ctf misc 压缩包+音频知识点

原创

z.volcano 于 2021-10-04 21:21:42 发布 2451 收藏 13

分类专栏： # 笔记 文章标签： misc

本文链接：https://blog.csdn.net/weixin_45696568/article/details/118573215

笔记 专栏收录该内容

3 篇文章 0 订阅

订阅专栏

最近不想学习，放点以前写的笔记…

.

压缩文件题

拿到一个文件先用010editor或是winhex打开，看一下文件头，确定文件类型，然后修改后缀。如果一个压缩文件打不开，看看**文件头文件尾有没有问题**，有些时候会用rar的文件头配zip的文件尾，需要自行修复。

打开压缩文件后，如果有`注释信息`，一定要看，往往会给出压缩包密码的提示或是后面解题的提示。

CRC32爆破

工具地址：https://github.com/theonlypwner/crc32

原理

> 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

例题：bugku的**奇怪的png图片**，一般是加密的txt文件，而且**字节数小于等于6**，往往是爆破不出密码的，因为考察点在crc32

..			文件夹		
图片真实crc32.t...	6	20	文本文档	2021/1/10 21:...	59F1D4BE
pass3.txt *	4	16	文本文档	2021/1/10 20:...	694AEF57
pass2.txt *	4	16	文本文档	2021/1/10 20:...	E3A35EED
pass1.txt *	4	16	文本文档	2021/1/10 20:...	1D9F11E5
flag.txt *	27	41	文本文档	2021/1/10 21:...	2B54F20B

比如要爆破选中的这个文档，就执行

```
crc32-master>python3 crc32.py reverse 0x59f1d4be
4 bytes: {0xde, 0x9f, 0xfd, 0x06}
verification checksum: 0x59f1d4be (OK)
alternative: 2KHAZK (OK)
alternative: 9070yo (OK)
alternative: FRzUbd (OK)
alternative: GNt8xi (OK)
alternative: I1WyA7 (OK)
alternative: NXl6nL (OK)
alternative: PzWHOM (OK)
alternative: WcPvef (OK)
alternative: YlOFYh (OK)
alternative: aDNRsJ (OK)
alternative: c5fqG_ (OK)
alternative: oKQbOD (OK)
alternative: pudqtH (OK)
alternative: uq8An2 (OK)
```

根据题目选择最接近的即可。

伪加密

分为rar伪加密和zip伪加密，前者很少考察，特点是打开压缩文件时会报错，不过如果使用360压缩，会直接无视。

`zip伪加密`的原理可以参考这篇文章，破解方式一般有两种，第一种是根据文章里的步骤，把被修改过的加密位恢复，我更喜欢用**ZipCenOp.jar**工具进行破解。

