攻防世界Misc_高手进阶区第二页WriteUp

原创

D-R0s1 于 2019-09-23 10:31:47 发布 2267 收藏 3

分类专栏： CTF WriteUp 杂项

本文链接：https://blog.csdn.net/CliffordR/article/details/101195924


CTF WriteUp 同时被 2 个专栏收录

28 篇文章 3 订阅

订阅专栏


杂项

7 篇文章 0 订阅

订阅专栏

文章目录

0x00 写这篇博客的目的

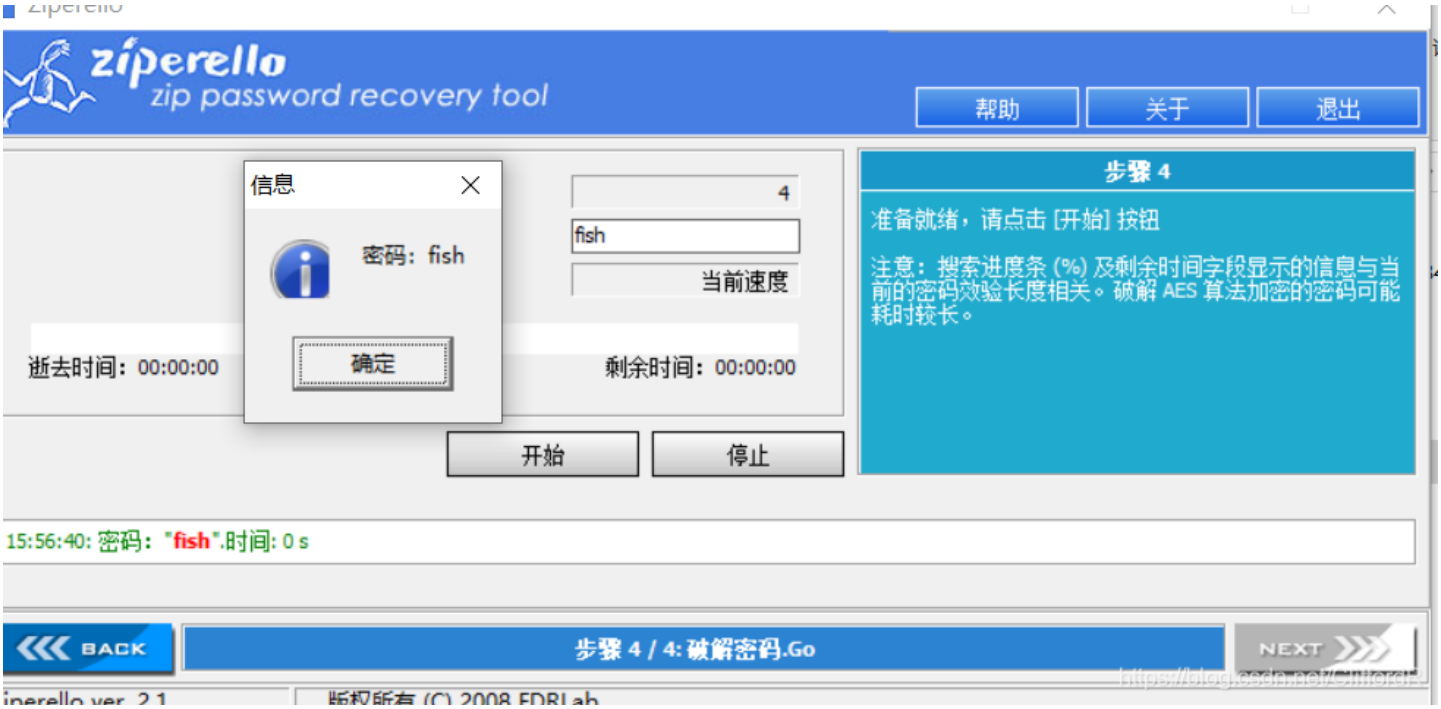
对于CTF中的Misc来说，做题经验显得十分重要，而做题经验的获得很大一部分取决于刷题量。为了避免大家在刷题过程到处搜WriteUp浪费时间，现在把我的一些做题方法分享出来，希望对大家有帮助。当然，大家有更好的解决方法欢迎在评论区留言，互相学习，共同进步。

0x01 解题步骤

1.János-the-Ripper

a.拿到手是一个没有后缀的文件，常规操作先扔到kali中识别一下文件的类型，
一般来说，大佬们习惯放进winhex中看文件头，我8大星，我就放进winhex中。
b.放进kali中得知这是一个压缩包，解压
c.解压过程中发现图压缩包存在密码，无法正常解压出来，首先试一下是不是伪加密，伪加密的解决方法有好多，包括在winhex改十六进制等等的一些方法，而我在这里推荐一种比较简单的方法，在winrar中使用自动修复即可修复伪加密。

