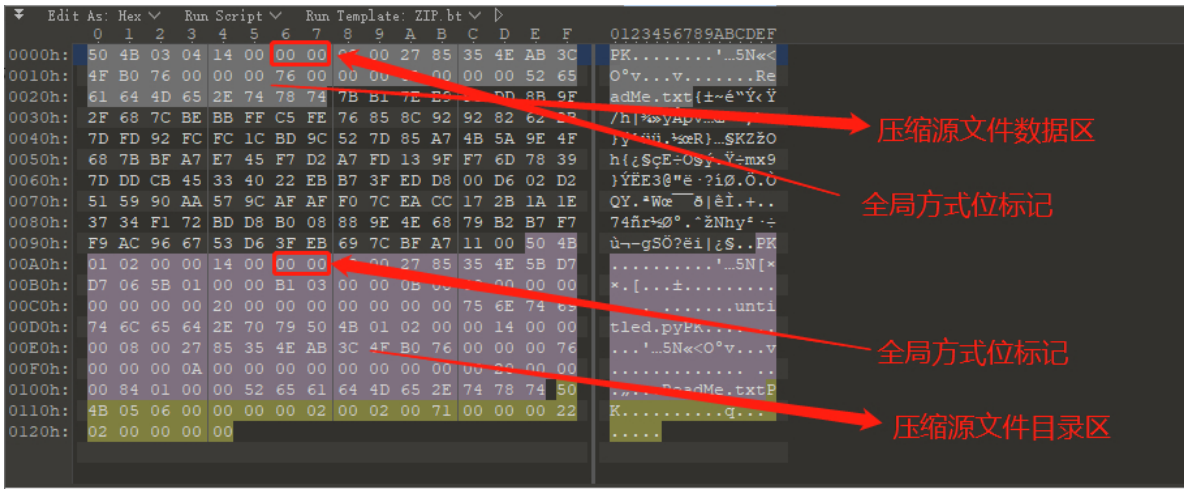
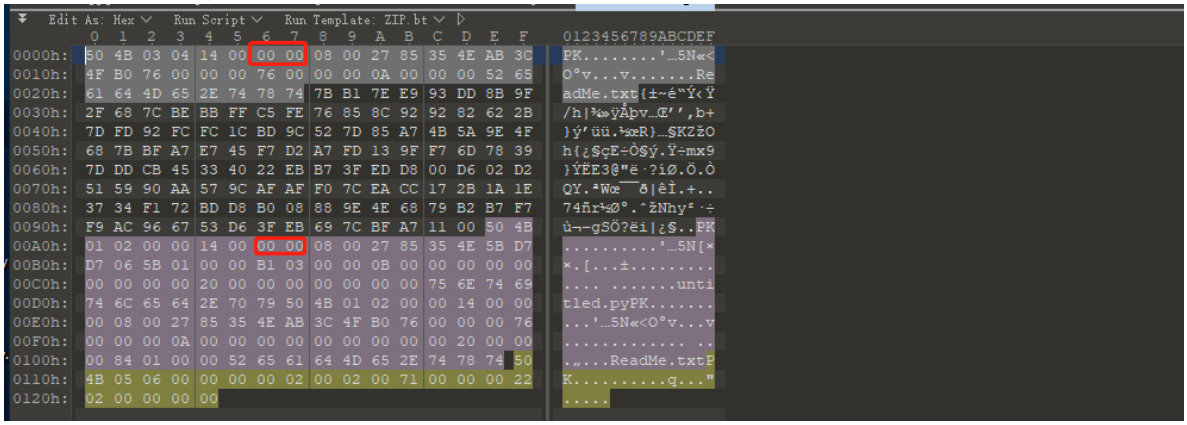
伪加密、爆破、明文攻击、CRC32碰撞

zip文件格式：
zip文件由三部分组成：压缩的文件内容源数据、压缩的目录元数据、目录结束标识结构
详情点击：[zip格式说明]

