记一道明文破解的漫长斗争史

转载

weixin_33824363 于 2018-10-31 17:19:00 发布 219 收藏 1
文章标签： shell

相信很多看过我博客的朋友，都记得我曾经写过 Zip 加密文件破解的文章，如果有没看过的读者朋友们参看这里：https://www.cnblogs.com/ECJTUACM-873284962/p/9387711.html

里面提到了明文攻击的手法，当时我做了一下 15 年强网杯的**爆破?**，我当时通过 ARCHPR 和 pkcrack 两种手段均未成功破解压缩包，上网找相关的 writeup ，未果，我也不知道为啥没成功，时隔三个月，我再次尝试的时候，今早 0:00 的时候，成功地跑出了答案，欲哭无泪～～只能说出题人太过分了，竟然埋下了这个坑。

作为一个目前还在役的 CTF 选手，我会给大家讲解一些有意思的赛题啊，讲解我是如何踩坑的，我又是如何绕过这个坑的。一方面是自我的总结，一方面也是给大家一点学习思路，如何去分析一道赛题，我不会去重复的解读一些基础知识，更多的基础知识请参看 ctf-wiki 上面的内容，目前我是主要负责维护 Web 和 Misc 部分内容。本文也会在后续同步更新到 ctf-wiki 上，当然也欢迎大家一起来 Contribute 。

ctf-wiki 地址：https://ctf-wiki.github.io/ctf-wiki/

本文给大家带来的赛题是来自 15 年强网杯的**爆破?**

赛题地址为：https://static2.ichunqiu.com/icq/resources/ctf/qwb/6005400ffa8ecd5053ab56d0f868d198.zip