修复以后还是需要密码，那么也就是说不存在伪加密的这个现象
d.接下来爆破一下密码



比较轻松的就把密码爆破出来了

e.然后解压出里面的文件得到flag

2.2017_Dating_in_Singapore

a.由题目和一大串字符想到是类似日历的东西
b.找了份17年的日历根据所给的字符可以描出flag
c.出题人脑洞是真的大

3. 神奇的Modbus

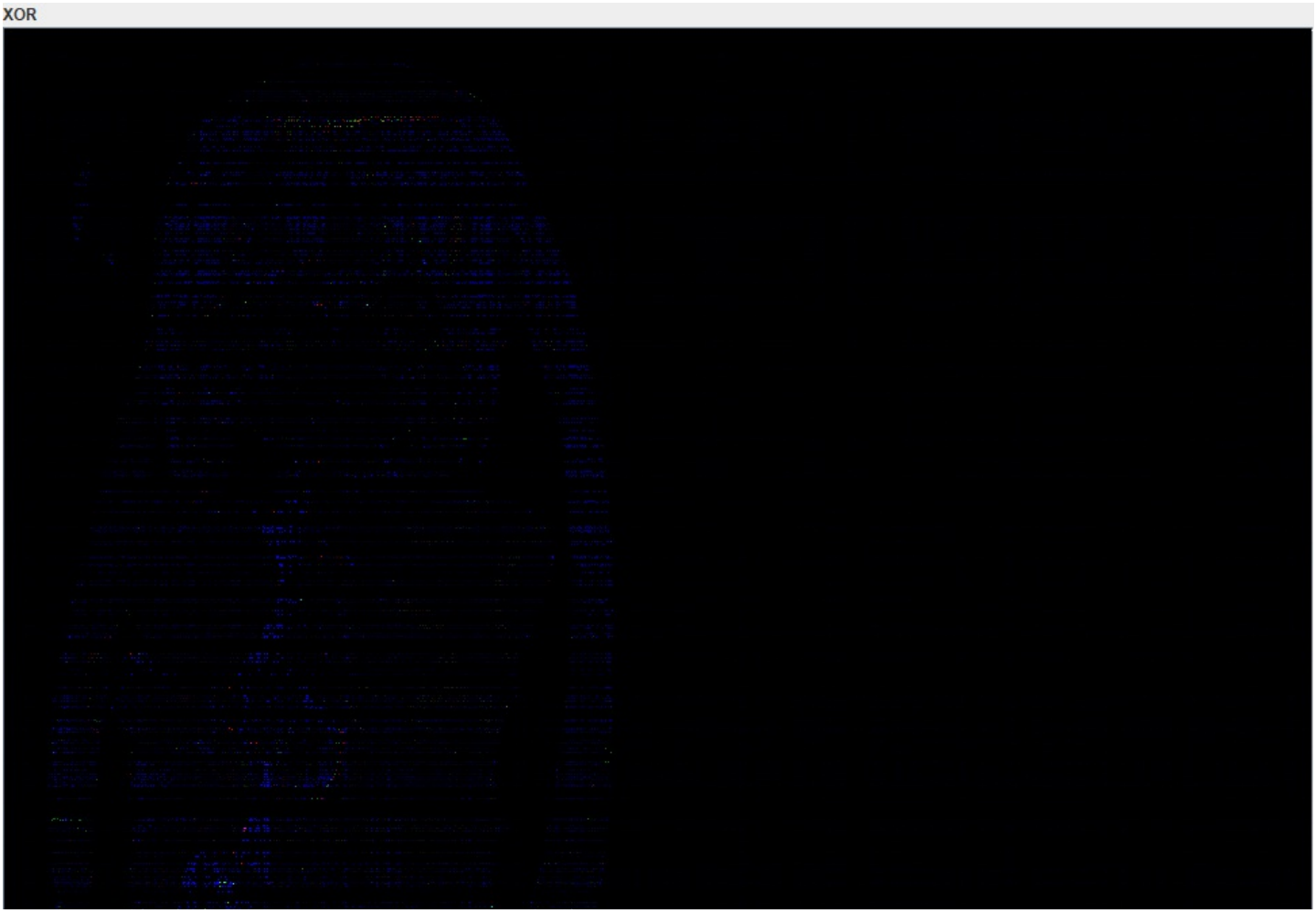
a.拿到手是一个流量包，根据提示是modbus协议
b.过滤一下存在modbus的协议
c.追踪一下tcp流发现flag

```
.....................".............................#.. .c.t.f.
{.E.a.s.y._.M.d.b.u.s.}.....................................
3.....................................`......................_.....
1.....................................].....................
```

d.得到了这个falg之后呢，一直提交不上，但是我在网上查找其他的writeup也是这么说的，有解决的师傅评论区指点我一下。

4. 4-1

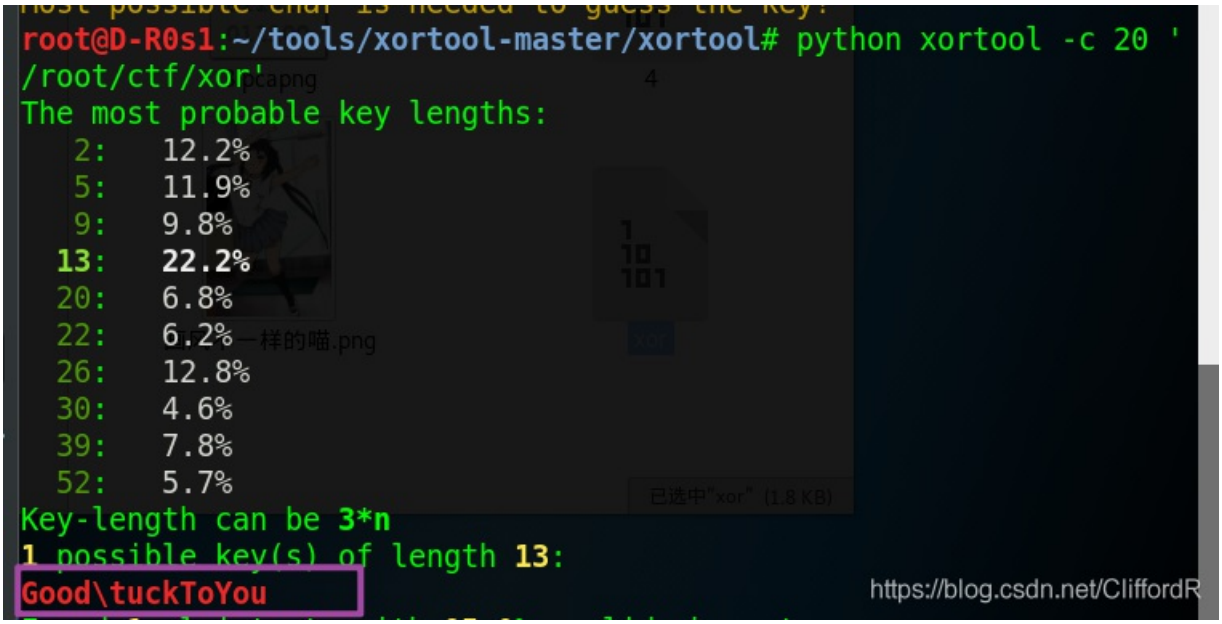
a.拿到手是一个图片，首先放入winhex中走个形式查找一下flag，并且看一下有没有别的可疑的地方
b.虽然没有查找到flag但是从winhex中可以看到图片中是包含其他文件的，放入kali中foremost分离一下
c.分离出来其中存在一个zip文件，解压一下，最终得到两张照片。
d.两张照片的话首先想到在stegsolve中比较一下或者是盲水印攻击
e.在stegsolve中打开比较一下



发现有条纹状的东西，怀疑是盲水印攻击
f.在这里介绍一下盲水印攻击的使用方法
提取图片中的盲水印：
python bwm.py decode 无水印图片 有水印图片 提取出的图片
合成盲水印图片：
python bwm.py encode 无水印图片 水印 有水印图片
根据提示图片2应该是包含水印的图片，提取一下得到结果

5. 5-1

a.拿到文件没有后缀，把它放入kali中是一个二进制文件，在winhex中和binwalk都没有什么收获。
b.这其实是一个需要异或的文件
c.使用xortool异或得到key



尝试出了key：GoodLuckToYou，对原文件进行异或，脚本如下：

```
key = 'GoodLuckToYou'
flag = ''
with open('cipher') as f:
   con = f.read()
   for i in range(len(con)):
       flag += chr(ord(con[i]) ^ ord(key[i%13]))
f = open('flag.txt', 'w')
f.write(flag)
f.close()
```

注意脚本运行环境是在python2中

```
with Wickham, and the results are
s. wdflag{You Are Very Smart}Though
s, Elizabeth and Darcy, begin the
```

