ctf中zip加密三种破解方式

0x00

暴力破解

手动生成字典——木头字典生成器

ziperello —— 字典破解密码

局限性在于，只要把密码设置在八位数以上并且混杂大小写特殊字符基本就没暴力破解什么事情了。

0x01

伪加密

zip16进制标准格式

```
压缩源文件数据区：
50 4B 03 04：这是头文件标记（0x04034b50）
14 00：解压文件所需 pkware 版本
00 00：全局方式位标记（有无加密）
08 00：压缩方式

压缩源文件目录区：
50 4B 01 02：目录中文件文件头标记(0x02014b50)
3F 00：压缩使用的 pkware 版本
14 00：解压文件所需 pkware 版本
00 00：全局方式位标记（有无加密，这个更改这里进行伪加密，改为09 00打开就会提示有密码了）
08 00：压缩方式
```

用Python脚本检测伪加密，并且解密

```
import sys

def removefade(para1):
    # 读取原zip文件
    zipfile = open(para1,'rb')
    zipfile_content = zipfile.read().encode('hex')
    zipfile.close()

    zf_len = len(zipfile_content)

    # 查找加密标志位并处理
    for i in xrange(zf_len):
        comp_con = zipfile_content[i:i+8]
        if comp_con == '504b0102':
            zipfile_content = zipfile_content[:i+17] + '0' + zipfile_content[i+18:]
        if comp_con == '504b0304':
            zipfile_content = zipfile_content[:i+13] + '0' + zipfile_content[i+14:]


    # 将处理后内容写入新文件
    newzip = open(para1[:-4] + 'repair.zip','wb')
    newzip.write(zipfile_content.decode('hex'))
    newzip.close()
    print('Done')


if __name__ == '__main__':
    if(len(sys.argv) == 1):
        print('\nusage example:')
        print(' python dzipfade.py a.zip\n')
    else:
        para = sys.argv
        removefade(para[1])
```

0x02明文攻击

zip加密方法其实有两种，第一种是AES加密，第二种是标准zip2.0的加密方式。现在主流的加密方法是标准zip2.0加密方法。AES相比于标准zip2.0更加安全，但是AES并不能被大多数的压缩软件支持，标准zip2.0的通用性更强。

标准**zip2.0**加密方式

设置的密码会生成3个4 bytes的key ，总共96bit的秘钥，分别加密压缩文件内的所有文件。因此，只要知道这个96bit的秘钥，在不知道明文密码的情况下，也可以对加密zip进行解压

明文攻击的前提

知道一段至少13bytes的明文，知道的明文越多，成功率就越高。要采用相同的加密算法，**CRC32**值相同。

准备工作

1. 一个加密了的压缩文件，其中包含已知明文文件

555.txt是我们已知的明文文件

2. 将已知的明文文件压缩为zip

可以看到加密和不加密的CRC32值是相同的，说明它们采用了相同的压缩方式，这是明文攻击的前提。

加密原理

加密方法

PKZIP中使用的加密方法由Roger Schlafly提供。ZIP文件在解压缩前必须先解密。每个加密文件具有一个12字节的加密文件头扩展信息,存储于数据区的起始位置,加密前先设置一个起始值,然后被三个32位的密钥加密。密钥被使用者提供的口令初始化。12个字节加密之后,由PKZIP的伪随机数产生方法,结合PKZIP中使用CRC-32算法对密钥进行更新。

具体实施分为三步:

1.用口令对三个32位密钥初始化。

K(0)=305419896,K(1)=591751049,K(2)=878082192

循环 for i=0 to length(password)-1

调用更新密钥函数 update_keys(password(i))

结束循环(循环口令长度次)

其中更新密钥函数为:

update_keys(char):

Key(0)=crc32(key(0),char)

Key(1)=Key(1)+(Key(0)& 000000ffH)

Key(1)=Key(1)*134775813+1

Key(2)=crc32(Key(2),Key(1)〉〉24)

end update_keys

CRC32函数中,给定一个4字节的CRC值和一个字符,返回一个由CRC-32算法更新的CRC。具体为:

crc32(c,b)=crc32tab[(c^b)&0xff]^(c> >8),crc32tab[256]的值为固定的256个4字节数。

2.读取并加密12字节的加密头,再次对密钥进行初始化。

将12个字节的加密头读入缓冲区buffer(0)至buffer(11),循环for i=0 to 11

C=buffer(i)^decrypt_byte()

update_keys(C)

buffer(i)=C

结束循环(循环12次)

其中的decrypt_byte()函数为:

unsigned char decrypt_byte()

local unsigned short temp

temp=Key(2)¦2

decrypt_byte=((temp*(temp^1))> > 8)&0xff

end decrypt_byte

该步结束后,缓冲区中最后的二个字节buffer(10)和buffer(11)将成为加密文件校验码的二个最高位(按低至高顺序存放)。对ZIP加密文件进行解压缩前,PKUNZIP软件将使用者提供的口令按上述二个步骤进行处理,得到的结果与校验码的二个高位字节进行比较,只有当提供了正确的口令时,结果一致,才能进行后续的解压缩过程,否则,PKZIP报告错误信息,程序自动结束。