测试环境

本题我将会从 Windows 和 Linux 两个系统环境下讲解如何去解决这个问题，我们需要用到的实验环境有：

- Windows 10 家庭版
- ARCHPR 4.53
- Ubuntu 18.04
- pkcrack 1.2.2

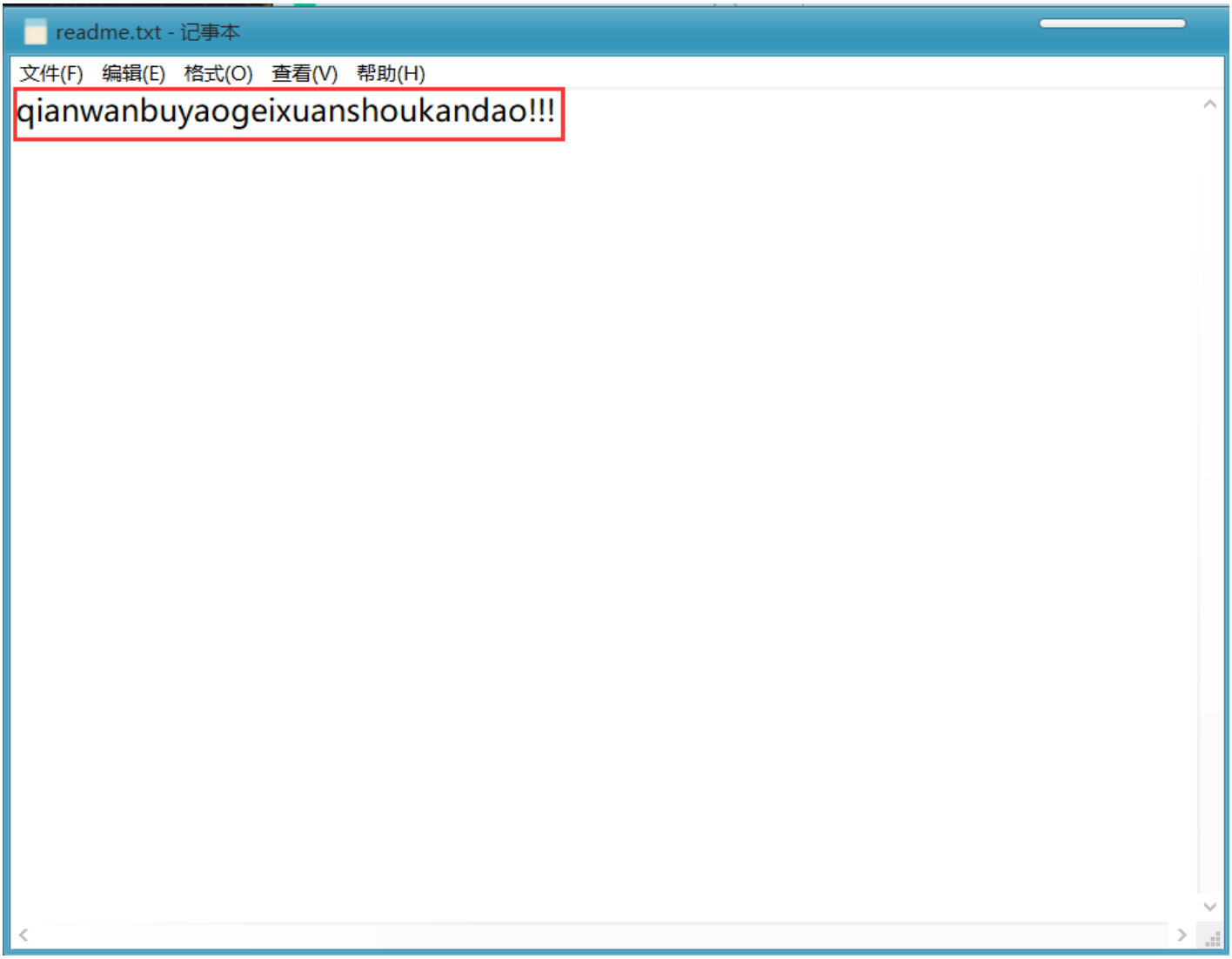
赛题分析

首先我们拿到这样一道题，题目标题为**爆破?**，很明显这题肯定是要用到一个破解工具，很暴力的说。

第一步、分析压缩包文件

我们下载了这个压缩包以后，我们看到文件名是 ***.zip** 结尾，我们可以立即想到破解压缩包常用的几种方式，我在博客上都有写过这些，具体参考原文：https://www.cnblogs.com/ECJTUACM-873284962/p/9387711.html。

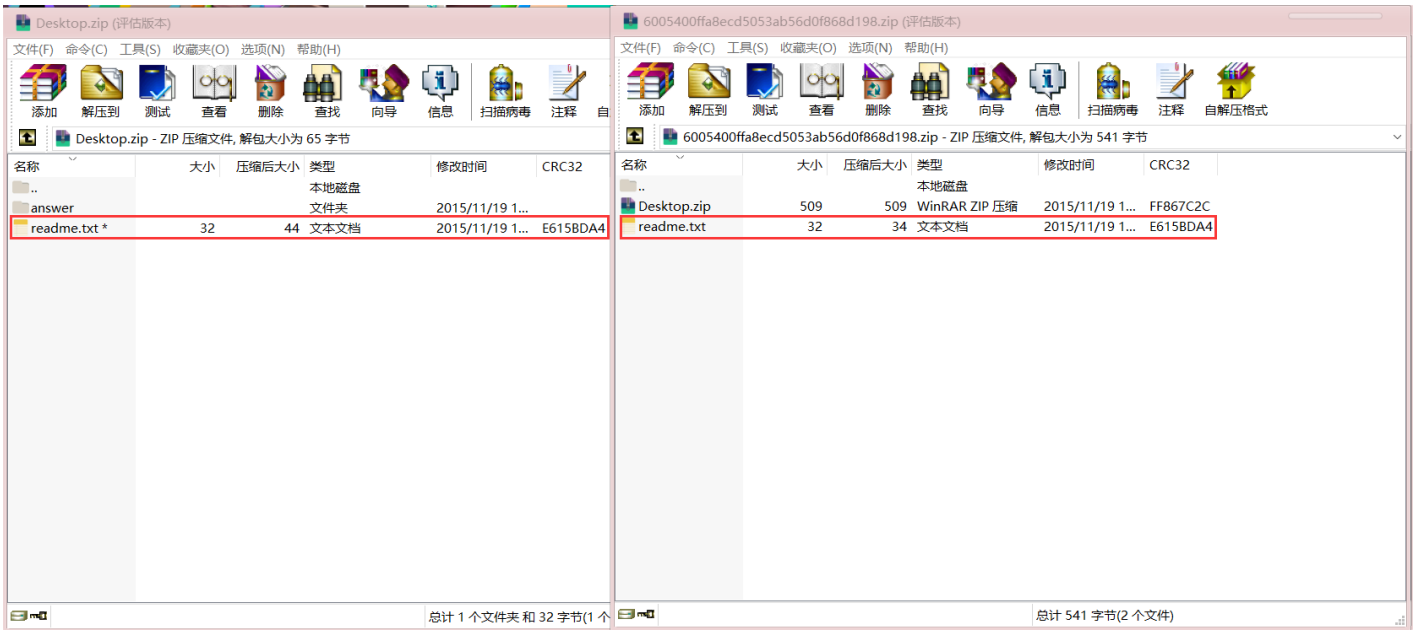
我们将其压缩包解压出来，发现里面有两个文件，分别为 Desktop.zip 和 readme.txt ，我们看看 readme.txt 里面写了什么？