6.MISCall

a.文件下载下来没有后缀，放进kali中得知是一个压缩包文件，加后缀名为zip然后解压
b.同样的方法解压后发现是一个ctf文件，有一个txt文件但里面没有flag，但是看见一个.git文件
c.git log(查看git记录)

```
root@D-R0s1:~/ctf# git log
commit bea99b953bef6cc2f98ab59b10822bc42afe5abc (
HEAD -> master)
Author: Linus Torvalds <torvalds@klaava.Helsinki.
Fi>
Date:   Thu Jul 24 21:16:59 2014 +0200

    Initial commit
```

d.git stash list（查看修改列表）

```
stash@{0}: WIP on master: bea99b9 Initial commit
```

存储列表应该是有东西的
e.git stash show(校验一下列表中的存储文件)

```
root@D-R0s1:~/ctf# git stash show
 flag.txt | 25 ++++++++++++++++++++++++-
 s.py     |  4 ++++
 2 files changed, 28 insertions(+), 1 deletion(-)
```

f.git stash apply(重新储存，把上面的文件复原)

这样就得到了flag.txt和s.py，运行s.py即得flag。

7、can_has_stdio?

a.文件下载后解压是一个无后缀文件，先随手用notepad++打开
b.明显的brainfuck编码

解码方法可以在线解码也可以使用工具解码

c.补充一下brainfuck的编码知识

BrainFuck语言

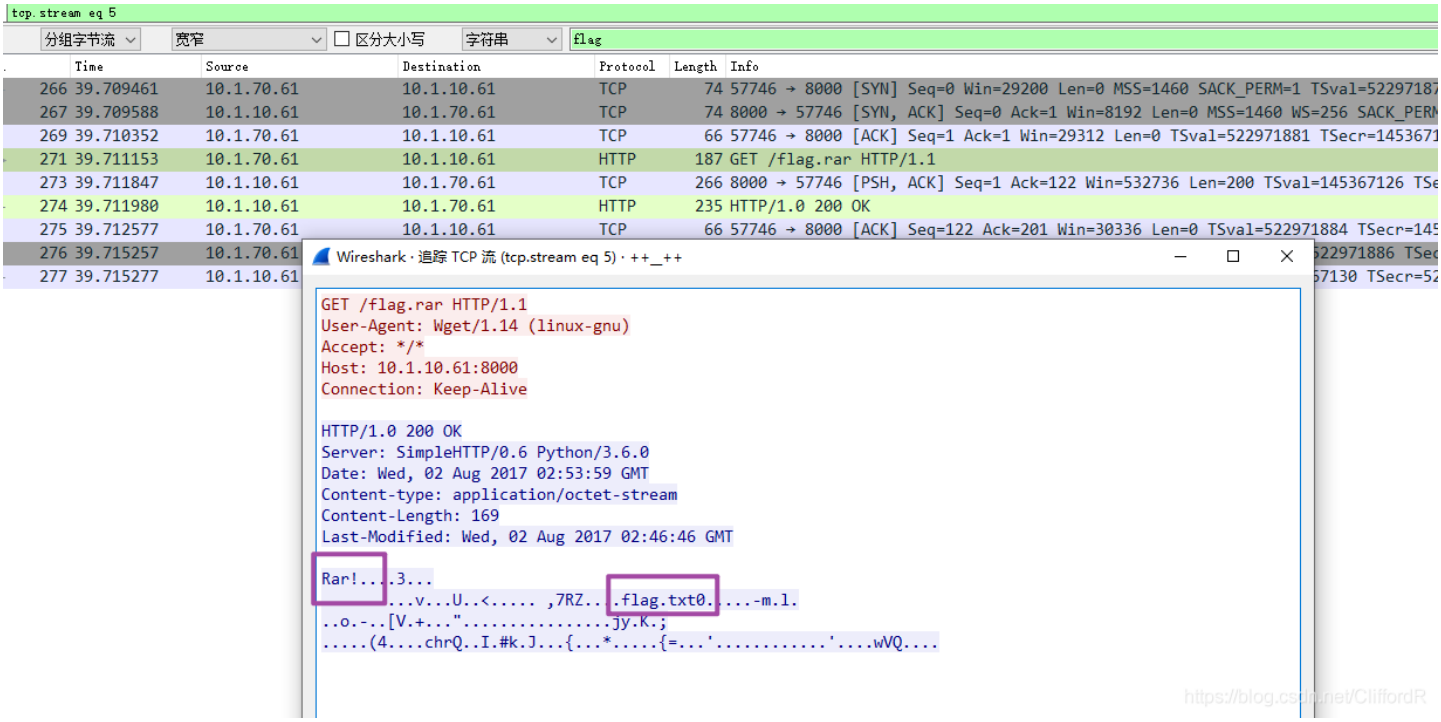
极简的一种图灵完备的语言，由Urban Müller在1993年创造，由八个指令组成（如下表）。工作机制与图灵机非常相似，有一条足够长的纸带，初始时纸带上的每一格都是0，有一个数据读写头指向纸带的初始位置，读写头的行为由指令指示。

指令	含义
>	指针向右移动一位
<	指针向左移动一位
+	指针所指位置的值增加1字节
-	指针所指位置的值减少1字节
.	将指针所指位置的值按ASCII表输出
,	接受1字节的输入，存储在当前指针所指位置
[	当指针当前处的值为0时，跳转到对应]之后；否则，顺序执行
]	跳转回对应[处

8. 3-1

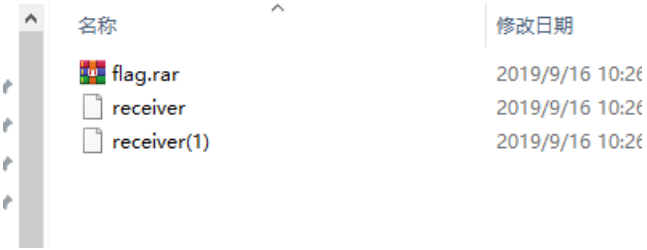
a.到手文件没有后缀，放进kali中发现是rar压缩文件，改后缀解压，发现又是一个没有后缀的文件，放进kali发现是一个流量包