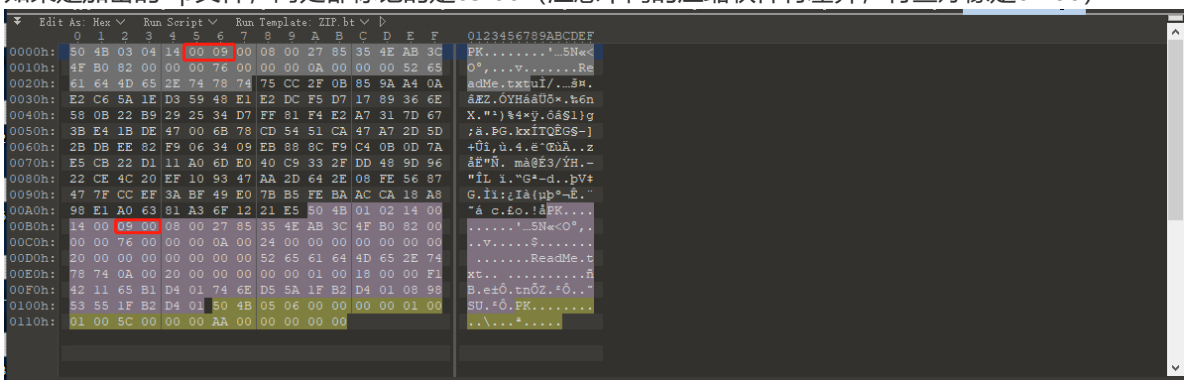
zip伪加密



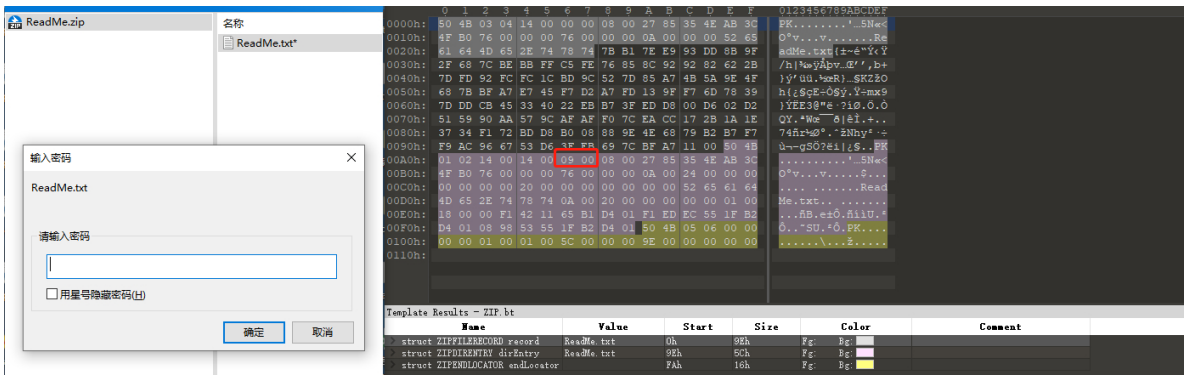
若是没有加密的zip文件，两处标记都是00 00



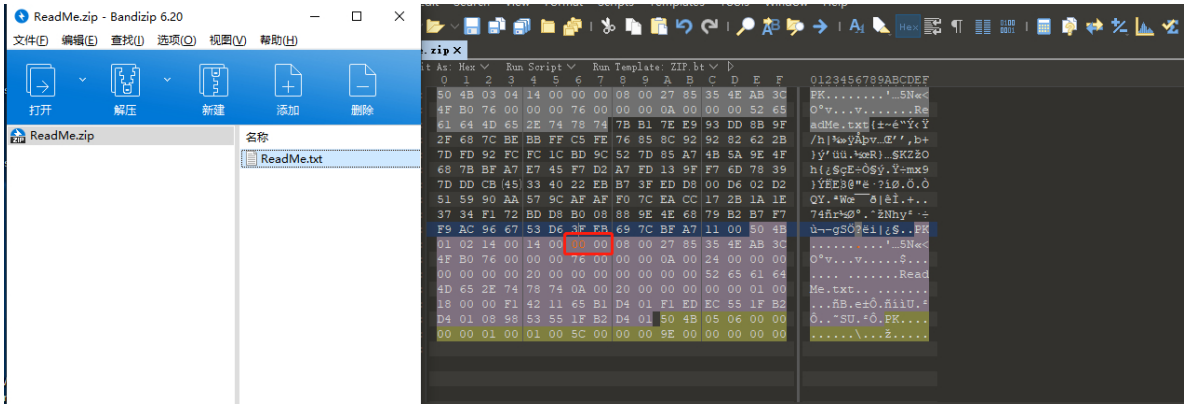
如果是加密的zip文件，两处都标记的是09 00（注意不同的压缩软件有差异，有些好像是01 00）