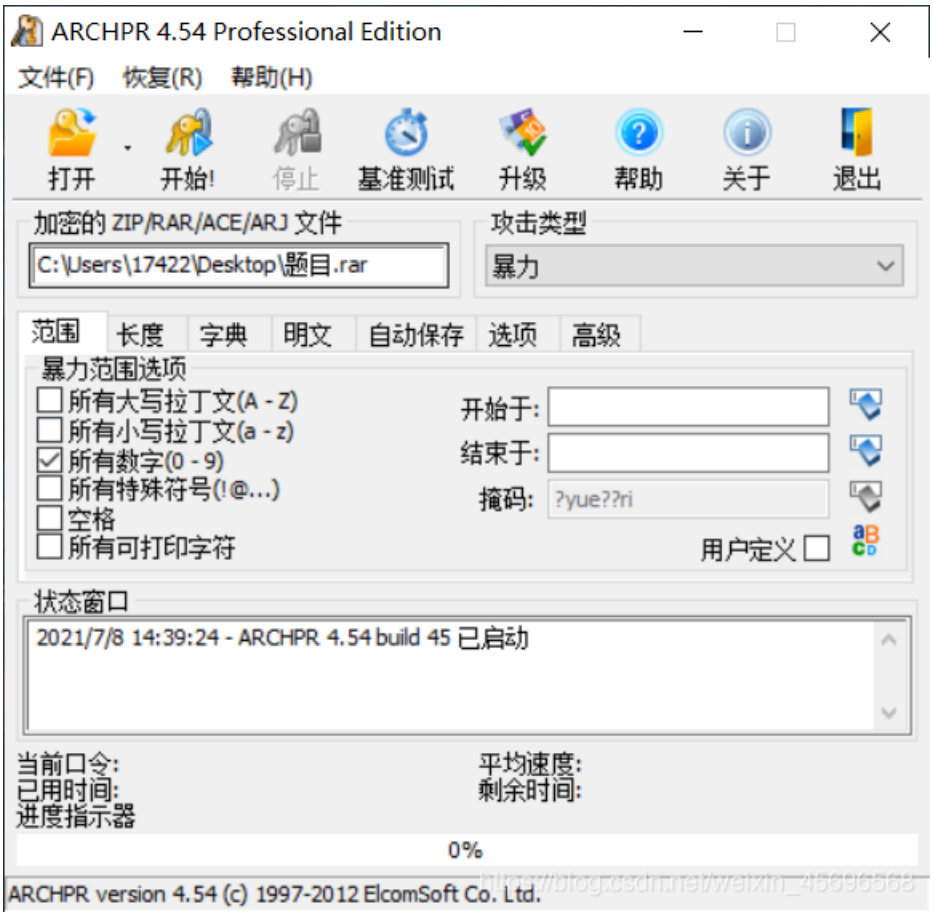
常用工具及其功能

一般使用`ARCHPR`或是`Ziperello`，前者功能多一点，这里介绍前者。

暴力破解

普通的暴力破解，自行选择范围(密码的组成)和密码长度，同时可以规定从哪里开始，到哪里截止，缩短爆破时间

除此之外还有**掩码爆破**和**字典爆破**，稍微看看就能上手了



明文攻击

想要实现明文攻击，有几个条件

> 1、完整的明文文件，这个明文文件通常会是一个txt文本，偶尔是别的格式，比如校赛那题用的就是jpg格式。如果是一个txt，注意编码方式，这个很重要。
