

# ZIP 已知明文攻击深入利用

“ 你加密的压缩包比你想象中的还不安全！

## 一、前言

你加密的压缩包比你想象中的还不安全！

哪怕对于信息安全人员来说，很多时候给压缩包加上一个密码就以为的是万事大吉了。但事实是，很多情况下，你的加密压缩包，远远没有你想象的安全。

## 二、内容概要

以往进行 ZIP 已知明文攻击，通常需要一个完整的明文文件。而本文讨论的攻击方式只需要知道加密压缩包内容的 12 个字节，即可进行攻击破解降低了已知明文的攻击难度。同时，结合各类已知的文件格式，更扩宽了 ZIP 已知明文攻击的攻击面。

## 三、正文

### 3.1 ZIP 已知明文攻击的一般利用

以往出现在网络安全竞赛中的已知明文攻击考点，或者大部分网上的文章，都需要知道加密 zip 文件中的一个完整明文文件并且要求明文以相同的标准被压缩，这才有可能攻击成功。

其实传统的已知明文攻击要成功需要三个条件，在此我将条件列出来：

- 完整的明文文件

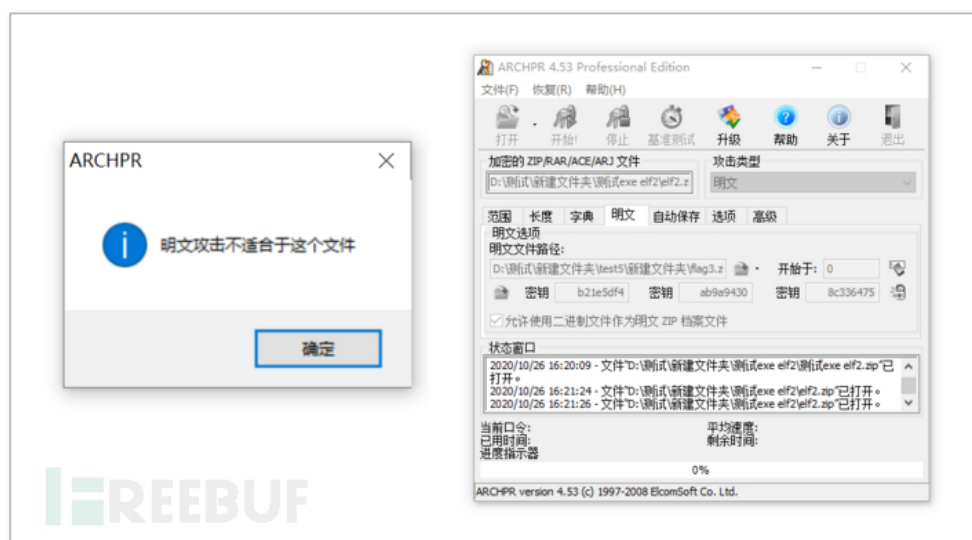
- 明文文件需要被相同的压缩算法标准压缩（也可理解为被相同压缩工具压缩）

- 明文对应文件的加密算法需要是 ZipCrypto Store

第三点是我们实际应用中常常会被忽略的。因竞赛中遇到的题目，都是提前设置好的。

- AES256-Deflate/AES256-Store 加密的文件不适用于明文攻击。

名称	算法	大小	压缩后大小	修改时间
1312.txt	AES-256 Deflate	4	34	2020-11-10 13:38
mingwen.txt	AES-256 Deflate	18	32	2020-11-10 13:39



ZIP 的加密算法大致分为两种 ZipCrypto 和 AES-256, 各自又分 Deflate 和 Store。

- ZipCrypto Deflate
- ZipCrypto Store
- AES-256 Deflate
- AES-256 Store

ZipCrypto 算是传统的 zip 加密方式。只有使用 ZipCrypto Deflate /Store 才可以使用 ZIP 已知明文攻击进行破解。

传统的 ZIP 已知明文攻击利用，windows 下可以使用 AZPR，linux 下可以使用 pkcrack。

## 3.2 ZIP 已知明文攻击的深入利用

本文要探讨的攻击方法并不需要知道压缩包文件中完整的内容，只需要

本文要探讨的攻击方法并不需要知道压缩包中完整的明文，只需在已知加密压缩包中的少部分明文字节时即可进行攻击破解。而各类文件都有其自身固定的文件格式，结合这类格式，极大扩展了 ZIP 明文攻击的攻击面。