b.使用wireshark打开这个流量包。搜索字符串（flag）发现存在一个压缩包



把这个压缩包先分离出来

c.导出http流，发现存在三个文件



这个压缩包还是加密过的，并不是伪加密，其他的两个文件并没有发现相关密码的东西，那么现阶段的目标就变成了找出这个压缩包的密码

d.把目光重新的放到wireshark中，一个一个追踪流，~~但我觉得这个方法超极笨~~，有效率高的方法欢迎师傅们评论，在其中的一个tcp流中发现有系统命令。

根据命令得知存在四个文件，其中这个1文件就是压缩包，2文件是一个base64，3文件是一个加密脚本

e.修改3中加密脚本如下

```python
# coding:utf-8

__author__ = 'YFP'

from Crypto import Random
from Crypto.Cipher import AES
import sys
import base64


V = 'QWERTYUIOPASDFGH'


def decrypt(encrypted):

  aes = AES.new(IV, AES.MODE_CBC, IV)

  return aes.decrypt(encrypted)


def encrypt(message):

  length = 16

  count = len(message)

  padding = length - (count % length)

  message = message + '\0' * padding

  aes = AES.new(IV, AES.MODE_CBC, IV)

  return aes.encrypt(message)


# str = 'this is a test'

# example = encrypt(str)

# print(decrypt(example))

str = '19aaFYsQQKr+hVX6hl2smAUQ5a767TsULEUebWSajEo='  #文件2中的base64

print(decrypt(base64.b64decode(str)))
```

f.跑一下脚本的到flag

9.适合作为桌面

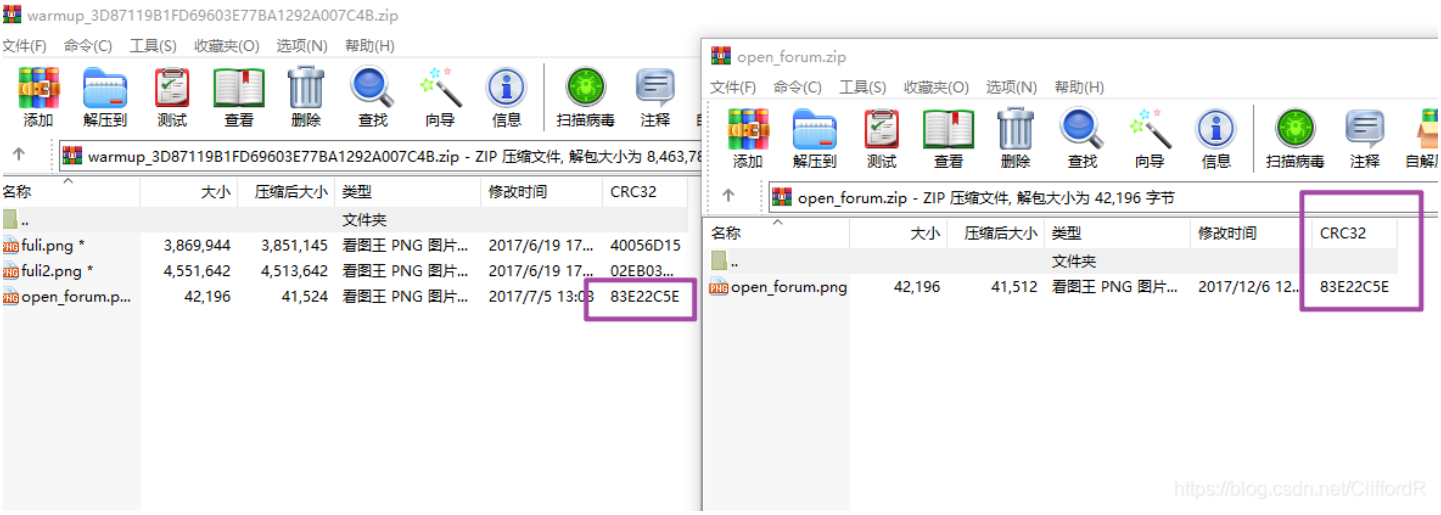
a.拿到手是一个压缩包，解压后是一张图片
b.常规操作使用winhex，notepad++，binwalk检测没有收获，使用stegsolve查看发现二维码
c.使用qrcode扫描出二维码是一串十六进制的字符，开头为03 F3 0D 0A，这是一个python反编译的.pyc文件的十六进制文件头。
d.把它放到winhex中还原一下，后缀改成.pyc
e.反编译，执行命令uncompyle6 PYC反编译.pyc

```
C:\Users\D-R0s1>uncompyle6 F:\ctf文件\PYC反编译.pyc
# uncompyle6 version 3.4.0
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 20:40:30) [MSC v.1500 64 bit (AMD64)]
# Embedded file name: 1.py
# Compiled at: 2016-10-18 15:12:57


def flag():
    str = [
     102, 108, 97, 103, 123, 51, 56, 97, 53, 55, 48, 51, 50, 48, 56, 53, 52, 52, 49, 101, 55, 125]
    flag = ''
    for i in str:
        flag += chr(i)

    print flag
# okay decompiling F:\ctf文件\PYC反编译.pyc

C:\Users\D-R0s1>
```

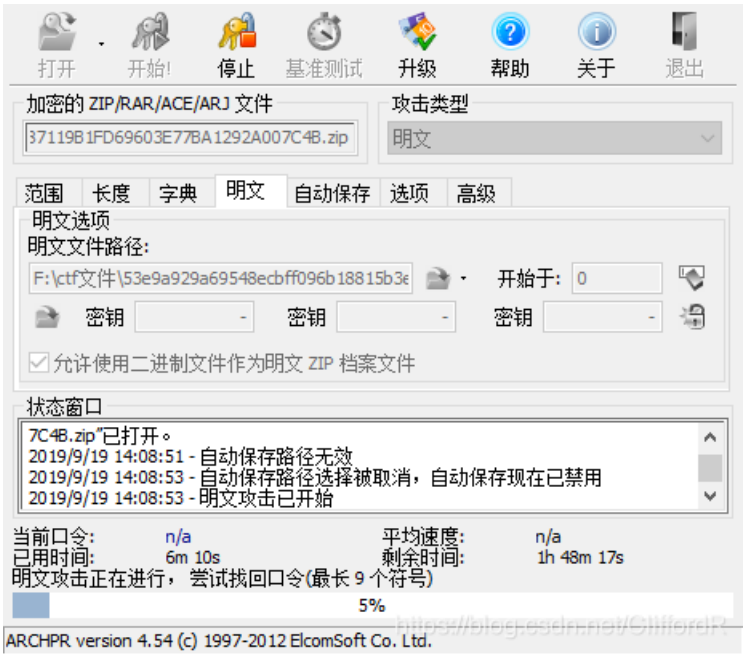

f.运行一下反编译出来的脚本得到flag

10.warmup

a.下载得到一张图片和一个压缩包，发现压缩包中存在密码，尝试在图片中寻找解压密码，并没有找到
b.考虑明文攻击，将图片压缩成zip文件，看一下crc32的值



两个值相同，那么可以肯定就是从明文攻击入手，使用archpr进行明文攻击



恢复后一个文件夹中有两个一样的照片，在stegsolve中比较一下发现有条纹，大概率是盲水印，进行盲水印攻击

```
root@D-R0s1:~/tools/盲水印/BlindWaterMark-master# python bwm.py decode '/root/to
ols/盲水印/BlindWaterMark-master/fuli.png'  '/root/tools/盲水印/BlindWaterMark-m
aster/fuli2.png'  123.png
image</root/tools/盲水印/BlindWaterMark-master/fuli.png> + image(encoded)</root/
tools/盲水印/BlindWaterMark-master/fuli2.png> -> watermark<123.png>
```