若将未加密的zip文件中的压缩源文件目录区的全局方式位标记改为01 00 （或者09 00），就会被压缩软件认为已加密，这就是所谓的伪加密了



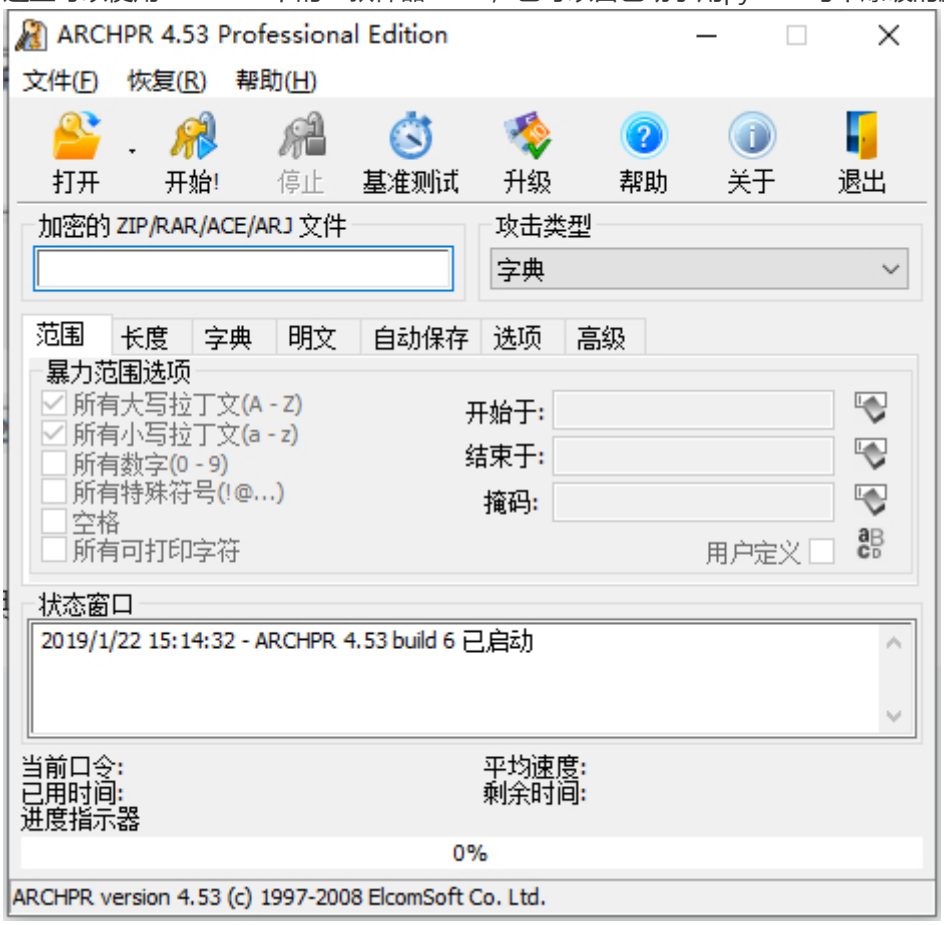
破解伪加密的zip，只要把压缩文件目录区的全局方式标记改为00 00 （除windows外的系统（如kali）可直接打开伪加密压缩包）