3.读取压缩的数据流并以加密密钥对其进行加密。

压缩数据流按下述过程加密:

循环 直至数据流结束

C=数据流的一个字节

temp=C^decrypt_byte()

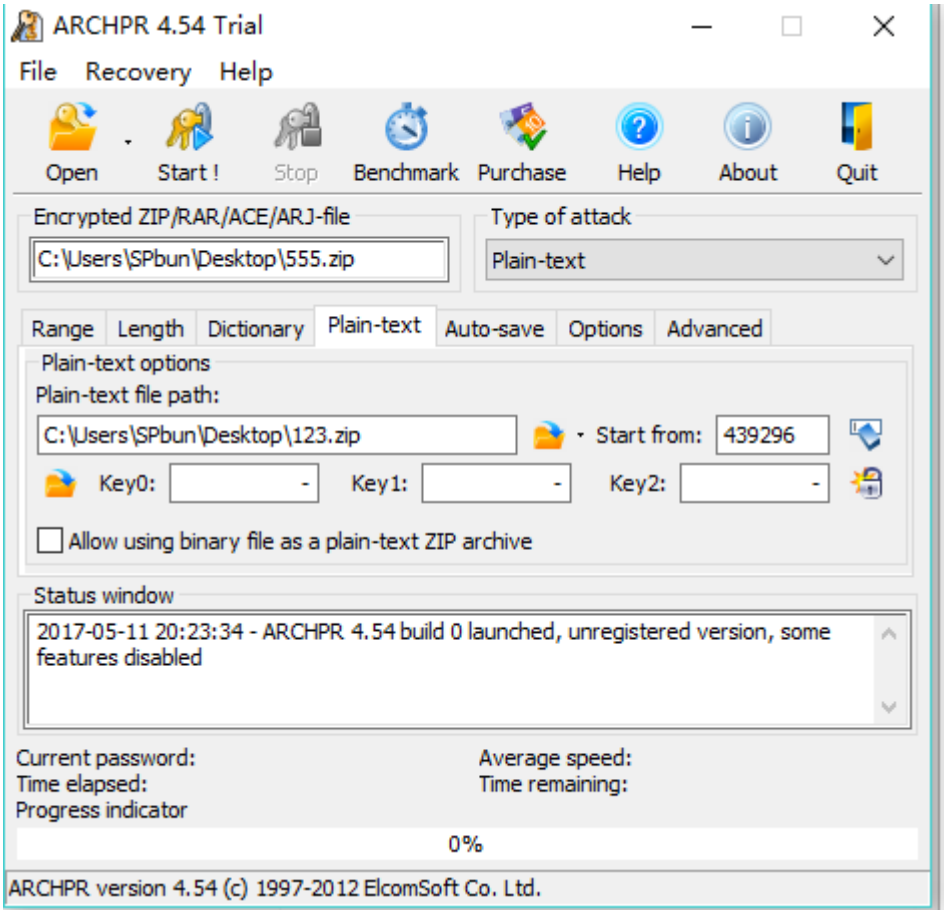
update_keys(temp)

```
            输出temp
            结束循环
```

3.明文攻击的工具

1.Advanced Archive Password Recovery

这个工具不仅能进行明文攻击，还可以暴力破解或者密码攻击。



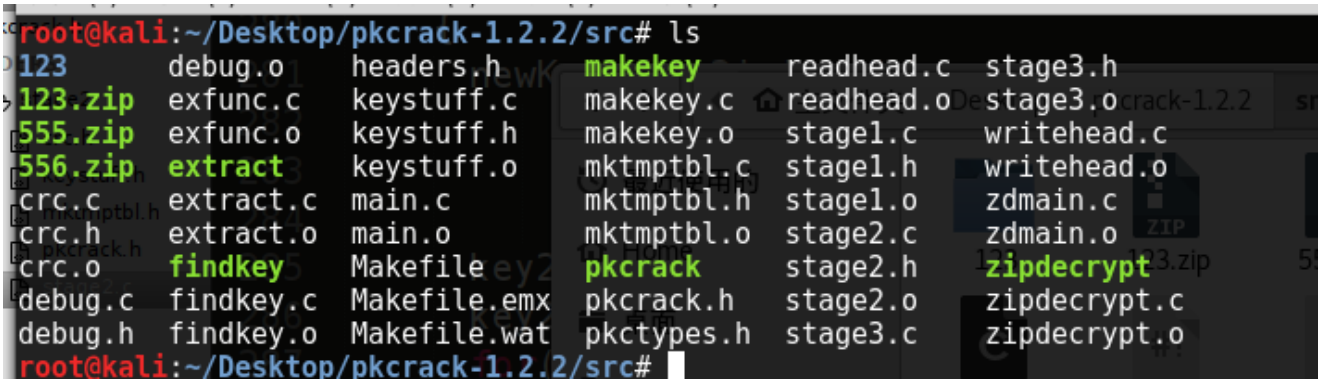
简单易用，但是效率讲真不高，而且成功率也不高。碰撞出key也不马上告诉我，要跑完所有可能值最后才弹出来。大概因为这是免费版吧。