> 2、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），有时候出题人会给出压缩方式的提示，不过即便没给，多试几次也能试出来。具体的判定方法就是，我们压缩得到的文件的crc32校验值和题目中对应的文件的crc32值是否相同。

满足条件就可以开始攻击了，注意一点，`不需要等它攻击完成`

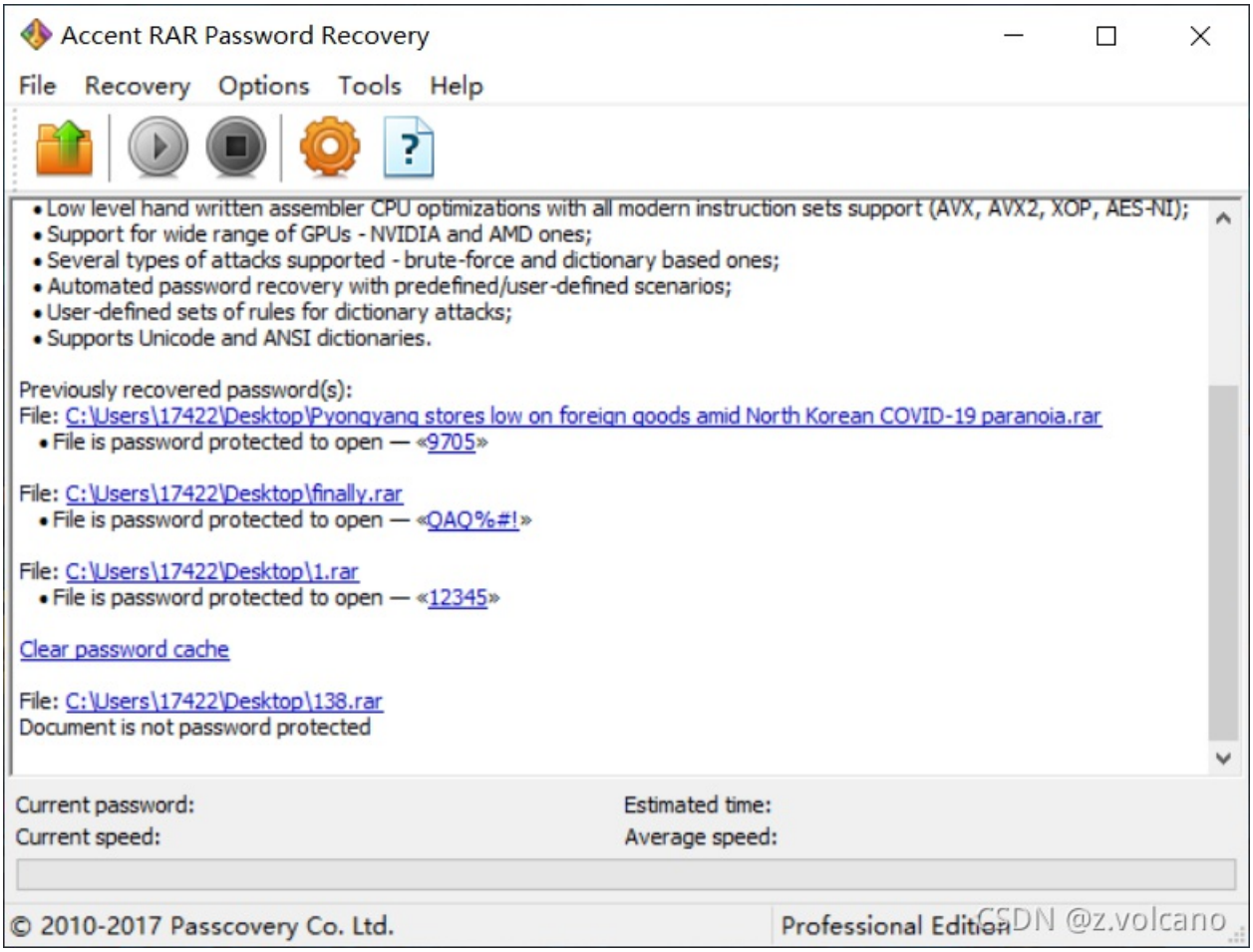
这里先是会先是显示`正在搜索秘钥`，然后会尝试`找回口令`，如果密码非常复杂，找回口令的这个过程可能会持续好几个小时，甚至是几天，所以不需要等它，直接停止。