得到水印图

11.simple_transfer

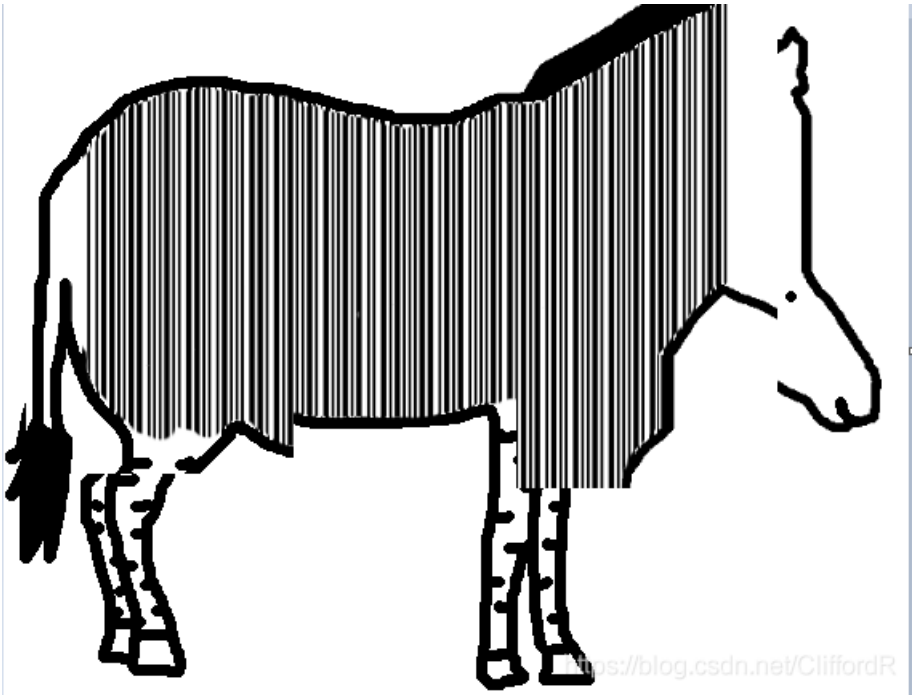
a.到手是一个流量包，放进wireshark中打开一波分析后并没有分析出什么东西来，只看到一个一个pdf样式的东西，断断续续的。
b.自己流量分析水平实在是low的一匹，干脆放到kali中foremost一下，结果把这个pdf分离出来了

六个金币的题目就这样一个foremost到手了，慌的一匹。

12.Banmabanma

a.到手一个斑马的图片，观察发现斑马的身上是条形码

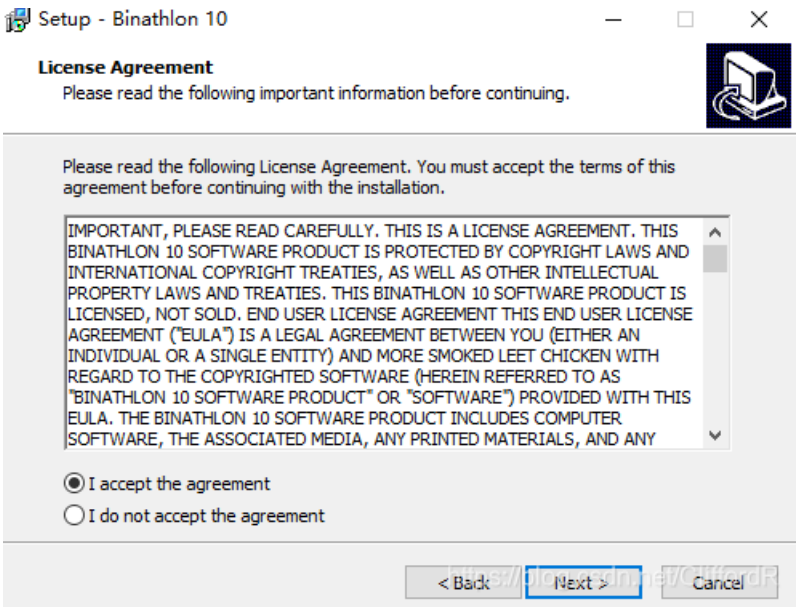
b.为了便于扫描条形码，我们把条形码上下都拉长一点，这里有个小技巧，在画图软件中打开图片，使用虚线框住，然后按住alt+上下键就可以在上或者下方拉长图片。最后得到这么一个图片。



虽然奇丑，已经没有马的样子，选取中间完整的部分截取出来放到bctester工具里识别一下就得到了flag。也可以使用在线识别网站

13. Just-No-One

a.在安装的那一段文本中找flag



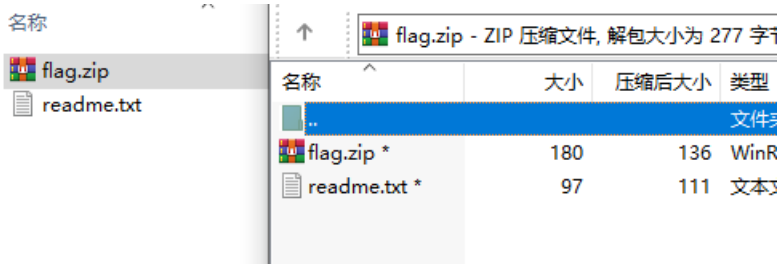
这就是flag

ED OF THE POSSIBILITY OF SUCH DAMAGES, OR ANY CLAIM BY A THIRD P
7A. YOU MAY SUBMIT THIS TO GET TEN POINTS: ILOVEREADINGEULAS. 8.
LEASE OR PREVIOUSLY RELEASED VERSION. YOU NOW MAY USE THAT UPGRA

