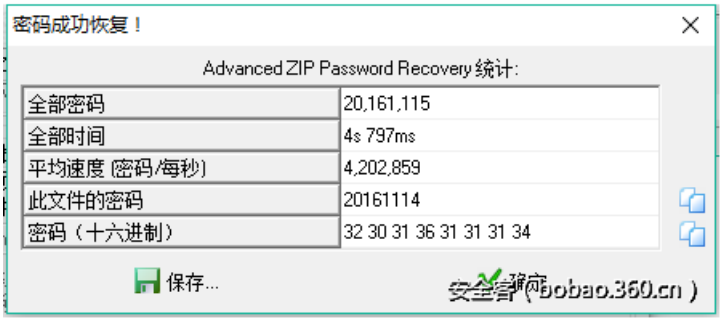


另：此题后续为简单的base64解密；爆破在密码长度小于6位时较快，因此如果在7位之内没有爆破出结果时，基本就可以考虑换个方法了；此题的正规解法是培根密码的转换

2. 字典，还以之前的ISCC 2017 Basic-07举例，从图片中分离出一个加密的zip压缩包，爆破无果后考虑字典攻击（可从网上下载字典，但大多数题目需要自己构造字典，文末的网盘连接里提供了常见的字典）



字典攻击的结果如下图，在字典选择合适的情况下，用很短的时间就能找到密码



继续以此题为例，解压后的压缩包有一个txt文档和一个握手包，txt内容如下：

因此可知握手包的密码为ISCC****的形式（*代表大写字母或数字），自己写程序构造字典

```python
#coding:utf-8
import string

pw = 'ISCC'
s = string.digits + string.uppercase#s 为后四位密码的可选字符集

f = open('dic.txt', 'w')
for i in s:
    for j in s:
        for p in s:
            for q in s:
                f.write(pw + i + j + p + q + '\n')#注意字典中的每一条记录都以
\n 结尾

f.close()
```

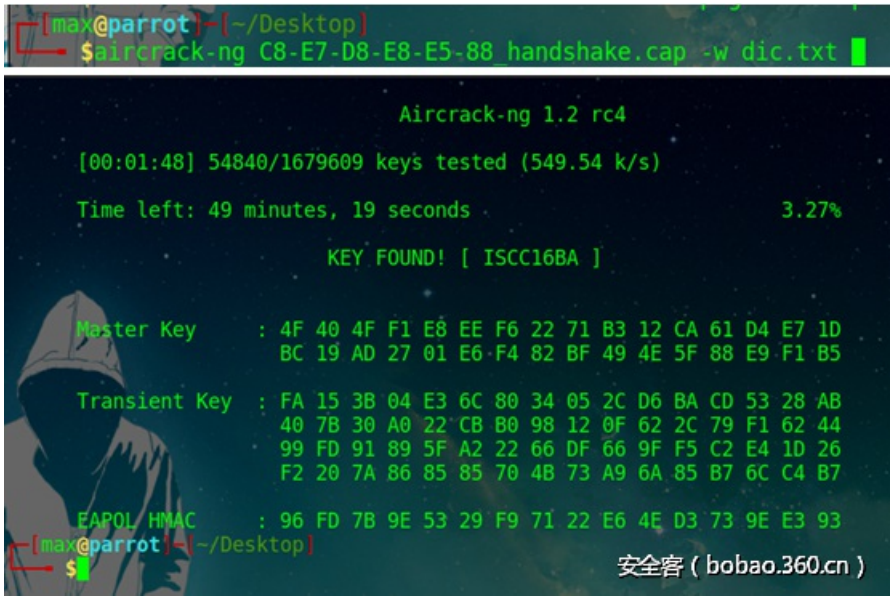
安全客 (bobao.360.cn)