具体要求如下：

至少已知明文的 12 个字节及偏移，其中至少 8 字节需要连续。

明文对应的文件加密方式为 ZipCrypto Store

该方法对于 ZIP 加密的算法有要求，明文对应的文件加密方式需要为 ZipCrypto Store。经测试，Winrar (v5.80)、7zip (v19.00) 默认状态下加密使用的就是 AES256 算法，直接排除。360 压缩 (v4.0.0.1220)、好压 (v6.2) 使用的是 ZipCrypto，不固定使用 Store 或 Deflate（如果要固定使用 ZipCrypto Store 算法加密，可以在压缩的时候指定压缩方式为“存储”）。

以下破解用到的压缩包，都是经 360 压缩或者好压加密打包的。

### 3.2.1 使用到的工具项目

bkcrack: <https://github.com/kimci86/bkcrack>

bkcrack 安装：

```
apt install cmake -y
```

```
cmake .
```

```
make // 在 src 下生成 bkcrack 文件
```

```
cp bkcrack /usr/sbin/bkcrack // 作为系统命令使用
```

bkcrack 常用参数：

-c 提取的密文部分

-p 提取的明文部分

-x 压缩包内目标文件的偏移地址 部分已知明文值

-C 加密压缩包

-o offset -p 参数指定的明文在压缩包内目标文件的偏移量

在此我们不是“造轮子”，而是“使用轮子”，偏向于实操，利用已有的手段工具去解决现有的问题。话不多说，上实操案例。

### 3.3 实操案例

案例中演示的压缩包等，都可在文末附件中下载。

#### 3.3.1 加密文本破解

文本类文件被加密成 zip 时，有很大的概率以 ZipCrypto Store 方式加密存储。

创建加密 zip:

生成 uuid，将字符串“flag{16e371fa-0555-47fc-b343-

74f6754f6c01}”保存为 flag.txt。然后用 360 压缩将文件添加为加密 ZIP: flag\_360.zip

名称	算法	大小	压缩后大小
flag.txt	ZipCrypto Store	42	54

攻击破解:

采用 8+4 的方式提取部分已知明文来进行攻击测试，

flag{16e371fa-0555-47fc-b343-74f6754f6c01}

我们利用以下这部分明文，来进行攻击破解:

\*lag{16e3\*\*\*\*\*74f6\*\*\*\*\*

-----  
-----  
-----

#准备已知明文

```
echo -n "lag{16e3"> plain1.txt // 连续的 8 明文
```

```
echo -n "74f6" | xxd // 额外明文的十六进制格式，  
37346636
```

#攻击

```
bkcrack -C flag_360.zip -c flag.txt -p plain1.txt -o 1 -x 29  
37346636
```

#由于时间较长，为防止终端终端导致破解中断，可以加个小技巧

```
bkcrack -C flag_360.zip -c flag.txt -p plain1.txt -o 1 -x 29  
37346636 > 1.log& // 后台运行，结果存入 1.log
```

// 加上 time 参数方便计算爆破时间

```
time bkcrack -C flag_360.zip -c flag.txt -p plain1.txt -o 1 -x
```

29 37346636 > 1.log&

// 查看爆破进度

tail -f 1.log

-----  
-----  
-----

注：· -p 指定的明文不需要转换，-x 指定的明文需要转成十六进制

· 提到的偏移都是指“已知明文在加密前文件中的偏移”。

历时近 16 分钟，成功得到密钥，这不是压缩包的加密密码，而是 ZIP 内部的三段密钥。

```
root@labs-a3:~/blus/1027# time bkrack -C flag_360.zip -c flag.txt -p plain1.txt -o 1 -x 29 37346636 > 1.log&
[1] 19625
root@labs-a3:~/blus/1027# tail -f 1.log
Generated 4194304 Z values.
[22:15:46] Attack on 4194304 Z values at index 8
39.3 % (1646785 / 4194304)
[22:31:43] Keys
b21e5df4 ab9a9430 8c336475

real    15m56.836s
user    380m38.762s
sys     0m29.351s
```

b21e5df4 ab9a9430 8c336475

使用该密钥进行解密：

```
bkrack -C flag_360.zip -c flag.txt -k b21e5df4 ab9a9430
8c336475 -d flag.txt
```

```
root@labs-a3:~/blus/1027# bkrack -C flag_360.zip -c flag.txt -k b21e5df4 ab9a9430 8c336475 -d flag.txt
Wrote deciphered text.
root@labs-a3:~/blus/1027# cat flag.txt
flag{16e371fa-0555-47fc-b343-74f6754f6c01}root@labs-a3:~/blus/1027#
```

### 3.3.2 利用 PNG 图片文件头破解

PNG 文件头：

PNG_header																
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52   %PNG.....IHDR

