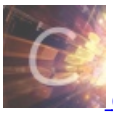
[Misc]2022DASCTF Apr X FATE 防疫挑战赛 wp

原创

是Mumuzi 于 2022-04-24 13:03:05 发布 2050 收藏 9

分类专栏： buuctf ctf 文章标签： 信息安全

本文链接：https://blog.csdn.net/qq_42880719/article/details/124371307



文章目录

Misc

SimpleFlow

一个签到题，一个个翻tcp流发现在第50流压缩了一个flag.txt，在第52发现该flag.zip,原始数据提取出来后发现需要密码，密码是在压缩时提供的，该流量为蚁剑流量，于是在第50流找到`g479cf6f058cf8=1DY2QgIi9Vc2Vycy9jaGFuZy9TaXRlcy90ZXN0Ijt6aXAgLVAgUGFTc1ppUFdvckQgZmxhZy56aXAgLi4vZmxhZy50eHQ7ZWNobyBbU107cHdkO2VjaG8gW0Vd`

base64解码`Y2QgIi9Vc2Vycy9jaGFuZy9TaXRlcy90ZXN0Ijt6aXAgLVAgUGFTc1ppUFdvckQgZmxhZy56aXAgLi4vZmxhZy50eHQ7ZWNobyBbU107cHdkO2VjaG8gW0Vd`