2.PKcrack

这个工具专门用于明文攻击，相比于前者，效率有了明显的提高，因为它在尝试用可能的key加密明文前，会进行二次筛选。

但这个工具只能在xp或者linux上面运行。win10会报错（32位软件不兼容），但是我在xp上面用会闪退。所以最后在kali上运行

安装过程很简单，解压后进去make一下就会多出几个编译成功的可执行文件，我们要用到的是pkcrack



要用到六个参数

-c 加密zip中的明文文件的文件名

-C 加密zip的文件名

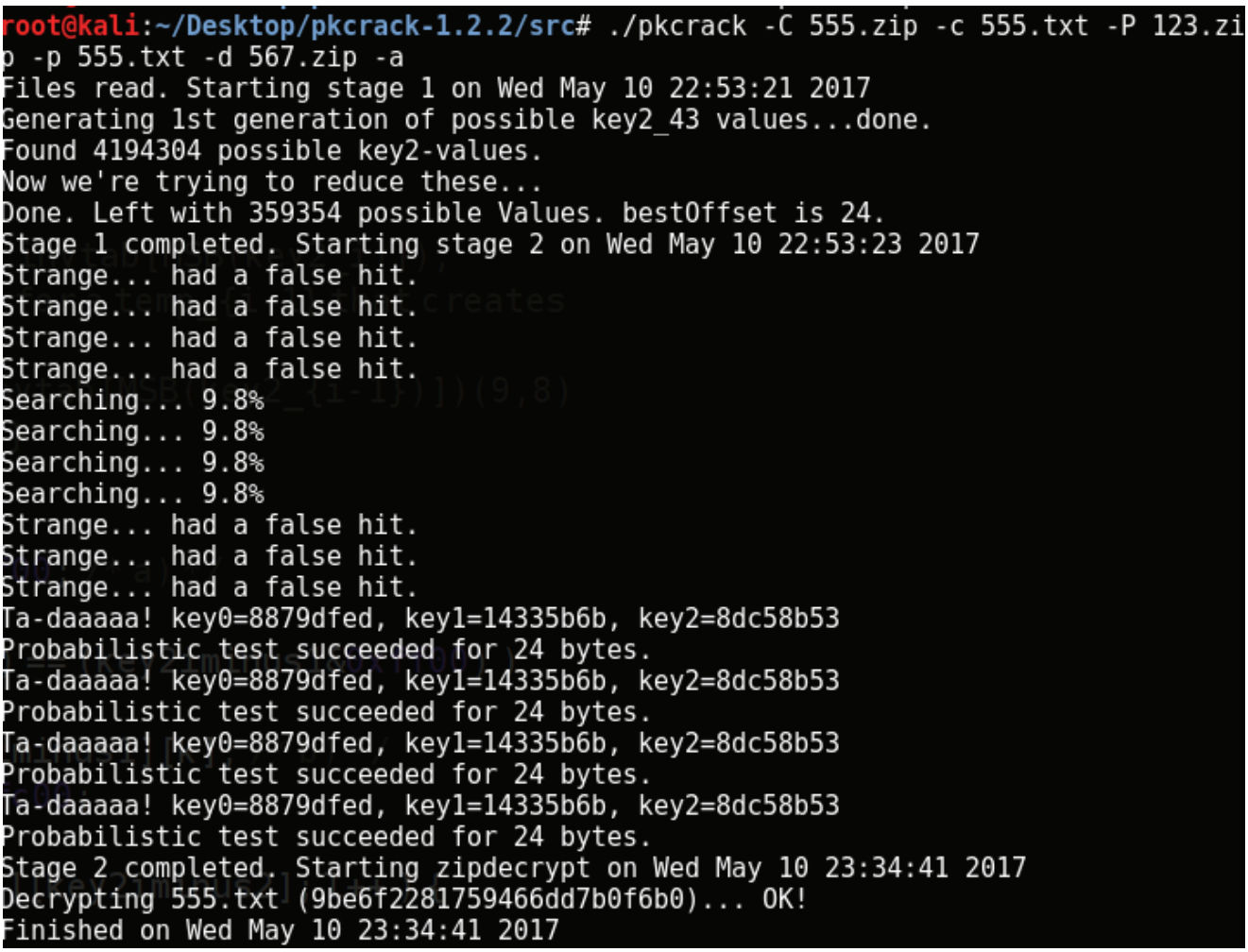
-p 明文zip中的明文文件的文件名

-P 明文zip的文件名

-d 成功攻击后输出的zip文件名

-a 成功碰撞后显示key值

```
./pkcrack -C 555.zip -c 555.txt -P 123.zip -p 555.txt -d 567.zip -a
```



虽然效率有所提高，但是还是要破解四十分钟