89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52

进口 16 个字节的要求，含 4 张图片，含 5 个 1 字节，把打包成加

满足，12 个字节的要求。拿一张图片和一个 flag.txt 一起打包成加密 ZIP 压缩包：png4.zip

名称	算法	大小	压缩后大小
2.png	ZipCrypto Store	18 670	18 682
flag.txt	ZipCrypto Store	38	50

攻击破解：

-----  
-----  
-----  
#准备已知明文

```
echo 89504E470D0A1A0A0000000D49484452 | xxd -r -ps >  
png_header
```

#攻击

```
time bkcrack -C png4.zip -c 2.png -p png_header -o 0  
>1.log&  
tail -f 1.log
```

-----  
-----  
-----

耗时近 7 分钟破解出秘钥：e0be8d5d 70bb3140 7e983fff

```
root@labs-a3:~/blus/1027# tail -f 1.log  
Generated 4194304 Z values.  
[02:48:48] Z reduction using 8 bytes of known plaintext  
100.0 % (8 / 8)  
889179 values remaining.  
[02:48:49] Attack on 889179 Z values at index 7  
82.4 % (733107 / 889179)  
[02:55:43] Keys  
e0be8d5d 70bb3140 7e983fff  
  
real 6m55.013s  
user 164m33.940s  
sys 0m13.780s
```

利用秘钥解密文件：

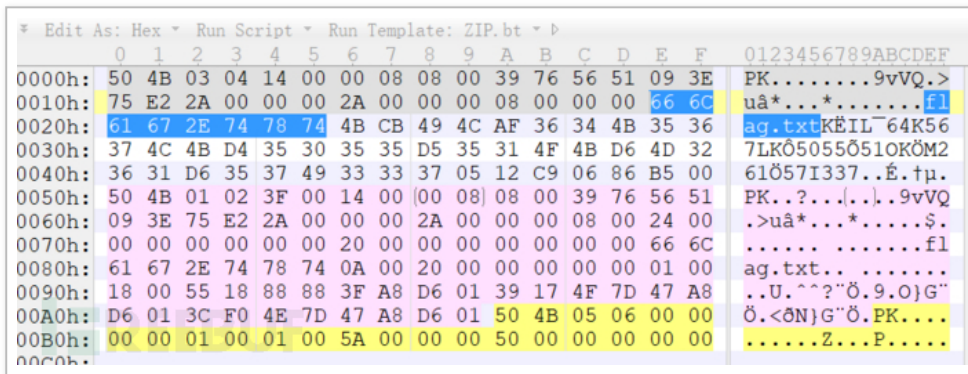
```
bkcrack -C png4.zip -c flag.txt -k e0be8d5d 70bb3140  
7e983fff -d flag.txt
```

```
root@labs-a3:~/blus/1027# bkcrack -C png4.zip -c flag.txt -k e0be8d5d 70bb3140 7e983fff -d flag.txt
```

```
Wrote deciphered text.
root@labs-a3:~/blus/1027# cat flag.txt
flag{2f09f201dc590e0ecd71a90272996666}root@labs-a3:~/blus/1027#
```

### 3.2.3 利用压缩包格式破解

将一个名为 flag.txt 的文件打包成 ZIP 压缩包后，你会发现文件名称会出现在压缩包文件头中，且偏移固定为 30。且默认情况下，flag.zip 也会作为该压缩包的名称。



所以，当一个加密压缩包中存在另一个 ZIP 压缩包时，且能够知道或猜测该压缩包内的文件名称时，可以尝试进行已知明文攻击。

将 flag.zip 与其他文件（选用一张图片）一起用好压打包成加密 ZIP 压缩包：test5.zip

名称	算法	大小	压缩后大小
2.png	ZipCrypto Deflate	2 409 786	2 404 831
flag.zip	ZipCrypto Store	192	204

已知的明文片段有：

- “flag.txt” 8 个字节，偏移 30
- ZIP 本身文件头：50 4B 03 04 ， 4 字节

8+4，满足了破解的最低要求

攻击：

```
-----
-----
-----
```

```
echo -n "flag.txt" > plain1.txt    //--n 参数避免换行，不然文件中会出现换行符，导致攻击失效
time bkcrack -C test5.zip -c flag.zip -p plain1.txt -o 30 -x 0 504B0304 >1.log&
```

tail -f 1.log

-----

-----

-----

得到秘钥：

b21e5df4 ab9a9430 8c336475



```
root@labs-a3:~/blus/1026/test2# tail -f 1.log
Generated 4194304 Z values.
[23:17:41] Attack on 4194304 Z values at index 37
79.4 % (3331264 / 4194304)
[23:51:24] Keys
b21e5df4 ab9a9430 8c336475

real    33m43.071s
user    772m4.320s
sys     0m58.340s
```