14.我们的秘密是绿色的

a.到手是一张图片，一番尝试后无果继续看一下题目，我们的秘密是绿色的，刚好有一个工具的名字叫做our secret，使用our secret打开，发现需要密码，在刚开始的一番尝试中并没有发现什么像是密码的东西。
b.尝试图片上的绿色的数字，把它作为密码。
0405111218192526
成功得到一个名为try的压缩包
c.解压压缩包，需要密码，提示是生日，那么进行六位数字的密码爆破，没有找到密码，继续尝试一下八位数字的密码爆破。成功获得密码

d.解压后，又是一个压缩包，也是需要密码，并且发现，里外都有相同的文件

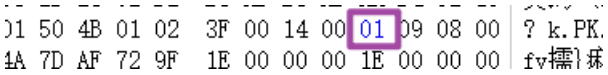


比较像是明文攻击

e.把readme.txt使用winrar压缩成zip，使用archpr进行明文攻击，得到密码

Y29mZmVl

f.解压得到flag.zip解压竟然还需要密码。。。。。无fuck说了，没有提示，也没相同的文件，先考虑一下伪加密把，winhex把01改成00，关于伪加密的知识可以参考一下这个师傅写的博客。



g.成功得到文本文件，先栅栏后凯撒得到flag。

15.Erik-Baleog-and-Olaf

http://taskcode3.cn/?p=116

这个题目倒是不难，但是要修复这个图片就太难了，附上一个链接

16.py-py-py

a.到手是一个.pyc文件，把它反编译一下

[外链图片转存失败(img-5glXrijm-1569205782128)(en-resource://database/753:1)]

反编译后的代码

```
import sys, os, hashlib, time, base64
fllag =
'9474yeUMWODKruX7OFzD9oekO28+EqYCZHrUjWNm92NSU+eYXOPsRPEFrNMs7J+4qautoqOrvq28pLU='
def crypto(string, op='encode', public_key='ddd', expirytime=0):
    ckey_lenth = 4
    public_key = public_key and public_key or ''
    key = hashlib.md5(public_key).hexdigest()
    keya = hashlib.md5(key[0:16]).hexdigest()
    keyb = hashlib.md5(key[16:32]).hexdigest()
    keyc = ckey_lenth and (op == 'decode' and string[0:ckey_lenth] or hashlib.md5(str(time.time())).hexdigest()[
32 - ckey_lenth:32]) or ''
    cryptkey = keya + hashlib.md5(keya + keyc).hexdigest()
    key_lenth = len(cryptkey)
    string = op == 'decode' and base64.b64decode(string[4:]) or '0000000000' + hashlib.md5(string + keyb).hexdig
est()[0:16] + string
    string_lenth = len(string)
    result = ''
    box = list(range(256))
    randkey = []
    for i in xrange(255):
        randkey.append(ord(cryptkey[(i % key_lenth)]))

    for i in xrange(255):
        j = 0
        j = (j + box[i] + randkey[i]) % 256
        tmp = box[i]
        box[i] = box[j]
        box[j] = tmp

    for i in xrange(string_lenth):
        a = j = 0
        a = (a + 1) % 256
        j = (j + box[a]) % 256
        tmp = box[a]
        box[a] = box[j]
        box[j] = tmp
        result += chr(ord(string[i]) ^ box[((box[a] + box[j]) % 256)])

    if op == 'decode':
        if result[0:10] == '0000000000' or int(result[0:10]) - int(time.time()) > 0:
            if result[10:26] == hashlib.md5(result[26:] + keyb).hexdigest()[0:16]:
                pass
            return result[26:]
        else:
            return
    else:
        return keyc + base64.b64encode(result)


if __name__ == '__main__':
    while True:
        flag = raw_input('Please input your flag:')
        if flag == crypto(fllag, 'decode'):
            print('Success')
            break
        else:
            continue
```

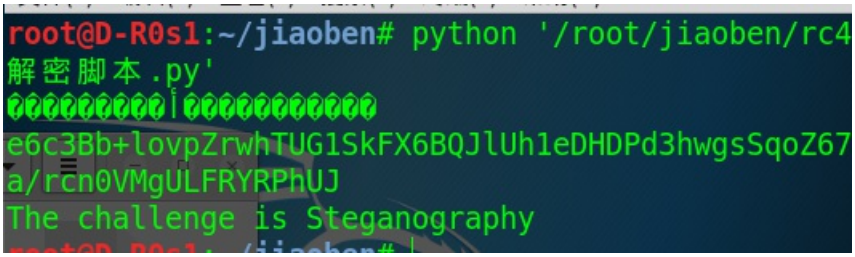
可以看出进行了rc4加密，ddd就是key

b.rc4算法脚本

```
# /usr/bin/python# coding=utf-8import sys, os, hashlib, time, base64


def rc4(string, op='encode', public_key='ddd', expirytime=0):
    ckey_lenth = 4
    public_key = public_key and public_key or ''
    key = hashlib.md5(public_key).hexdigest()
    keya = hashlib.md5(key[0:16]).hexdigest()
    keyb = hashlib.md5(key[16:32]).hexdigest()
    keyc = ckey_lenth and (
    op == 'decode' and string[0:ckey_lenth] or hashlib.md5(str(time.time())).hexdigest()[32 - ckey_lenth:32]) or
 ''
    cryptkey = keya + hashlib.md5(keya + keyc).hexdigest()
    key_lenth = len(cryptkey)
    string = op == 'decode' and base64.b64decode(string[4:]) or '0000000000' + hashlib.md5(string + keyb).hexdig
est()[
                                                                          0:16] + string
    string_lenth = len(string)

    result = ''
    box = list(range(256))
    randkey = []

    for i in xrange(255):
        randkey.append(ord(cryptkey[i % key_lenth]))

    for i in xrange(255):
        j = 0
        j = (j + box[i] + randkey[i]) % 256
        tmp = box[i]
        box[i] = box[j]
        box[j] = tmp

    for i in xrange(string_lenth):
        a = j = 0
        a = (a + 1) % 256
        j = (j + box[a]) % 256
        tmp = box[a]
        box[a] = box[j]
        box[j] = tmp
        result += chr(ord(string[i]) ^ (box[(box[a] + box[j]) % 256]))

    if op == 'decode':
        if (result[0:10] == '0000000000' or int(result[0:10]) - int(time.time()) > 0) and result[10:26] == hashl
ib.md5(
                        result[26:] + keyb).hexdigest()[0:16]:
            return result[26:]
        else:
            return None
    else:
        return keyc + base64.b64encode(result)

if __name__ == '__main__':
    string = '我在这里呢，你在那里呢'
    print(string)
    str = rc4(string, 'encode')
```

```
    str = rc4(string, 'encode')
    print(str)
    fllag = '9474yeUMWODKruX7OFzD9oekO28+EqYCZHrUjWNm92NSU+eYXOPsRPEFrNMs7J+4qautoqOrvq28pLU='
    rc = rc4(fllag, 'decode')
    print(rc)
```