随后会发现`秘钥已经恢复`，也就是说原压缩包已经破解成功，点击确定，然后保存解密后的压缩包即可。

关于前面提到的编码问题和加密方式的问题，可以参考这篇博客的`[*CTF2019]otaku`

rar5.0加密

上面说的两种工具没办法破解rar5.0的加密，不过可以用破解版的`Accent RAR Password Recovery`进行爆破，这个工具也支持暴力破解、掩码爆破、字典爆破。



或者先用john获取rar文件的hash值，再用`hashcat`爆破

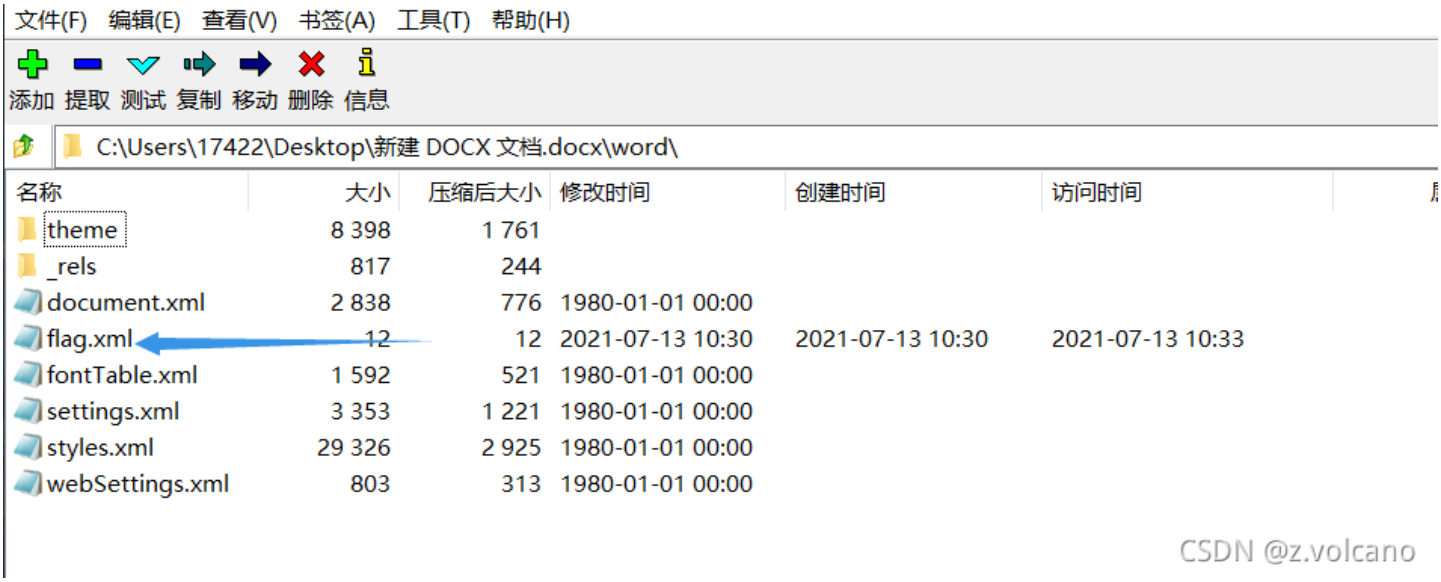
其他格式的压缩文件

常见的压缩文件格式有rar、zip和7z。

除此之外，还有一些常见的格式也是压缩文件，比如**docx**、**xls**、**ppt**、等，如果直接在这些文件中找不到flag，可以考虑**把后缀改成zip**，然后翻找文件。

例如，在一个word里面只有如下一句话，除此之外没有别的东西。

修改后缀为zip或是直接以压缩包打开，会发现flag，也可能是在其他位置，这里只是举个例子。



音频题

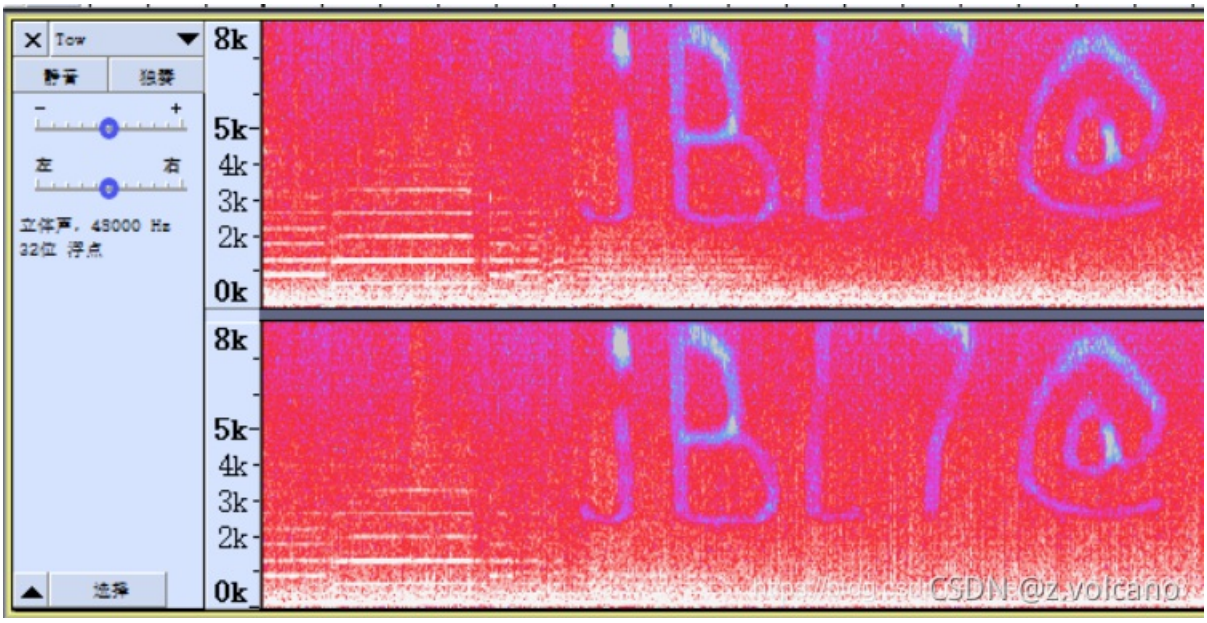
有时候会直接把flag或者关键信息塞进文件的字节信息里，可以用010或者winhex查看，更多时候会把信息隐藏在`波形`或者`频谱`中，可以使用`Audacity`工具来查看。