运行此程序得到字典如下：

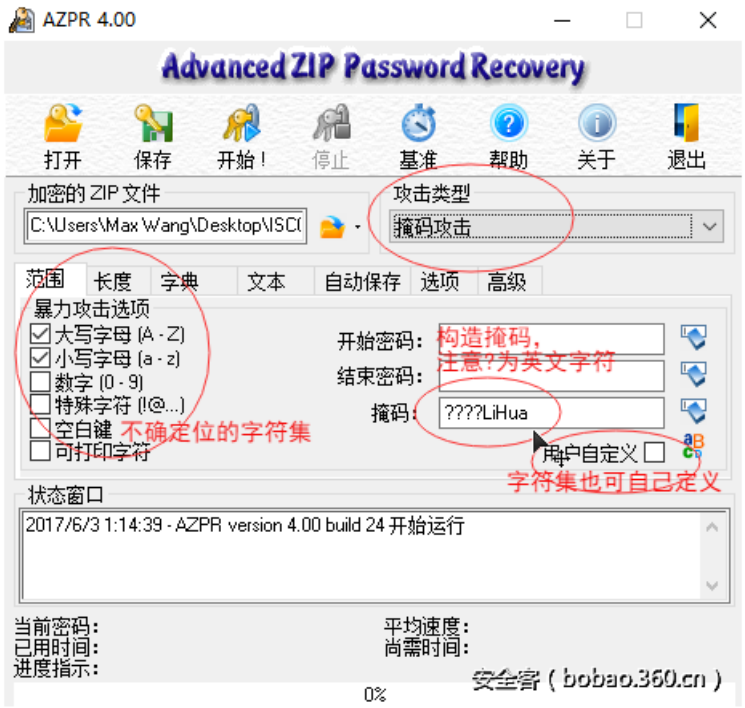
```
ISCC0000
ISCC0001
ISCC0002
ISCC0003
ISCC0004
ISCC0005
ISCC0006
ISCC0007
ISCC0008
ISCC0009
ISCC000A
ISCC000B
ISCC000C
ISCC000D
ISCC000E
ISCC000F
ISCC000G
ISCC000H
ISCC000I
```

安全客 (bobao.360.cn)

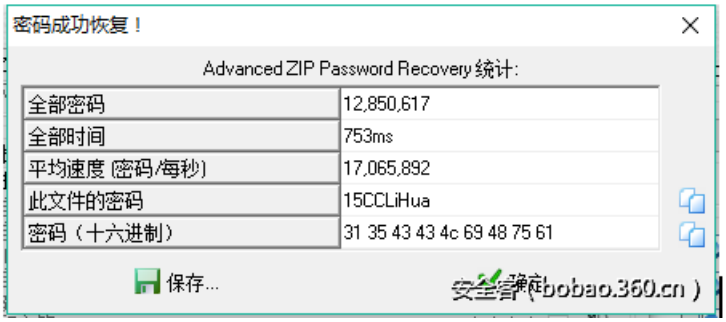
之后用aircrack-ng来选中字典跑除握手包的密码如下图，不再详述



3. 掩码攻击，以ISCC 2017 Misc-06为例，题目给了一个jpg图片，用0x02中的方法分离出加密的压缩包，根据题目提示：注意署名， 构造????LiHua的掩码（?可在自己定义的字符集中任意选择）进行掩码攻击，如下图：



攻击结果如下，只耗费了很少的时间就找到了密码



0x05. 明文攻击

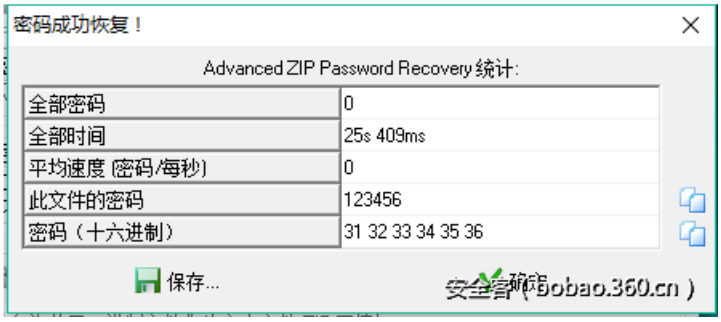
明文攻击是一种较为高效的攻击手段，大致原理是当你不知道一个zip的密码，但是你有zip中的一个已知文件（文件大小要大于12Byte）时，因为同一个zip压缩包里的所有文件都是使用同一个加密密钥来加密的，所以可以用已知文件来找加密密钥，利用密钥来解锁其他加密文件，更详细的原理请读者自行谷歌