打开以后竟然是qianwanbuyaogeixuanshoukandao!!!，出题人不想让选手看到，这出题人还是有点意思。我们再看看那个 Desktop.zip ，我们可以看到里面有个 readme.txt 文件和 answer 文件夹，answer 文件夹下有 key.txt 文件， flag 应该就藏在这里了。

第二步、分析破解方式

这题目拿到手上，我们首先发现解压出来的文件和Desktop.zip压缩包中都含有同样一个文件 readme.txt ，而且并没有给出其他相关信息，且文件大小大于 12Byte ，我们再对比压缩包中的 readme.txt 和原压缩包中的 readme.txt 的 CRC32 的值，我们发现两个值相同，这说明解压出的 readme.txt 是加密压缩包里的 readme.txt 的明文，于是我们可以大胆地猜测这极可能是个明文加密。



第三步、尝试明文攻击

既然我们已经知道了它是明文攻击的话，我们将对其压缩包进行破解，由于解压出的readme.txt是加密压缩包里的 `readme.txt` 的明文，将 `readme.txt` 压缩成 **.zip** 文件，然后在软件中填入相应的路径即可开始进行明文攻击，这里我们将介绍 `Windows` 和 `Ubuntu` 下使用不同的方式进行明文攻击。

方法一、 pkcrack 进行明文攻击

`pkcrack` 下载链接：https://github.com/flag-porter/CTF-Tools/blob/master/Tools/Crypto/pkcrack/install

我们可以直接写个 `shell` 脚本下载就好了：

```
#!/bin/bash -ex

wget https://www.unix-ag.uni-kl.de/~conrad/krypto/pkcrack/pkcrack-1.2.2.tar.gz
tar xzf pkcrack-1.2.2.tar.gz
cd pkcrack-1.2.2/src
make

mkdir -p ../../bin
cp extract findkey makekey pkcrack zipdecrypt ../../bin
cd ../../
```

把文件保存，改为 `pkcrack-install.sh` ，然后跑到当前目录下，给它加一个执行权限 `x` 。

```
chmod 777 install.sh
```