Audacity

直接看

有时候直接打开就能看到**flag**或者**关键信息，波形图和频谱图记得都看一下**，记得`放大后仔细翻看`，不要低估出题人狗的程度

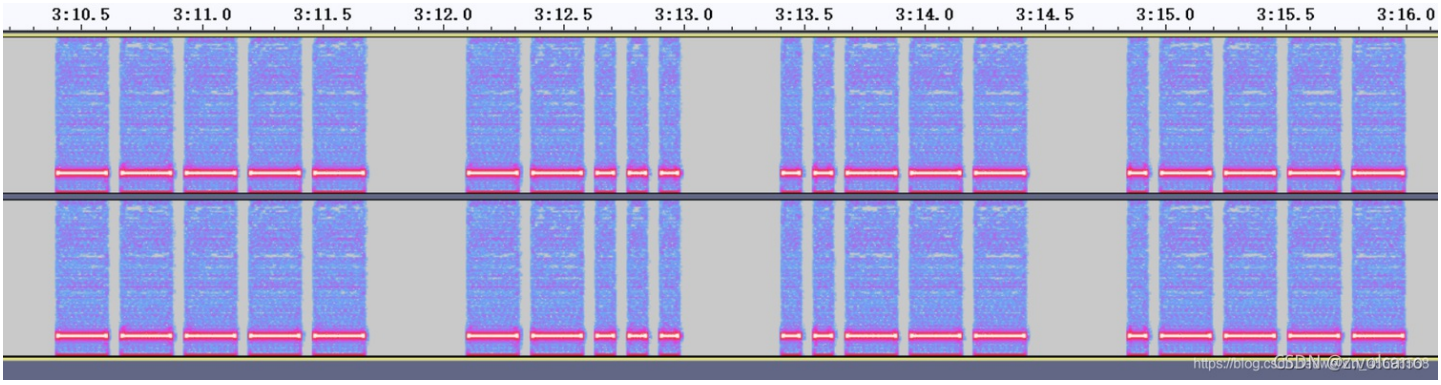
下图就是频谱图中直接看到信息



摩斯密码解密

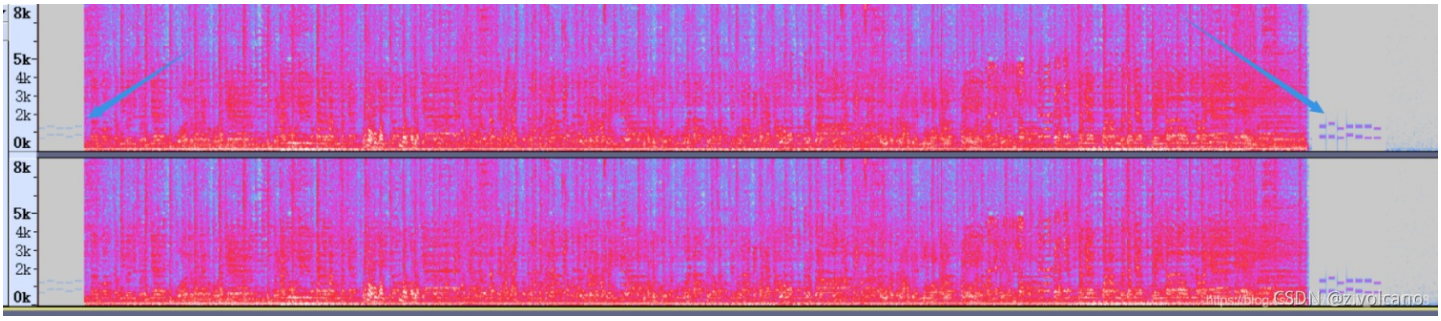
很常见的一个考察点，这里以频谱图为例(波形图中也能看到，不过**频谱图往往更清楚点**)，`长一点的和短一点分别换成-和.`(一般情况下)，然后拿去解密，下图解密结果是`0721`