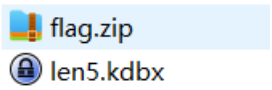
得到密码`PaSsZiPWorD`

解开压缩包得到flag

```
DASCTF{f3f32f434eddbc6e6b5043373af95ae8}
```

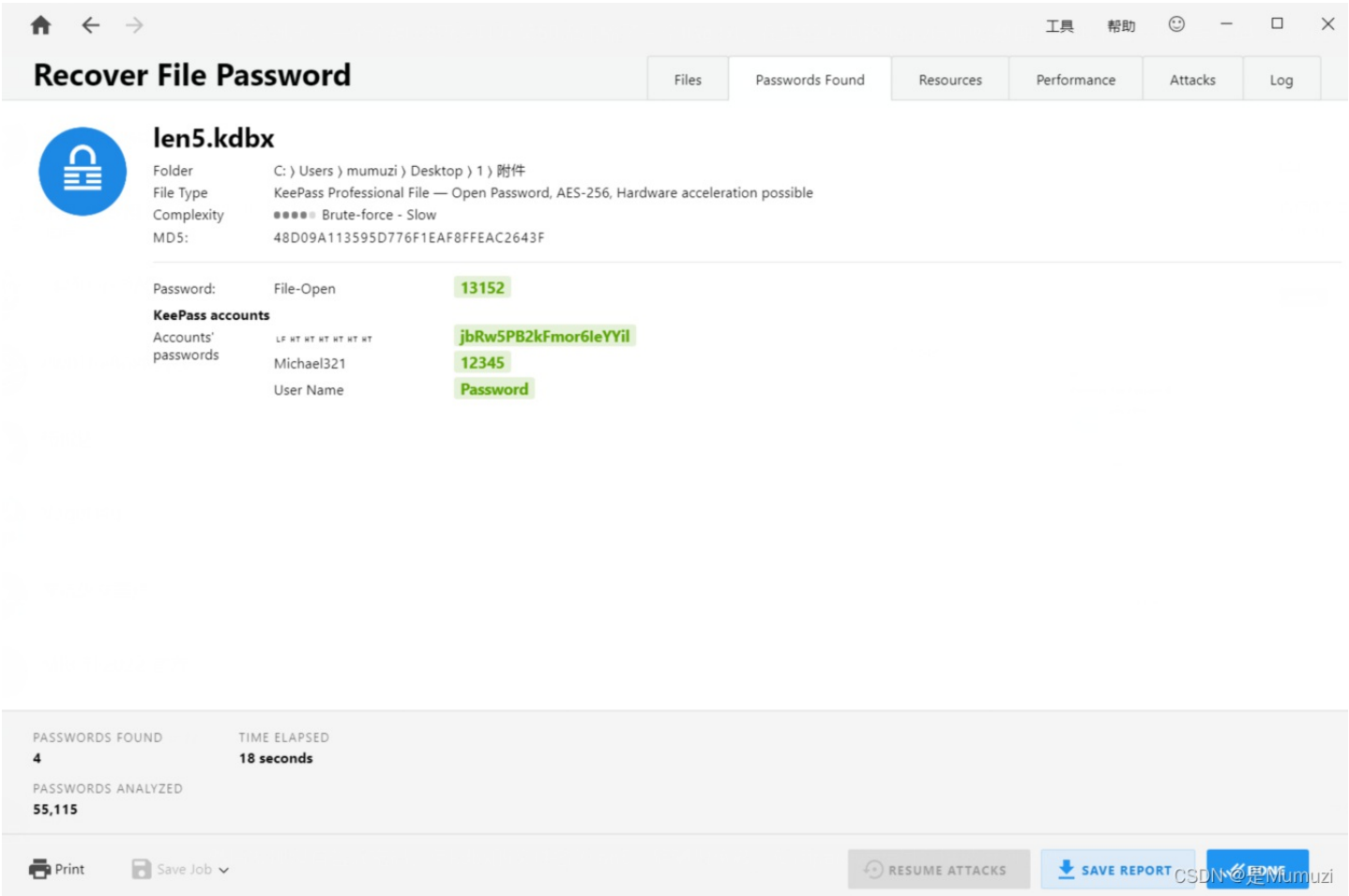
熟悉的猫

首先解压得到两个文件



这个kdbx正好上周做国外取证题的时候用过，因此知道是用什么工具打开

然后发现没有告诉密码，但kdbx的文件名为len5，猜测为爆破，使用passware进行爆破,18秒后得到密码



解开的密码为13152

里面的内容是`jbRw5PB2kFmor6IeYYil`，用该密码解压压缩包。得到一个hint和一张图片

根据图片特征和题目，猜测为猫脸变换，这里只给了一串数字根据积累能联想到可能有两种解密的方式。

均尝试后发现没有反应或者报错，因此我想用脚本解。发现出题人塞了个零宽进去

Original Text: Clear (length: 1063)

k=928982032787029079297059386766720215003947914272057573691234892045653003
248597170824098926419512066645649919914893546618714258726495240780009481998
832659815275909285198829276929014694628110159824930931595166203271443269827
744950570765508584256368206091081394250450793662555573558591327357505011855
235319268295531022032346346540864542233410144647107893314928733624177244833
338428740302833855616421538520769267636119285948674549756604384946996184385
540750545616824012331978580090993321469571182801348398173193377301733694465
563975838722671267677785497450878547943028089501009665825587612244542420184

Hidden Text: Clear (length: 6)

22*160

Encode »

« Decode

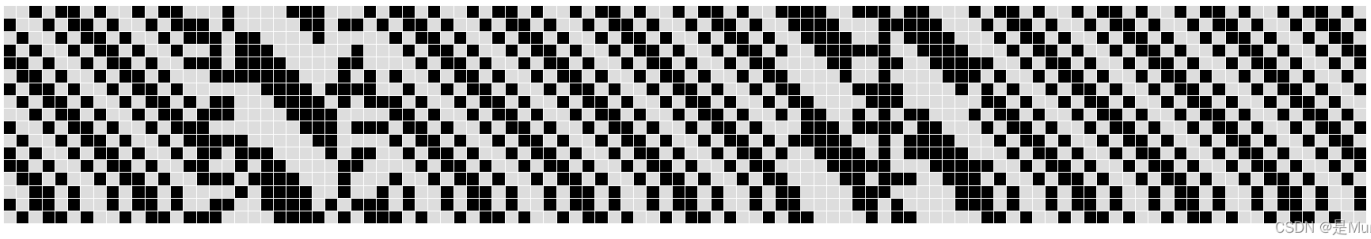
Steganography Text: Clear (length: 1111)

k=928982032787029079297059386766720215003947914272057573691234892045653003248597170824098926419512066645649919914893546618714258726495240780009481998326598152759092851988292769290146946281101598249309315951662032714432698274495059707655085842563682060910813942504507936625555735585913273575050118552353192682955310220323463465408645422334101446471078933149287336241772448338428740302833855616421538520769267636119285948674549756604384946996184385407505456168240123319785800909933214695711828013483981731933773017336944656397583872267126767778549745087854794302808950100966582558761224454242018467578959766617176016660101690140279961968740323327369347164623746391335756442566959352876706364265509834319910419399748338894746638758652286771979896573695823608678008814861640308571256880794312652055957150464513950305355055495262375870102898500643010471425931450046440860841589302890250456138060738689526283389256801969190204127358098408264204643882520969704221896973544620102494391269663693407573658064279947688509910028257209987991480259150865283245150325813888942058

Download Stego Text as File

这里给了一个22*160，试了下发现不是猫脸的参数，因此尝试去解k，用tupper自指发现貌似有点规律

5150325813888942058 作画



但是吧，这里和网上其他online decode都用的是标准的 17*106

这里专门给说22*160，于是去搜了个脚本改了一下大小

```python
def Tuppers_Self_Referential_Formula():
    k = 92898203278702907929705938676672021500394791427205757369123489204565300324859717082409892641951206664564
99199148935466187142587264952407800094819983265981527590928519882927692901469462811015982493093159516620327144326
98274495057076550858425636820609108139425045079366255557355859132735750501185523531926829553102203234634654086454
22334101446471078933149287336241772448338428740302833855616421538520769267636119285948674549756604384946996184385
40750545616824012331978580090993321469571182801348398173193377301733694465639758387226712676777854974508785479430
28089501009665825587612244542420184675789597666171760166601016901402799619687403233273693471646237463913357564425
66959352876706364265509834319910419399748338894746638758652286771979896573695823608678008814861640308571256880794
31265205595715046451395030535505549526237587010289850064301047142593145004644086084158930289025045613806073868952
62833892568019691902041273580984082642046438825209697042218969735446201024943912696636934075736580642799476885099
10028257209987991480259150865283245150325813888942058

    # 这里替换为你自己的K值

    def f(x, y):
        d = ((-22 * x) - (y % 22))
        e = reduce(lambda x, y: x * y, [2 for x in range(-d)]) if d else 1
        g = ((y // 22) // e) % 2
        return 0.5 < g

    for y in range(k + 21, k - 1, -1):
        line = ""
        for x in range(0, 160):
            if f(x, y):
                line += " ■"
            else:
                line += "  "
        print(line)


if __name__ == '__main__':
    Tuppers_Self_Referential_Formula()
```