或者直接可以：

```
chmod u+x install.sh
```

然后运行 `./pkcrack-install.sh`

```
#( 10/30/18@ 7:56下午 )( python@Sakura ):~
   ./pkcrack-install.sh
+ wget https://www.unix-ag.uni-kl.de/~conrad/krypto/pkcrack/pkcrack-1.2.2.tar.gz
--2018-10-30 19:57:04--  https://www.unix-ag.uni-kl.de/~conrad/krypto/pkcrack/pk
crack-1.2.2.tar.gz
正在解析主机 www.unix-ag.uni-kl.de (www.unix-ag.uni-kl.de)... 131.246.124.83, 20
01:638:208:ef34:0:ff:fe00:83
正在连接 www.unix-ag.uni-kl.de (www.unix-ag.uni-kl.de)|131.246.124.83|:443... 已
连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度:  174208 (170K) [application/x-gzip]
正在保存至: “pkcrack-1.2.2.tar.gz”

pkcrack-1.2.2.tar.g 100%[===================>] 170.12K  44.1KB/s    用时 3.9s

2018-10-30 19:57:09 (44.1 KB/s) - 已保存 “pkcrack-1.2.2.tar.gz” [174208/174208])

+ tar xzf pkcrack-1.2.2.tar.gz
+ cd pkcrack-1.2.2/src
+ make
gcc -O6 -Wall   -c -o crc.o crc.c
crc.c:24:13: warning: ‘RCSID’ defined but not used [-Wunused-variable]
 static char RCSID[]="$Id: crc.c,v 1.3 1997/09/18 18:07:24 lucifer Release1_2_1
$";
```

然后当前目录下会生成一个 `bin` 的文件夹，我们直接进入 `bin` 文件夹下，看到有 `pkcrack` 文件，直接对文件进行明文破解。

```
./pkcrack -c "readme.txt" -p readme.txt  -C ~/下载/misc/Desktop.zip -P ~/下载/misc/readme.zip -d ~/decrypt.z
```

我们所用到的参数选项如下：

```
-C:要破解的目标文件(含路径)

-c:破解文件中的明文文件的名字(其路径不包括系统路径,从zip文件一层开始)

-P:压缩后的明文文件

-p:压缩的明文文件中明文文件的名字(也就是readme.txt在readme.zip中的位置)
-d:指定文件名及所在的绝对路径，将解密后的zip文件输出
```

至于其他选项参看 `./pkcrack --help`

解密后的结果如下：

```
#( 10/30/18@ 1:10下午 )( python@Sakura ):~
   cd bin
#( 10/30/18@ 1:10下午 )( python@Sakura ):~/bin
   ls
extract  findkey  makekey  pkcrack  zipdecrypt
#( 10/30/18@ 1:10下午 )( python@Sakura ):~/bin
   ./pkcrack -c "readme.txt" -p readme.txt  -C ~/下载/misc/Desktop.zip -P ~/下载
/misc/readme.zip -d ~/decrypt.zip
Files read. Starting stage 1 on Tue Oct 30 13:12:37 2018
Generating 1st generation of possible key2_43 values...done.
Found 4194304 possible key2-values.
Now we're trying to reduce these...
Done. Left with 342485 possible Values. bestOffset is 24.
Stage 1 completed. Starting stage 2 on Tue Oct 30 13:12:40 2018
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
```

```
Ta-daaaaa! key0=df96dc88, key1=b432ddfd, key2=df4b9e93
Probabilistic test succeeded for 24 bytes.
Ta-daaaaa! key0=df96dc88, key1=b432ddfd, key2=df4b9e93
Probabilistic test succeeded for 24 bytes.
Ta-daaaaa! key0=df96dc88, key1=b432ddfd, key2=df4b9e93
Probabilistic test succeeded for 24 bytes.
Ta-daaaaa! key0=df96dc88, key1=b432ddfd, key2=df4b9e93
Probabilistic test succeeded for 24 bytes.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
```

```
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Stage 2 completed. Starting zipdecrypt on Tue Oct 30 15:27:40 2018
Decrypting answer/key.txt (93cc3c98bac382a9443cc4e7)... OK!
Decrypting readme.txt (03cb26263c718c55931e15e6)... OK!
Finished on Tue Oct 30 15:27:40 2018
#( 10/30/18@ 3:27下午 )( python@Sakura ):~/bin
```

```
#( 10/30/18@ 3:44下午 )( python@Sakura ):~
  ls
 bin                          pkcrack-1.2.2
 decrypt.zip                  pkcrack-1.2.2.tar.gz
 examples.desktop             pkcrack-install.sh
 get-pip.py                   sources.list.backup
 imooc-cookie.txt            'VirtualBox VMs'
 p7zip_9.20.1                 zsh-syntax-highlighting
 p7zip_9.20.1_src_all.tar.bz2 公共的
#( 10/30/18@ 3:44下午 )( python@Sakura ):~
  unzip decrypt.zip
Archive:  decrypt.zip
   creating: answer/
 extracting: answer/key.txt
 extracting: readme.txt
#( 10/30/18@ 3:45下午 )( python@Sakura ):~
  cat ./answer/key.txt
flag{7ip_Fi13_S0m3tim3s_s0_3a5y@}%
#( 10/30/18@ 3:45下午 )( python@Sakura ):~
```