一般这种信息都会藏在音频的头部或者尾部



拨号音隐写

有的音频文件，直接听是可以听到拨号音的，有能力的师傅可以直接听出对应结果，像我就只能放进Audacity里分析，查看频谱图



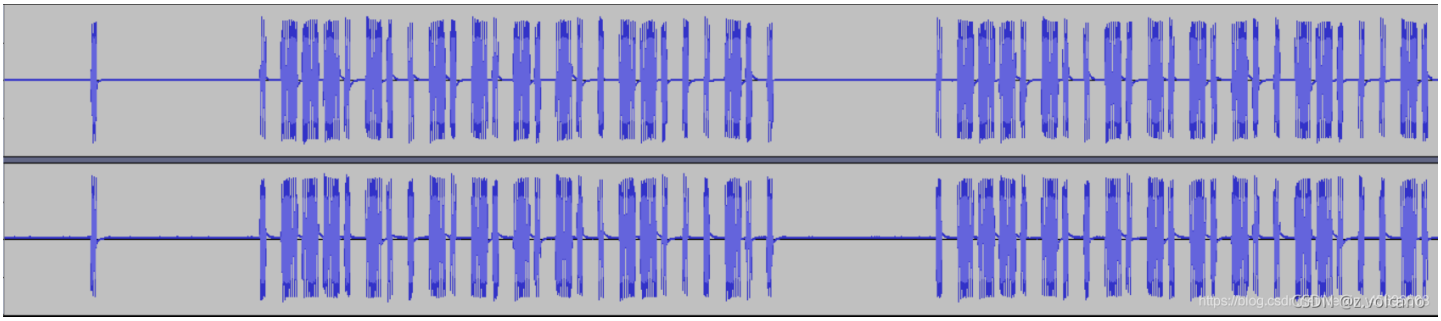
上图是一个例题，一般就是用脚本撸或者对照表格看。

双音多频键盘

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

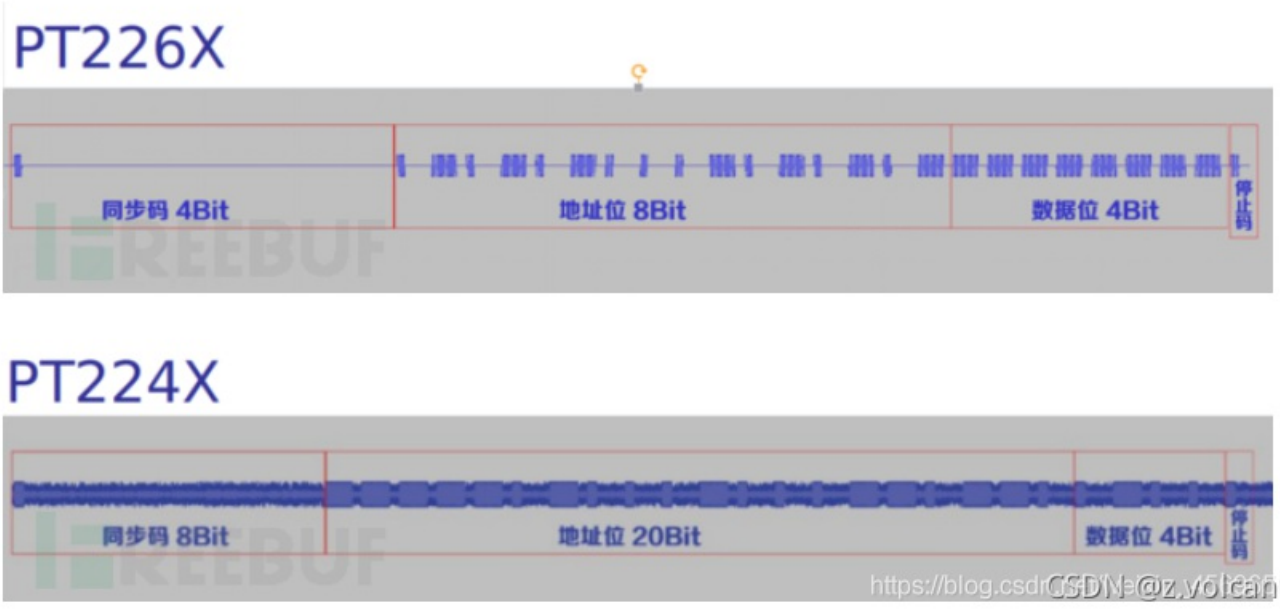
固定码遥控信号

直接看波形，发现宽窄不一，有点像前面的摩斯密码，但其实还是差挺多的



对照表格解密即可(这里给的不全)