这个脚本在python2中运行
运行结果



提示需要Steganography这个工具来解决
c.使用脚本

```
import argparse
import logging
import marshal
import opcode
import os
import py_compile
import sys
import math
import string
import types

print ('usage: python Stegosaurus.py -h')
print ('usage: python Stegosaurus.py 1.pyc -x')

if sys.version_info < (3, 6):
    sys.exit("Stegosaurus requires Python 3.6 or later")


class MutableBytecode():
    def __init__(self, code):
        self.originalCode = code
        self.bytes = bytearray(code.co_code)
        self.consts = [MutableBytecode(const) if isinstance(const, types.CodeType) else const for const in code.
co_consts]


def _bytesAvailableForPayload(mutableBytecodeStack, explodeAfter, logger=None):
    for mutableBytecode in reversed(mutableBytecodeStack):
        bytes = mutableBytecode.bytes
        consecutivePrintableBytes = 0
        for i in range(0, len(bytes)):
            if chr(bytes[i]) in string.printable:
                consecutivePrintableBytes += 1
            else:
                consecutivePrintableBytes = 0

            if i % 2 == 0 and bytes[i] < opcode.HAVE_ARGUMENT:
                if consecutivePrintableBytes >= explodeAfter:
```

```
                    if logger:
                        logger.debug("Skipping available byte to terminate string leak")
                    consecutivePrintableBytes = 0
                    continue
                yield (bytes, i + 1)


def _createMutableBytecodeStack(mutableBytecode):
    def _stack(parent, stack):
        stack.append(parent)

        for child in [const for const in parent.consts if isinstance(const, MutableBytecode)]:
            _stack(child, stack)

        return stack

    return _stack(mutableBytecode, [])


def _dumpBytecode(header, code, carrier, logger):
    try:
        f = open(carrier, "wb")
        f.write(header)
        marshal.dump(code, f)
        logger.info("Wrote carrier file as %s", carrier)
    finally:
        f.close()


def _embedPayload(mutableBytecodeStack, payload, explodeAfter, logger):
    payloadBytes = bytearray(payload, "utf8")
    payloadIndex = 0
    payloadLen = len(payloadBytes)

    for bytes, byteIndex in _bytesAvailableForPayload(mutableBytecodeStack, explodeAfter):
        if payloadIndex < payloadLen:
            bytes[byteIndex] = payloadBytes[payloadIndex]
            payloadIndex += 1
        else:
            bytes[byteIndex] = 0

    print("Payload embedded in carrier")


def _extractPayload(mutableBytecodeStack, explodeAfter, logger):
    payloadBytes = bytearray()

    for bytes, byteIndex in _bytesAvailableForPayload(mutableBytecodeStack, explodeAfter):
        byte = bytes[byteIndex]
        if byte == 0:
            break
        payloadBytes.append(byte)

    payload = str(payloadBytes, "utf8")

    print("Extracted payload: {}".format(payload))


def _getCarrierFile(args, logger):
    carrier = args.carrier
```

```
    carrier = args.carrier
    _, ext = os.path.splitext(carrier)

    if ext == ".py":
        carrier = py_compile.compile(carrier, doraise=True)
        logger.info("Compiled %s as %s for use as carrier", args.carrier, carrier)

    return carrier


def _initLogger(args):
    handler = logging.StreamHandler()
    handler.setFormatter(logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s'))

    logger = logging.getLogger("stegosaurus")
    logger.addHandler(handler)

    if args.verbose:
        if args.verbose == 1:
            logger.setLevel(logging.INFO)
        else:
            logger.setLevel(logging.DEBUG)

    return logger


def _loadBytecode(carrier, logger):
    try:
        f = open(carrier, "rb")
        header = f.read(12)
        code = marshal.load(f)
        logger.debug("Read header and bytecode from carrier")
    finally:
        f.close()

    return (header, code)


def _logBytesAvailableForPayload(mutableBytecodeStack, explodeAfter, logger):
    for bytes, i in _bytesAvailableForPayload(mutableBytecodeStack, explodeAfter, logger):
        logger.debug("%s (%d)", opcode.opname[bytes[i - 1]], bytes[i])


def _maxSupportedPayloadSize(mutableBytecodeStack, explodeAfter, logger):
    maxPayloadSize = 0

    for bytes, i in _bytesAvailableForPayload(mutableBytecodeStack, explodeAfter):
        maxPayloadSize += 1

    logger.info("Found %d bytes available for payload", maxPayloadSize)

    return maxPayloadSize


def _parseArgs():
    argParser = argparse.ArgumentParser()
    argParser.add_argument("carrier", help="Carrier py, pyc or pyo file")
    argParser.add_argument("-p", "--payload", help="Embed payload in carrier file")
    argParser.add_argument("-r", "--report", action="store_true", help="Report max available payload size carrie
r supports")
```

```
    argParser.add_argument("-s", "--side-by-side", action="store_true", help="Do not overwrite carrier file, ins
tall side by side instead.")
    argParser.add_argument("-v", "--verbose", action="count", help="Increase verbosity once per use")
    argParser.add_argument("-x", "--extract", action="store_true", help="Extract payload from carrier file")
    argParser.add_argument("-e", "--explode", type=int, default=math.inf, help="Explode payload into groups of a
 limited length if necessary")
    args = argParser.parse_args()

    return args


def _toCodeType(mutableBytecode):
    return types.CodeType(
        mutableBytecode.originalCode.co_argcount,
        mutableBytecode.originalCode.co_kwonlyargcount,
        mutableBytecode.originalCode.co_nlocals,
        mutableBytecode.originalCode.co_stacksize,
        mutableBytecode.originalCode.co_flags,
        bytes(mutableBytecode.bytes),
        tuple([_toCodeType(const) if isinstance(const, MutableBytecode) else const for const in mutableBytecode.
consts]),
        mutableBytecode.originalCode.co_names,
        mutableBytecode.originalCode.co_varnames,
        mutableBytecode.originalCode.co_filename,
        mutableBytecode.originalCode.co_name,
        mutableBytecode.originalCode.co_firstlineno,
        mutableBytecode.originalCode.co_lnotab,
        mutableBytecode.originalCode.co_freevars,
        mutableBytecode.originalCode.co_cellvars
        )


def _validateArgs(args, logger):
    def _exit(msg):
        msg = "Fatal error: {}\nUse -h or --help for usage".format(msg)
        sys.exit(msg)

    allowedCarriers = {".py", ".pyc", ".pyo"}

    _, ext = os.path.splitext(args.carrier)

    if ext not in allowedCarriers:
        _exit("Carrier file must be one of the following types: {}, got: {}".format(allowedCarriers, ext))

    if args.payload is None:
        if not args.report and not args.extract:
            _exit("Unless -r or -x are specified, a payload is required")

    if args.extract or args.report:
        if args.payload:
            logger.warn("Payload is ignored when -x or -r is specified")
        if args.side_by_side:
            logger.warn("Side by side is ignored when -x or -r is specified")

    if args.explode and args.explode < 1:
        _exit("Values for -e must be positive integers")

    logger.debug("Validated args")
```

```
def main():
    args = _parseArgs()
    logger = _initLogger(args)

    _validateArgs(args, logger)

    carrier = _getCarrierFile(args, logger)
    header, code = _loadBytecode(carrier, logger)

    mutableBytecode = MutableBytecode(code)
    mutableBytecodeStack = _createMutableBytecodeStack(mutableBytecode)
    _logBytesAvailableForPayload(mutableBytecodeStack, args.explode, logger)

    if args.extract:
        _extractPayload(mutableBytecodeStack, args.explode, logger)
        return

    maxPayloadSize = _maxSupportedPayloadSize(mutableBytecodeStack, args.explode, logger)

    if args.report:
        print("Carrier can support a payload of {} bytes".format(maxPayloadSize))
        return

    payloadLen = len(args.payload)
    if payloadLen > maxPayloadSize:
        sys.exit("Carrier can only support a payload of {} bytes, payload of {} bytes received".format(maxPayloa
dSize, payloadLen))

    _embedPayload(mutableBytecodeStack, args.payload, args.explode, logger)
    _logBytesAvailableForPayload(mutableBytecodeStack, args.explode, logger)

    if args.side_by_side:
        logger.debug("Creating new carrier file name for side-by-side install")
        base, ext = os.path.splitext(carrier)
        carrier = "{}-stegosaurus{}".format(base, ext)

    code = _toCodeType(mutableBytecode)

    _dumpBytecode(header, code, carrier, logger)


if __name__ == "__main__":
    main()
```