我们可以看到，我们下午 1:10 开始跑的，下午 3:27 才求解出秘钥。

我们得出了最终的flag为：**flag{7ip_Fi13_S0m3tim3s_s0_3a5y@}**

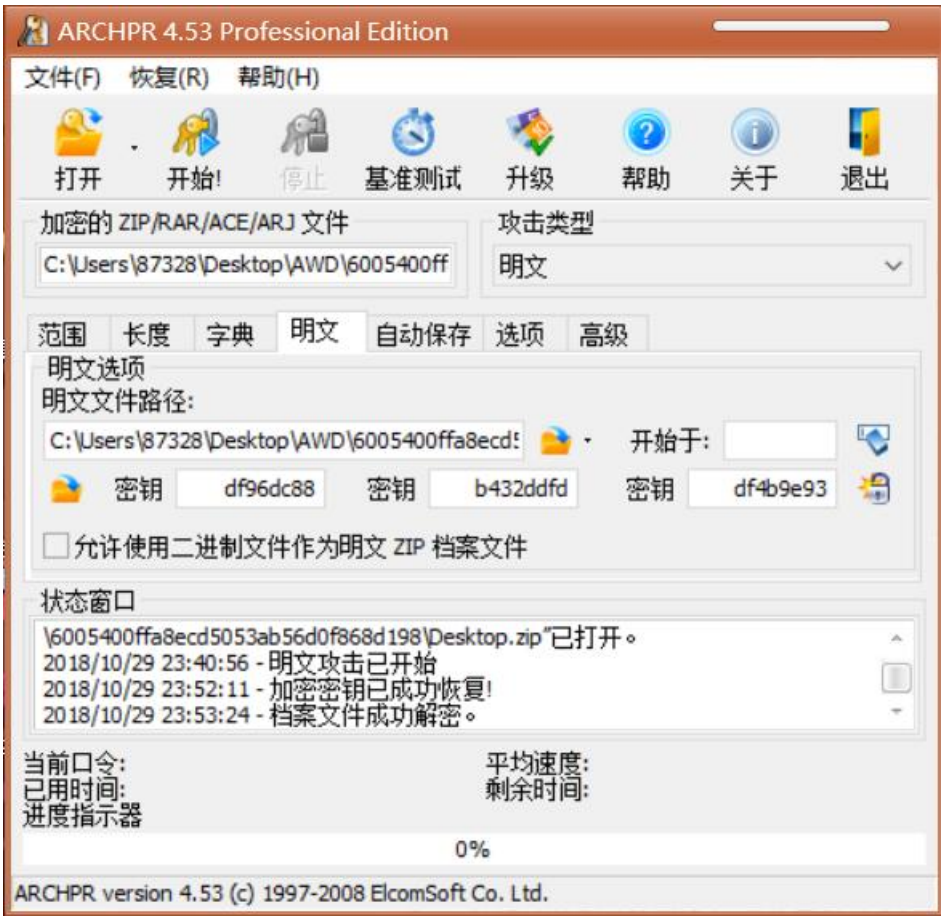
坑点来了

看起来一切都很顺利的样子，同样花了两个多小时，为啥我在博客园上写了我跑了两个小时都没跑出来呢？或者说有朋友遇到了和我一样的问题，我明明和你是一样的，为啥我跑不出结果？