利用秘钥解密：

bkcrack -C test5.zip -c flag.zip -k b21e5df4 ab9a9430  
8c336475 -d flag.zip

flag.zip 可以直接成功解密。

但若想解密 2.png，由于是 ZipCrypto deflate 加密的，所以解密后  
需要 bkcrack/tool 内的 inflate.py 脚本再次处理。

-----

-----

-----

bkcrack -C test5.zip -c 2.png -k b21e5df4 ab9a9430  
8c336475 -d 2.png  
python3 inflate.py <2.png> 2\_out.png

-----

-----

-----

### 3.2.4 EXE 文件格式破解

EXE 文件默认加密情况下，不太会以 store 方式被加密，但它文件  
格式中的明文及其明显，长度足够。如果加密 ZIP 压缩包出现以  
store 算法存储的 EXE 格式文件，很容易进行破解。

大部分 exe 中都有这相同一段，且偏移固定为 64：





#查看进度

tail -f 1.log

-----  
-----  
-----

很快就解出了密钥:

```
root@labs-a3:~/blus/1026/test3# time bkrack -C nc64.zip -c nc64.exe -p mingwen -o 64 >1.log&
[1] 18210
root@labs-a3:~/blus/1026/test3# tail -f 1.log
Generated 4194304 Z values.
[04:02:50] Z reduction using 56 bytes of known plaintext
100.0 % (56 / 56)
146776 values remaining.
[04:02:51] Attack on 146776 Z values at index 71
9.7 % (14191 / 146776)
[04:03:00] Keys
b21e5df4 ab9a9430 8c336475

real    0m10.304s
user    3m22.158s
sys     0m0.268s
```

b21e5df4 ab9a9430 8c336475



解密:

bkrack -C nc64.zip -c nc64.exe -k b21e5df4 ab9a9430  
8c336475 -d nc64.exe

### 3.2.5 流量包 pcapng 格式解密

这个有例题: 钓鱼城杯 - 量子加密

具体格式介绍及解法参考官方的 writeup, 已打包在附件中

名称	算法	大小	压缩后大小
 capture.pcapng	Store	886 316	886 328
 hint_for_capture.txt	Store	36	48

加密算法都是 Store

Offset (h)	00	01	02	03	
00000000	0A	0D	0D	0A	....
00000004	84	00	00	00	....
00000008	4D	3C	2B	1A	M<+.
0000000C	01	00	00	00	....
00000010	FF	FF	FF	FF	yyyy

00000014	FF	FF	FF	FF	ÿÿÿÿ
00000018	03	00	2B	00	..+.
0000001C	36	34	2D	62	64-b
00000020	69	74	20	57	it W
00000024	69	6E	64	6F	indo
00000028	77	73	20	37	ws 7
0000002C	20	53	65	72	Ser
00000030	76	69	63	65	vice

选用第二段文件头格式：

00 00 4D 3C 2B 1A 01 00 00 00 FF FF FF FF FF FF FF FF

攻击：

```
-----
-----
-----
```

```
echo -n "00004D3C2B1A01000000FFFFFFFFFFFFFFFFFFFF" | xxd -
r -ps > pcap_plain1
```

```
time bkcraack -C 3.zip -c capture.pcapng -p pcap_plain1 -o 6
```

```
-----
-----
-----
```

```
root@labs-a3:~/blus# time bkcraack -C 3.zip -c capture.pcapng -p pcap_plain1 -o 6
Generated 4194304 Z values.
[04:26:26] Z reduction using 10 bytes of known plaintext
100.0 % (10 / 10)
642004 values remaining.
[04:26:27] Attack on 642004 Z values at index 13
79.1 % (507594 / 642004)
[04:31:17] Keys
e33a580c c0c96a81 1246d892

real    4m50.327s
user    115m5.465s
sys     0m7.134s
```

解密：

```
bkcraack -C 3.zip -c capture.pcapng -k e33a580c c0c96a81
1246d892 -d out.pcapng
```