固定码遥控信号的构成



sstv

几个月前遇到的一个新奇的东西，可以把图片隐藏到音频信息中

这里解密的话要用到一个工具`QSSTV`，在kali下安装：`apt install qsstv`
关于qsstv的具体使用参考这篇文章

音频隐写

这里只介绍我见过的几种

LSB 音频隐写

类似于图片隐写中的 LSB 隐写，主要使用 `Silenteye` 工具解密。

例题：**2015 广东省强网杯 - Little Apple**

DeepSound

自行使用搜索引擎下载，例题可参考`[INSHack2018] (not) so deep`

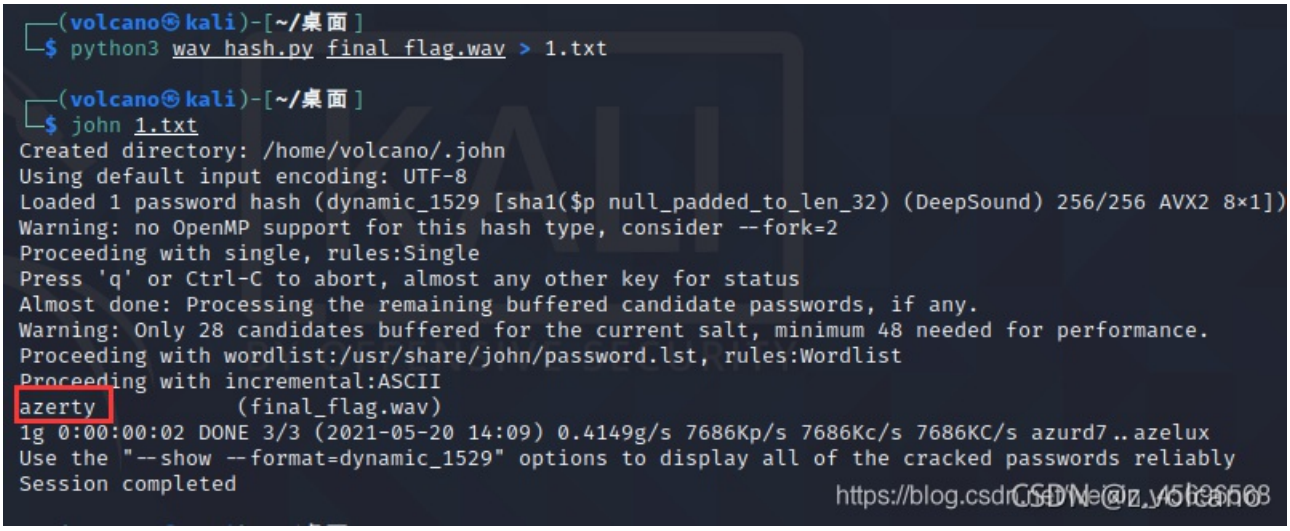
使用时**可能会需要密码**，先用脚本获取**hash值**

```
#!/usr/bin/env python3
'''
deepsound2john extracts password hashes from audio files containing encrypted
data steganographically embedded by DeepSound (http://jpinsoft.net/deepsound/).
This method is known to work with files created by DeepSound 2.0.
Input files should be in .wav format. Hashes can be recovered from audio files
even after conversion from other formats, e.g.,
    ffmpeg -i input output.wav
Usage:
    python3 deepsound2john.py carrier.wav > hashes.txt
    john hashes.txt
This software is copyright (c) 2018 Ryan Govostes <rgovostes@gmail.com>, and
it is hereby released to the general public under the following terms:
Redistribution and use in source and binary forms, with or without
modification, are permitted.
'''

import logging
import os
import sys
import textwrap
```

```
def decode_data_low(buf):
  return buf[::2]

def decode_data_normal(buf):
  out = bytearray()
  for i in range(0, len(buf), 4):
    out.append((buf[i] & 15) << 4 | (buf[i + 2] & 15))
  return out

def decode_data_high(buf):
  out = bytearray()
  for i in range(0, len(buf), 8):
    out.append((buf[i] & 3) << 6     | (buf[i + 2] & 3) << 4 \
             | (buf[i + 4] & 3) << 2 | (buf[i + 6] & 3))
  return out


def is_magic(buf):
  # This is a more efficient way of testing for the `DSCF` magic header without
  # decoding the whole buffer
  return (buf[0] & 15)  == (68 >> 4) and (buf[2]  & 15) == (68 & 15) \
     and (buf[4] & 15)  == (83 >> 4) and (buf[6]  & 15) == (83 & 15) \
     and (buf[8] & 15)  == (67 >> 4) and (buf[10] & 15) == (67 & 15) \
     and (buf[12] & 15) == (70 >> 4) and (buf[14] & 15) == (70 & 15)


def is_wave(buf):
  return buf[0:4] == b'RIFF' and buf[8:12] == b'WAVE'


def process_deepsound_file(f):
  bname = os.path.basename(f.name)
  logger = logging.getLogger(bname)

  # Check if it's a .wav file
  buf = f.read(12)
  if not is_wave(buf):
    global convert_warn
    logger.error('file not in .wav format')
    convert_warn = True
    return
  f.seek(0, os.SEEK_SET)

  # Scan for the marker...
  hdrsz = 104
  hdr = None

  while True:
    off = f.tell()
    buf = f.read(hdrsz)
    if len(buf) < hdrsz: break

    if is_magic(buf):
          hdr = decode_data_normal(buf)
          logger.info('found DeepSound header at offset %i', off)
          break

    f.seek(-hdrsz + 1, os.SEEK_CUR)
```