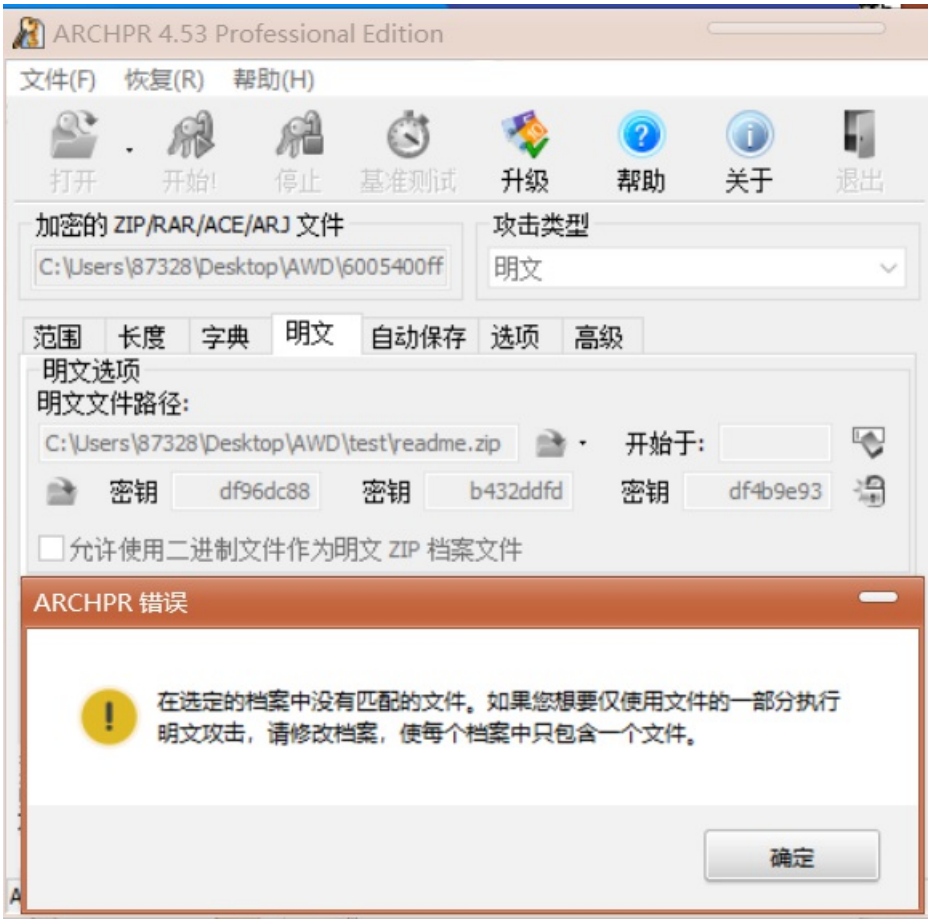
你们可能忽略了一些细节问题，有人曾想过原压缩包是通过什么方式压缩的嘛？还有就是我们生成的 readme.zip 又该以哪种方式去生成呢？我就是因为这个问题卡了整整三个月没做出来，不信的话我们可以看看第二种方法，在 Windows 下用 ARCHPR 进行明文攻击。