### 3.2.6 网站相关文件破解

网站目录中充斥着大量类型的文件，哪怕被打包成加密 ZIP，也很容易找到突破口。

例如：

robots.txt 的文件开头内容通常是 User-agent: \*

html 文件开头通常是 <!DOCTYPE html>

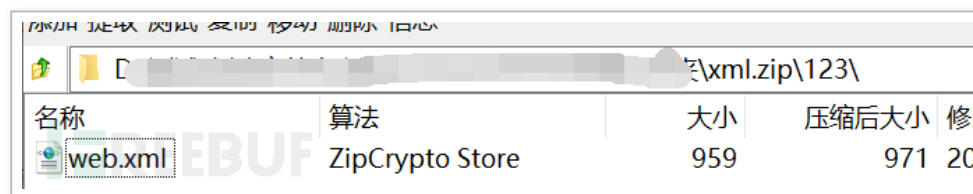
xml 文件开头通常是 <?xml version="1.0" encoding="UTF-8"?>

在此以 web.xml 为例，web.xml 是网络程序中的一个很重要的配置文件。

常见 xml 文件头为：

<?xml version="1.0" encoding="UTF-8"?>

网站目录肯定会涉及到多级目录，我们也同样进行模拟。在文件夹中创建一个二级目录“123”，并将一个 web.xml 放入该二级目录中，然后打包成加密 ZIP。



攻击：

```
echo -n '<?xml version="1.0"encoding="UTF-8"?>' > xml_plain
time bkcraack -C xml.zip -c 123/web.xml -p xml_plain -o 0 //
```

注意相对路径

攻击成功：

```
root@labs-a3:~/blus/1027-2# time bkcraack -C xml.zip -c 123/web.xml -p xml_plain -o 0
Generated 4194304 Z values.
[05:34:47] Z reduction using 30 bytes of known plaintext
100.0 % (30 / 30)
254534 values remaining.
[05:34:48] Attack on 254534 Z values at index 7
86.8 % (220866 / 254534)
[05:36:53] Keys
e0be8d5d 70bb3140 7e983fff

real    2m6.274s
user    49m37.895s
sys     0m3.059s
```

解密：

```
bkcraack -C xml.zip -c 123/web.xml -k e0be8d5d 70bb3140
7e983fff -d web.xml
```

### 3.2.7 SVG 文件格式破解

xml 格式的文件除了 .xml 以外，也包括 .svg 文件。SVG 是一种基于 XML 的图像文件格式。

名称	大小	CRC	算法
advice.jpg	54 799	7CA9F10A	ZipCrypto Deflate
spiral.svg	1 265	A99F1D0D	ZipCrypto Store

攻击：

// 已知明文

```
echo -n '<?xml version="1.0" '> plain.txt
```

```
bkcrack -C secrets.zip -c spiral.svg -p plain.txt -o 0
```

攻击成功：

```
root@labs-a3:~/blus/1028# echo -n '<?xml version="1.0" '> plain.txt
root@labs-a3:~/blus/1028# bkcrack -C secrets.zip -c spiral.svg -p plain.txt -o 0
Generated 4194304 Z values.
[23:17:22] Z reduction using 12 bytes of known plaintext
100.0 % (12 / 12)
549421 values remaining.
[23:17:23] Attack on 549421 Z values at index 7
29.2 % (160370 / 549421)
[23:18:54] Keys
c4038591 d5ff449d d3b0c696
```

解密：

// 解密 Store 算法 直接解密即可

```
bkcrack -C secrets.zip -c spiral.svg -k c4038591 d5ff449d
d3b0c696 -d spiral_deciphered.svg
```

// 解密 deflate 算法

```
bkcrack -C secrets.zip -c advice.jpg -k c4038591 d5ff449d
d3b0c696 -d advice.deflate
```

// 该文件使用了 deflate 算法压缩的，解码出来的是 Deflate 的数

据流，因此须将其解压

据流, 因此须将其解压缩。

```
python3 inflate.py <advice.deflate> advice.jpg
```

-----  
-----  
-----

## 四、结尾

### 4.1 结语

以上这些案例只是给打击做个示范, 打开大家的思路, 实际可用的场景有许多。例如一些 CTF 题目压缩包的非预期解, 或者网络上资源的破解。

### 4.2 注意点

已知的明文长度越长, 破解速度越快

图片、文本格式文件、压缩包是最容易以 store 算法被加密打包的

有时会出现攻击得到了密钥, 却无法解密正确文件的情况  
存在 rbkcrack 项目, 增加了部分支持

### 4.3 附件下载

链接:

<https://pan.baidu.com/s/1fuuHYFHSU2e0eIAjO7FLfw>

提取码: t2y4

### 4.4 参考文章

[https://www.aloxaf.com/2018/10/zip\\_crack/](https://www.aloxaf.com/2018/10/zip_crack/)

<https://zhuanlan.zhihu.com/p/129855130>

本文作者: 光通天下无患实验室 BlusKing