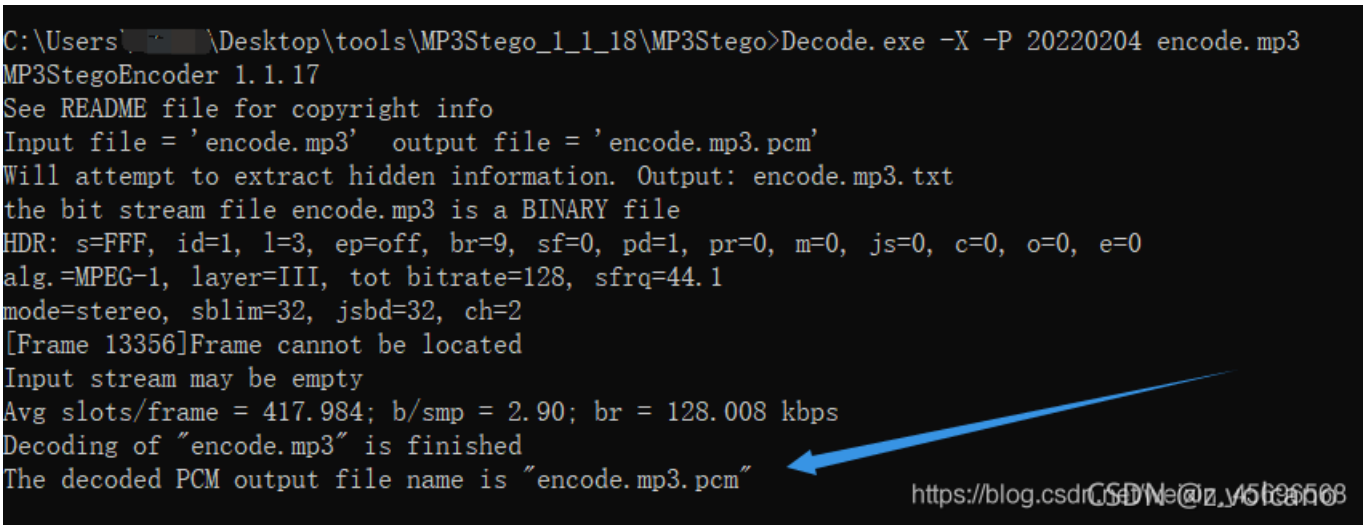
```
  if hdr is None:
    logger.warn('does not appear to be a DeepSound file')
    return

  # Check some header fields
  mode = hdr[4]
  encrypted = hdr[5]

  modes = {2: 'low', 4: 'normal', 8: 'high'}
  if mode in modes:
    logger.info('data is encoded in %s-quality mode', modes[mode])
  else:
    logger.error('unexpected data encoding mode %i', modes[mode])
    return

  if encrypted == 0:
    logger.warn('file is not encrypted')
    return
  elif encrypted != 1:
    logger.error('unexpected encryption flag %i', encrypted)
    return

  sha1 = hdr[6:6+20]
  print('%s:$dynamic_1529$%s' % (bname, sha1.hex()))


if __name__ == '__main__':
  import argparse

  parser = argparse.ArgumentParser()
  parser.add_argument('--verbose', '-v', action='store_true')
  parser.add_argument('files', nargs='+', metavar='file',
    type=argparse.FileType('rb', bufsize=4096))
  args = parser.parse_args()

  if args.verbose:
    logging.basicConfig(level=logging.INFO)
  else:
    logging.basicConfig(level=logging.WARN)

  convert_warn = False

  for f in args.files:
    process_deepsound_file(f)

  if convert_warn:
    print(textwrap.dedent('''
    ---------------------------------------------------------------
    Some files were not in .wav format. Try converting them to .wav
    and try again. You can use: ffmpeg -i input output.wav
    ---------------------------------------------------------------
    '''.rstrip()), file=sys.stderr)
```

然后用john爆破密码，拿到密码后即可解密



MP3Stego

比较常见，安装步骤自行百度，解密命令如下：`Decode.exe -X -P 密码 文件路径`



其余的想起来再补充...