方法二、ARCHPR进行明文攻击

首先这题我建议大家下 ARCHPR 4.53 版本，我是在这个版本下测试成功的。成功截图如下：



我相信很多朋友在用 ARCHPR 的时候遇到以下这种情况：



我当时内心是崩溃的，为啥会出现这种情况。

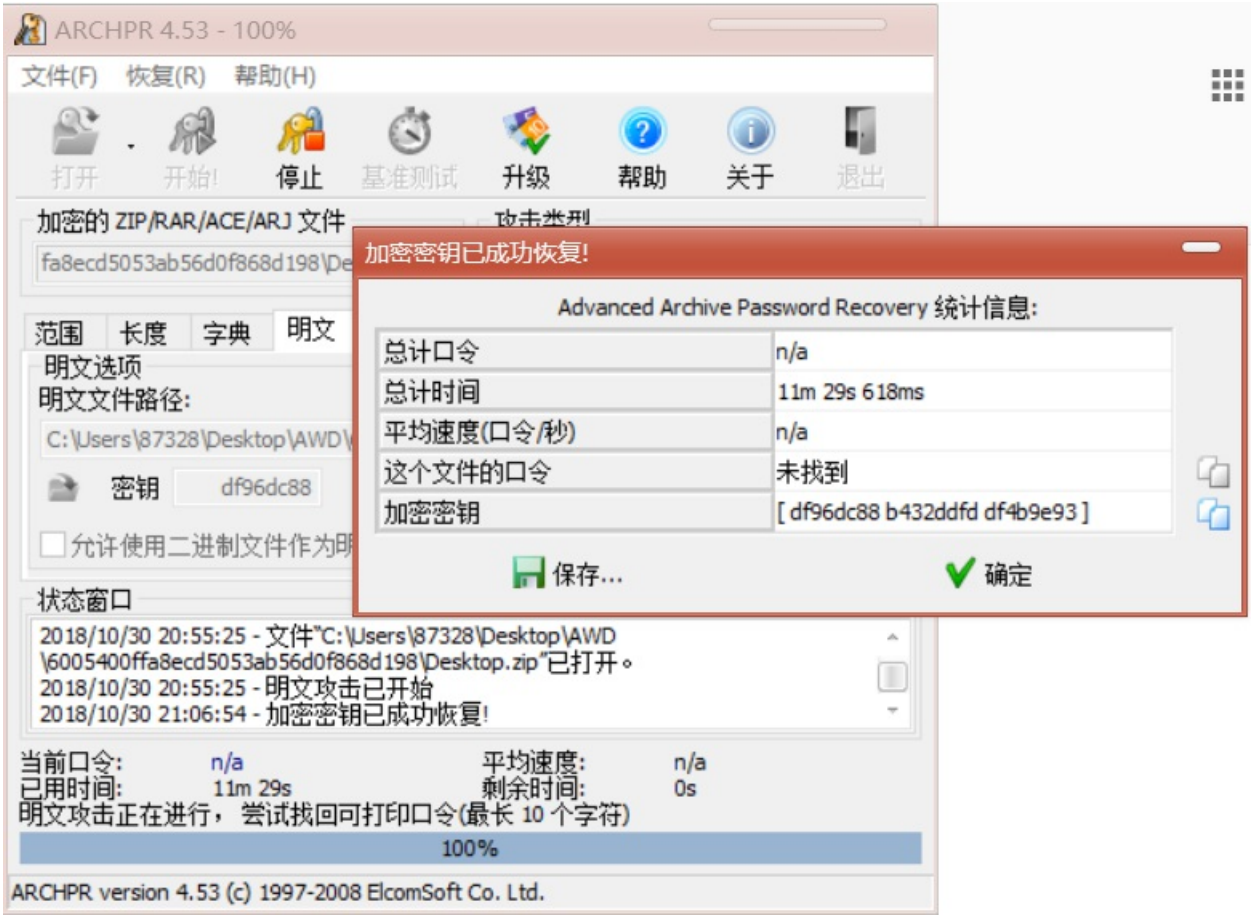
在后来的学习中发现，用 7z 压缩的文件得用 7z 来解压缩， 7z 是一种使用多种压缩算法进行数据压缩的档案格式，和传统的 zip， rar 相比，它的压缩比率更大，采用的压缩算法不同，自然而然就可能出现不匹配这种情况，所以我们在解压缩原压缩包和对文件进行加密的时候得先分析出题人是用什么方式进行加解密的，所以这题的问题显而易见就出来了，经过验证，我发现出题人是用 7z 进行压缩的。

再尝试

我们已经发现了这个问题，我们去官网下载 7zip ：https://www.7-zip.org/

然后我们对原压缩文件用 7z 进行解压缩，然后将 readme.txt 用7z进行压缩即可。然后我们就可以用 ARCHPR 进行明文攻击了。

结果如下：



我们将 Desktop_decrypted.zip 解压出来，查看 answer 目录下的 key.txt 即可。

所以最终的flag为：**flag{7ip_Fi13_S0m3tim3s_s0_3a5y@}**