爆破

爆破：逐个尝试选定集合中的可以组成的所有密码，直到遇到正确的密码。
分为暴力破解、掩码破解、字典破解这几种
1、暴力破解：选择密码范围，长度等，由软件组合生成密码进行破解
2、掩码破解：知道密码中的一部分，只需要按照规则构造其余部分进行破解
3、字典破解：通常是多数用户常用的一些密码集合，导入字典文件用字典中的密码进行破解（取决你的字典）

这里可以使用Windows下的一款神器AZPR，也可以自己动手用python写个爆破的脚本。



明文攻击

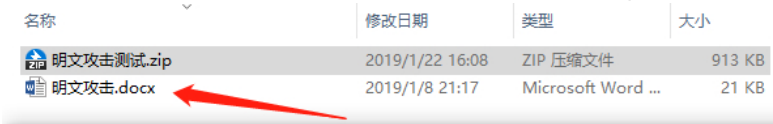
明文攻击（Known plaintext attack）：是一种攻击模式，指攻击者已知明文、密文及算法，求密钥的过程。

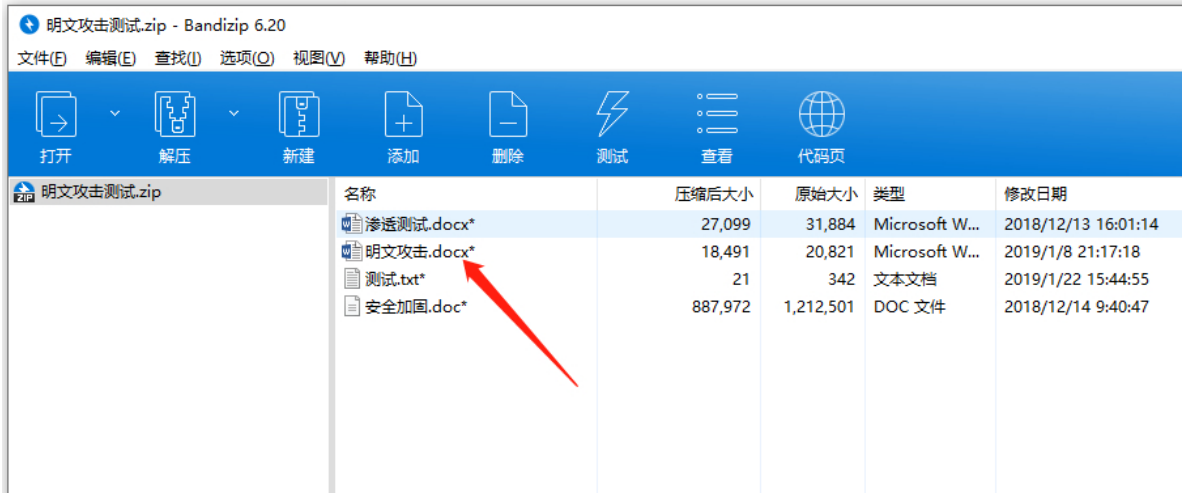
明文攻击是一种较为高效的攻击手段，如：当你不知道一个zip压缩包文件的密码，但是你有zip中的一个已知文件（文件大小要大于12Byte）时，因为同一个zip压缩包里的所有压缩文件使用同一个加密密钥来加密的，所以可以用已知文件来找加密密钥，利用密钥来解锁其他加密文件。

简单来说就是，zip明文攻击就是利用已知文件找到加密密钥，利用密钥来解释其他加密文件，因为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。