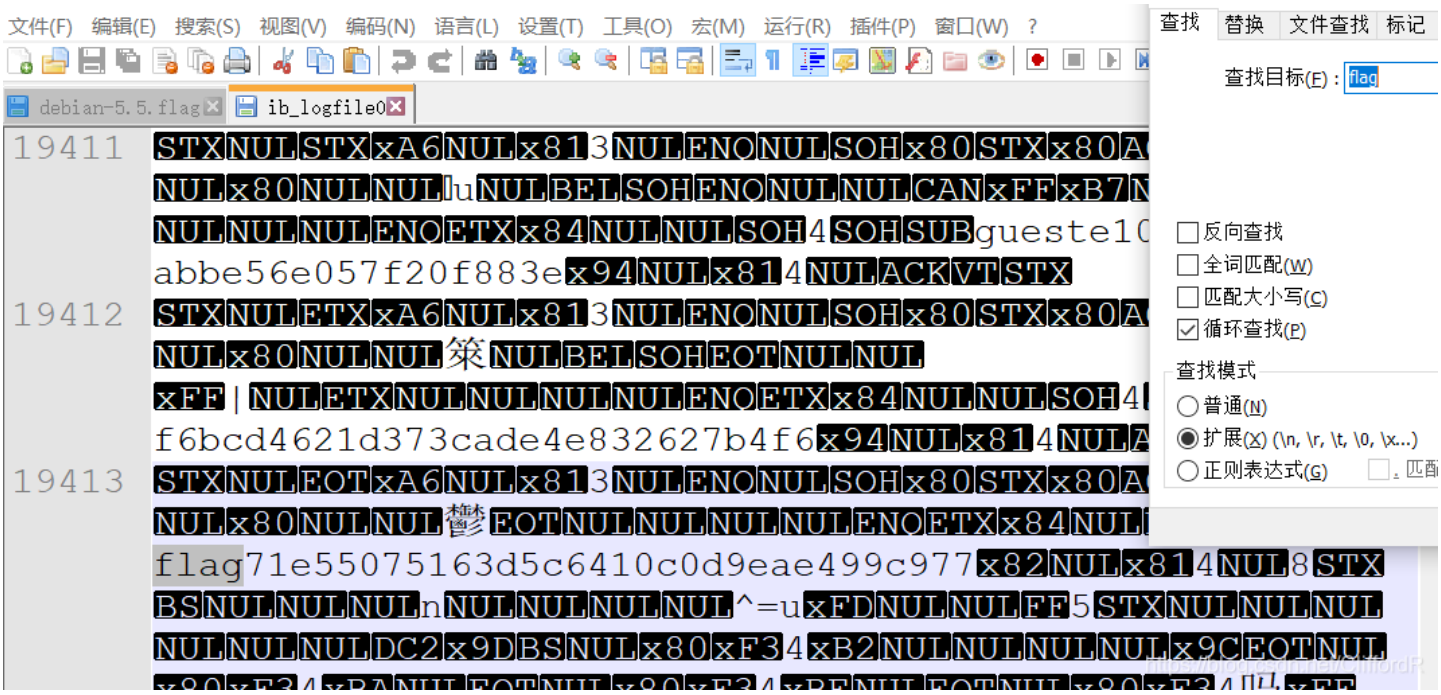
使用命令：python 脚本 需要提取的pyc文件 -x

```
C:\Users\D-ROs1>python F:\ctf脚本\Stegosaurus_pyc隐写.py F:\ctf文件\0ede1270c5c0433b94ad26d46a182b9c.pyc -x
usage: python Stegosaurus.py -h
usage: python Stegosaurus.py 1.pyc -x
Extracted payload: F1ag{HiD3_Pa10ad_1n_Python}

C:\Users\D-ROs1>
```

得到flag

17.mysql

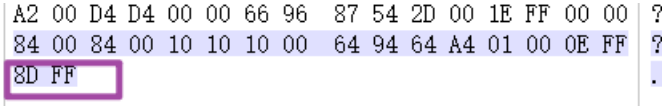
a.到手是一个压缩包，把压缩解压后有两个文件，其中一个是文件夹，打开文件夹在notepad++里查看文件内容，根据题目提示，存储了flag，那么在文件中查询flag。
b.一个文件一个文件的找，当找到ib_logfile0这个文件的时候，查询到了flag



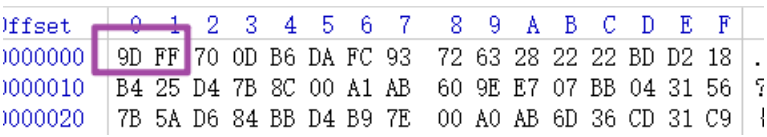
后面这一串就是flag，这也算是个非预期解了。

18.Reverse-it

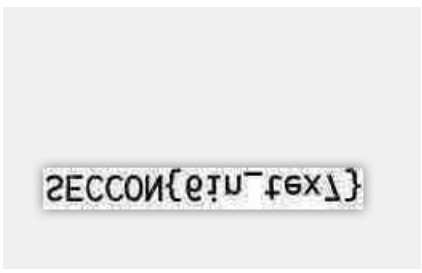
a.拿到题后，一番尝试无果，继续在winhex中查看有什么端倪，结合题目的提示，Reverse-it，也就是反向的意思
b.看一下文件的末尾



反过来也就是FFD8也就是jpg的文件头，开头是



也就是jpg的文件尾
c.把里面的十六进制数值复制出来，进行字符反转，然后写入winhex中保存为jpg格式得到一张图片



虽然是反着的，但也可以读出flag了。