然后再用知乎上一个参数正好反了的猫脸脚本去解，当然有小的改动

```python
import cv2
import numpy as np

def arnold_decode(image, shuffle_times, a, b):
    """ decode for rgb image that encoded by Arnold
    Args:
        image: rgb image encoded by Arnold
        shuffle_times: how many times to shuffle
    Returns:
        decode image
    """
    # 1:创建新图像
    decode_image = np.zeros(shape=image.shape,dtype=int)

    # 2: 计算N
    h, w = image.shape[0], image.shape[1]
    N = h  # 或N=w

    # 3: 遍历像素坐标变换
    for time in range(shuffle_times):
        for ori_x in range(h):
            for ori_y in range(w):
                # 按照公式坐标变换
                new_x = ((a * b + 1) * ori_x + (-b) * ori_y) % N
                new_y = ((-a) * ori_x + ori_y) % N
                decode_image[new_x, new_y, :] = image[ori_x, ori_y, :]
    return decode_image

img = cv2.imread("flag.png")  #变换的图片
a = 121
b = 144
st = 1
pic = arnold_decode(img,st,a,b)
cv2.imwrite('flag2.png',pic)  #保存得到的图片
```

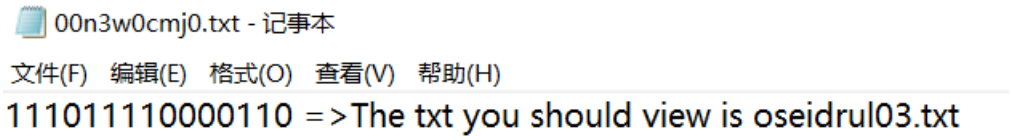
得到flag图片

```
DASCTF{751476c0-6cff-497f-9541-83ede0ebc5a0}
```

冰墩墩

解压出来是10w个txt文本，其中能找到start.txt

然后随便打开一个文本，内容如下



脚本倒是很好写了，看了一下start.txt

```
101000001001011 =>The txt you should view is m9312r95cr.txt 1 m9312r95cr.txt
1100000100 =>The txt you should view is 4oyjbqwl59.txt 2 4oyjbqwl59.txt
101000000000 =>The txt you should view is 68xaswpxdb.txt 3 68xaswpxdb.txt
0 =>The txt you should view is g2ny4snblo.txt 4 g2ny4snblo.txt
0 =>The txt you should view is vmsc9h6e1b.txt 5 vmsc9h6e1b.txt
1100000110010010 =>The txt you should view is tv9qllntzj.txt 6 tv9qllntzj.txt
101110001010100 =>The txt you should view is g0lr1v5ypb.txt 7 g0lr1v5ypb.txt
0 =>The txt you should view is 50enwszh2b.txt 8 50enwszh2b.txt
0 =>The txt you should view is v9oia0j69h.txt 9 v9oia0j69h.txt
```


这里把第一个前面补上0，就是504b，把第二行前面补上0，能凑出0304，发现这里每段都需要zfill(16)，因此写出如下脚本

```python
import re
flag = ''
next = 'start.txt'
i = 0

while len(next) != 0:
    try:
        f = open(next).read()
        tmp = re.search('(.*) =>',f).group(0)
        flag += tmp[:-3].zfill(16)
        tmp = re.search('is (.*)',f).group(0)
        next = tmp[3:]
        i += 1
        print(f,i,next)
    except:
        f2 = open('../out.txt','w')
        f2.write(flag)
        exit(0)
```

得到01串之后用Cyberchef转换一下

保存出来，得到一个pyc和一个无拓展名的文件，无拓展名的文件修一下发现是jpg。

然后用uncompyle6去逆一下pyc，发现逆出来也就是个画图的，因此猜要用剑龙

试了一下3.9和3.10版本都报错，还好服务器上有个3.6.9，解了一下解出了一串，交不上后就开始猜测为jpg隐写

```
ubuntu@VM-12-9-ubuntu:~/stegosaurus-master$ python3 stegosaurus.py -x BinDunDun.pyc
b'3\r\r\n.\xe5\x11b\x90\x1c\x00\x00'
Extracted payload: BingD@nD@n_in_BeiJing_Winter_Olympics
```

`BingD@nD@n_in_BeiJing_Winter_Olympics`

一般国外喜欢用steghide，而国内喜欢用jphs05，然后就用jphs解出来了

`REFTQ1RGe0dvb2RfSm9kX0dpdmVfVGhlX0ZGRkZMQGdfVG9fWW91IX0=`

```
DASCTF{Good_Jod_Give_The_FFFFL@g_To_You!}
```