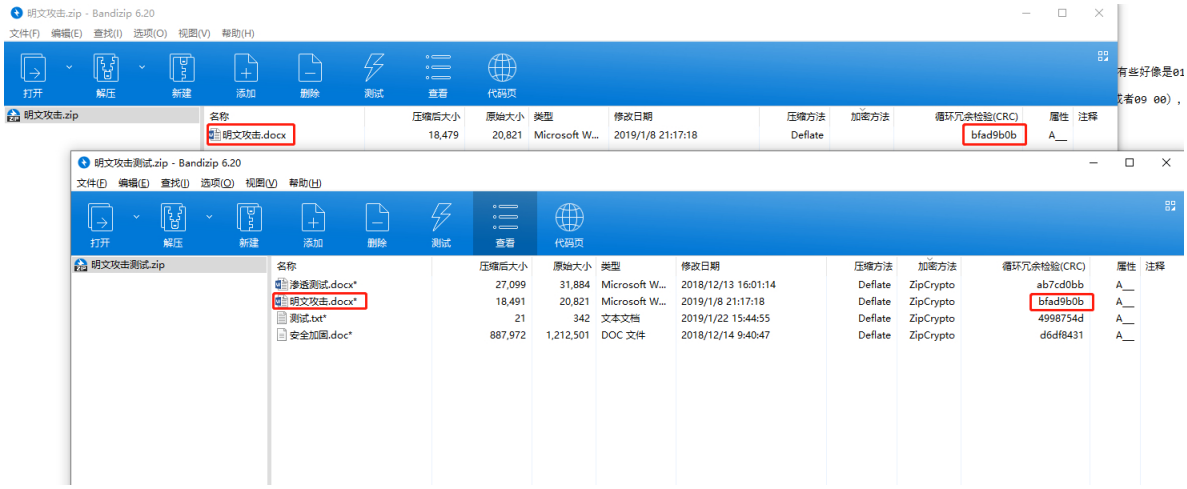
这里举个例子：

现在我压缩了带密码的四个文件，已知明文攻击测试.zip中的明文攻击.docx



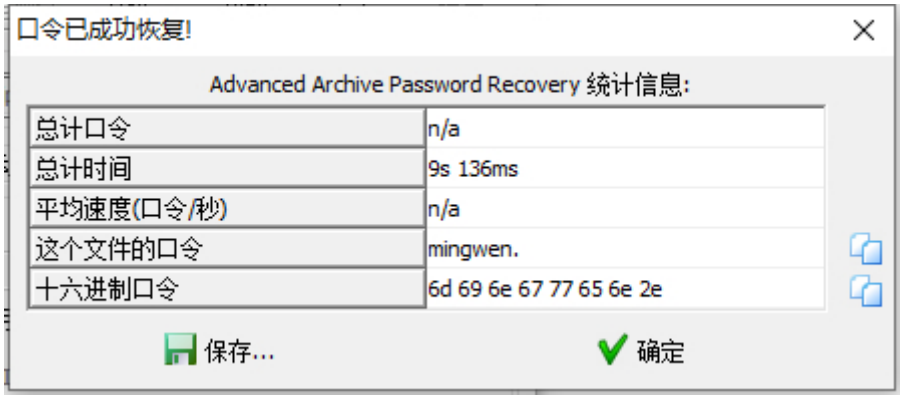


进行对已知文件进行压缩，对比crc值是否跟加密文件中的crc值一致

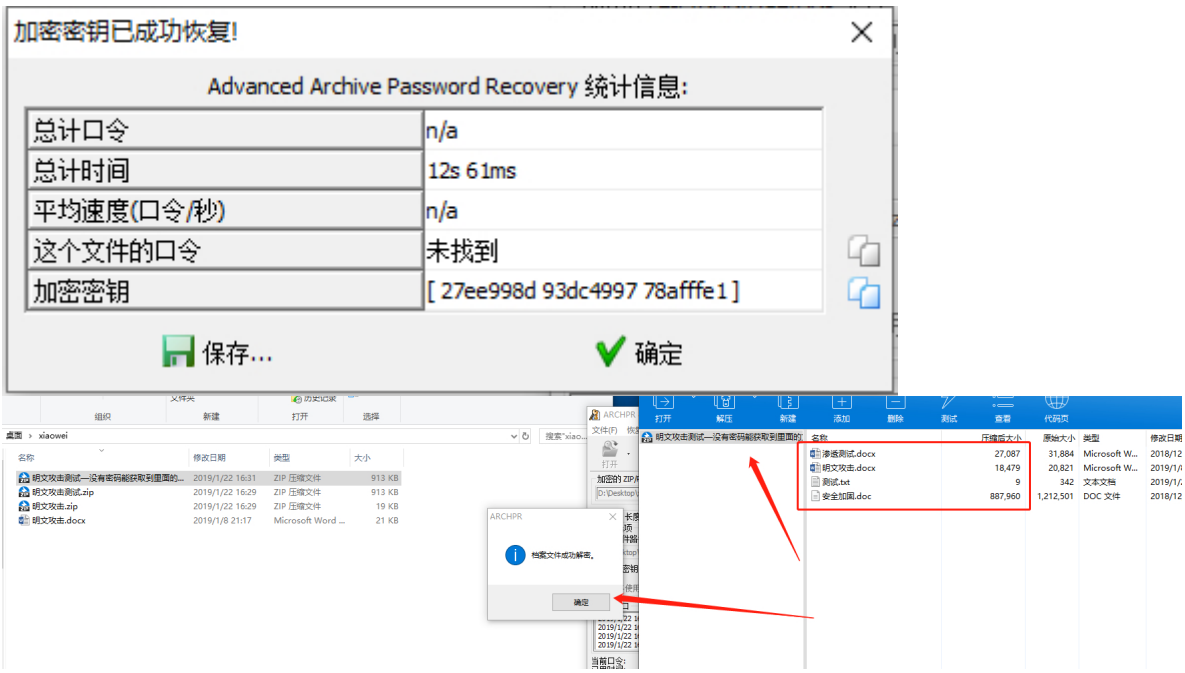


利用AZPR进行明文攻击

成功获取到密文



注意：当明文的大小比较小时，或者密文过长，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，如果出现错误可以多试几款压缩软件。



CRC32碰撞

CRC32：CRC本身是“冗余校验码”的意思，CRC32则标识会产生一个32bit（8位十六进制）的校验值。

CRC校验实用程序库，在数据存储和数据通讯领域，为了保证数据的正确，就不得不采用检错的手段。在诸多检错手段中，CRC是最著名的一种。CRC的全称是循环冗余校验。

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中的内容。

每个文件都有唯一的CRC32值，即便数据中一个bit发生变化，也会导致CRC32值不同。若是知道一段数据的长度和CRC32值，便可穷举数据，与其CRC32对照，以此达到暴力猜解的目的。但限于CPU的能力，通常只适用于较小文本文件。

案例：可以参考我写的Bugku-好多压缩包那篇文章。
bugku中的例子脚本如下：

```
# coding:utf-8

import binascii
import string
import zipfile
import base64

dicts = string.printable  # 可打印字符的字符串。ascii码33-126号


def collision_crc(crc):
    global out_file
    for a in dicts:
        for b in dicts:
            for c in dicts:
                for d in dicts:
                    strings = a + b + c + d
                    strings = strings.encode('utf-8')
                    if crc == (binascii.crc32(strings)):
                        out_file.write(strings.decode('utf-8'))
                        # print(strings)
                        return
```

```
                        # 以上定义一个方法，组合随机字符与CRC进行碰撞，判断如果相等及写入文件


def obtain_zip():
    for i in range(68):
        file = 'out' + str(i) + '.zip'
        zip_file = zipfile.ZipFile(file, 'r')  # 读取创建zip_file对象
        get_crc = zip_file.getinfo('data.txt')  # 压缩文件夹里的data.txt文件，获取文档内指定的文件信息
        crc = get_crc.CRC
        #  以上定义一个方法，获取68个zip的CRC的值
        collision_crc(crc)  # 再调用collision方法传参


out_file = open('out.txt', 'w')
obtain_zip()
out_file.close()

out_file2 = open('out.txt', 'r')

with open('flag.rar', 'wb') as rar:
    rar.write(base64.b64decode(out_file2.read()))   # 二进制将转换后的base64位写入文件
```