举个例子，已知 明文攻击.zip 中存在的文件 明文.txt，

因此将 明文.txt 压缩，这里需要判断明文压缩后的CRC32是否与加密文件中的一致，若不一致可以换一个压缩工具。

攻击过程如下：

点击开始，很快就恢复了密码



另：当明文的大小比较小时，攻击速度会比较慢；即使有时没有恢复密码，也可以使用明文攻击，最后点保存还是能得到压缩包里内容的。

0x06. CRC32碰撞

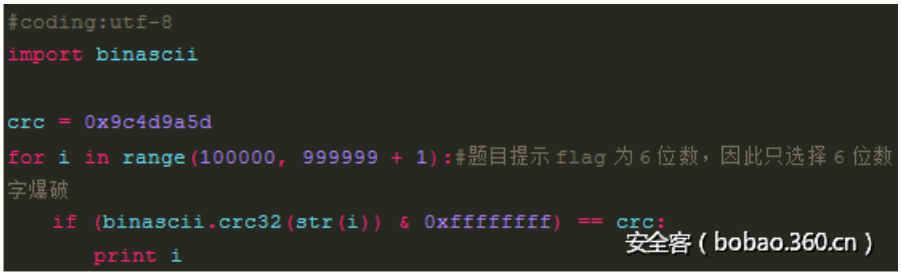
CRC32:CRC本身是"冗余校验码"的意思，CRC32则表示会产生一个32bit（8位十六进制数）的校验值。

在产生CRC32时，源数据块的每一位都参与了运算，因此即使数据块中只有一位发生改变也会得到不同的CRC32值，利用这个原理我们可以直接爆破出加密文件的内容

还是以之前HBCTF伪加密那道题为例，另一种解法就是CRC32碰撞，打开压缩包，可以看出压缩文件 flag6位数

的CRC32值为0x9c4d9a5d

因此写出碰撞的脚本如下：

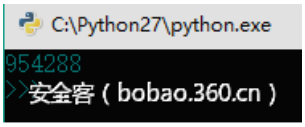


要特别注意

```
if (binascii.crc32(str(i)) & 0xffffffff) == crc:
```

在 Python 2.x 的版本中，binascii.crc32 所计算出来的 CRC 值域为[-2^31, 2^31-1] 之间的有符号整数，为了要与一般CRC 结果作比对，需要将其转为无符号整数，所以加上& 0xffffffff来进行转换。如果是 Python 3.x 的版本，其计算结果为 [0, 2^32-1] 间的无符号整数，因此不需额外加上& 0xffffffff 。

脚本的运行结果如下，即为压缩文件的内容：



再举另一个bugku中的例子，下载下来的文件是68个压缩包，并且根据binwalk的检查结果，每个压缩包里都有一个大小为4个字节，名为out.txt的压缩文件

```
C:\Users\Max Wang\Desktop\123
> binwalk out0.zip

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Zip archive data, encrypted at least v2.0 to extract, compressed size: 18, uncompressed size: 4, name: data.txt
110           0x6E            End of Zip archive, footer length: 22


C:\Users\Max Wang\Desktop\123
> binwalk out7.zip

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Zip archive data, encrypted at least v2.0 to extract, compressed size: 18, uncompressed size: 4, name: data.txt
110           0x6E            End of Zip archive, footer length: 22
```

安全客 (bobao.360.cn)

用如下的脚本碰撞出所有压缩包中的数据：

```
#coding:utf-8
import zipfile
import string
import binascii

def CrackCrc(crc):
    for i in dic:
        for j in dic:
            for p in dic:
                for q in dic:
                    s = i + j + p + q
                    if crc == (binascii.crc32(s) & 0xffffffff):
                        #print s
                        f.write(s)
                        return

def CrackZip():
    for I in range(68):
        file = 'out' + str(I) + '.zip'
        f = zipfile.ZipFile(file, 'r')
        GetCrc = f.getinfo('data.txt')
        crc = GetCrc.CRC
        #以上 3 行为获取压缩包 CRC32 值的步骤
        #print hex(crc)
        CrackCrc(crc)

dic = string.ascii_letters + string.digits + '+/='

f = open('out.txt', 'w')
CrackZip()
f.close()
```

安全客 (bobao.360.cn)

此题较为繁琐，之后的步骤不再展开

另：限于CPU的能力，CRC碰撞只能用于压缩文件较小的情况

0x07. 修改格式

这种情况花样较多，难以做一个详细的总结，因此只列举最常见的缺少文件头或文件尾。

放一个zip文件格式讲的较清楚的链接，通过对zip文件格式的了解，可以解释之前伪加密的问题，同时也可以对缺少文件头或文件尾有更直观的认识。

如上为正常zip，缺头zip和缺尾zip的binwalk扫描结果，根据扫描结果用16进制编辑器添加文件头或文件尾，即可修复zip。

总结

Zip不仅是我们生活中常用到的一种文件格式，在CTF中也经常遇到，这里做了一个关于CTF中zip的总结，如果对读者有帮助，鄙人不胜荣幸。