利用hashcat进行7z压缩包破解&图片恢复 —— 【i春秋2021春季联赛】Baby_steg

原创

Ve99 于 2021-06-02 16:42:41 发布 2230 收藏 4

分类专栏： [MISC]-CTF 文章标签： python 信息安全 算法 安全 windows

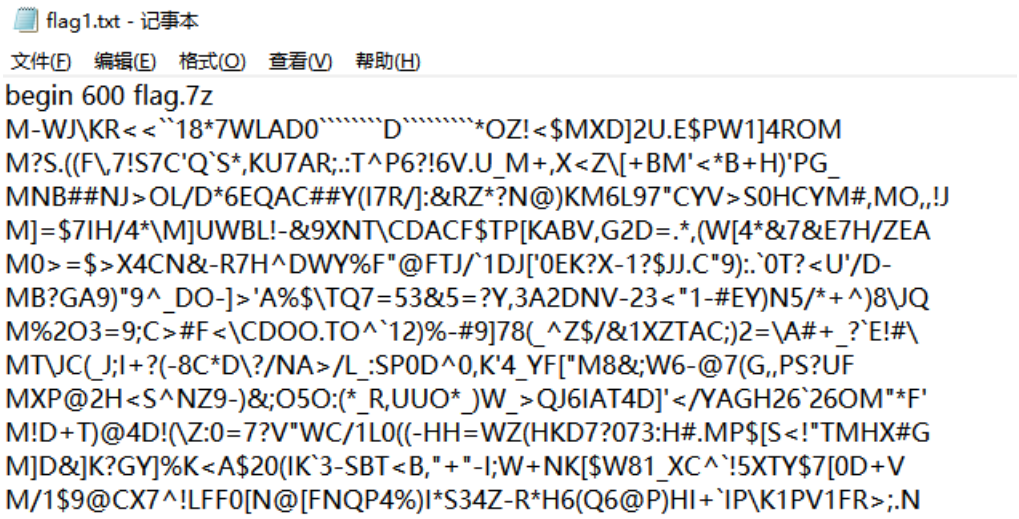
本文链接：https://blog.csdn.net/qq_42939527/article/details/117469351

文章目录

题目分析

题目提供了一个名为`password.7z`的加密压缩包，以及`flag1.txt`的文件

`flag1.txt内容：`

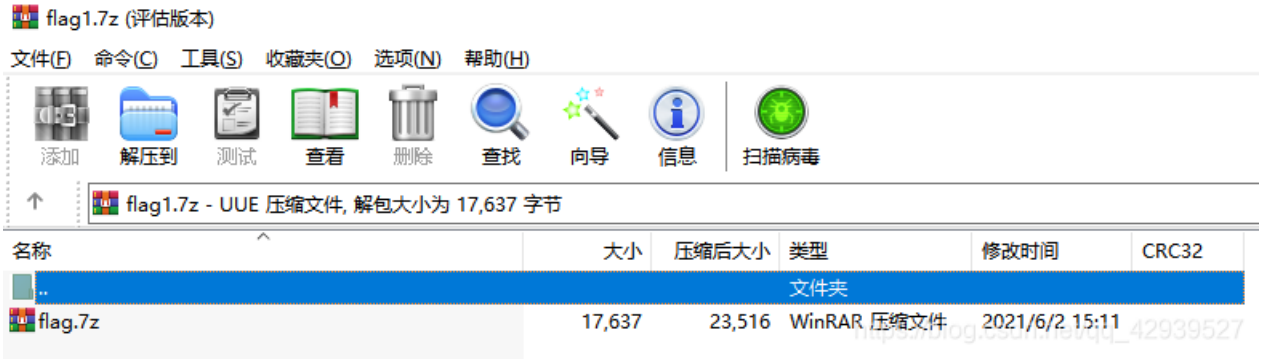


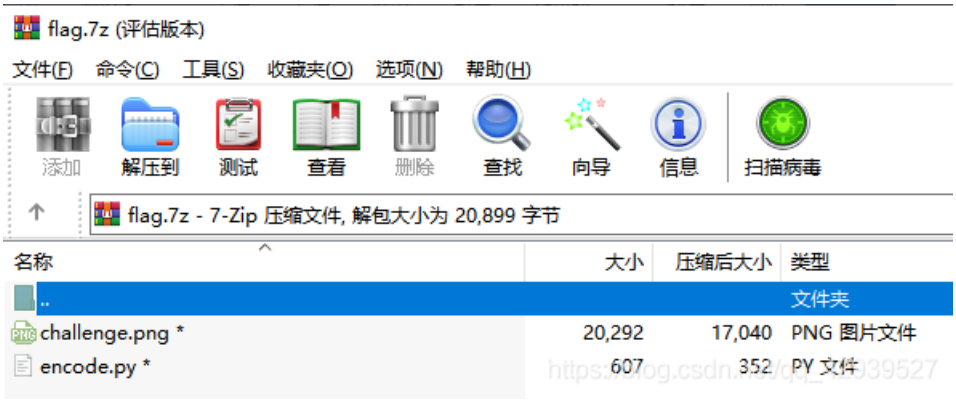
```
M+JEU]QC!4`?2]L7F^[+.?UF%CZAPZ7>`0B-4/8QLO76=G^1&W?O/Y`T@TS26
M4#%9_D*%C3AG%2Q?=`Y3'W"BZV#2%?FPIV[O@FN1-S>3#NL!QV91YYMG[#SM
M;VAWA1OKNRP1DB1RUU%HI('^<]72WL%0BUELMCF;8=817&/;G[4WO\-8,*&^
ML$_3VC\9R,C\PE`GST&"K\]CYJ6>)B5P)X;.J=V^S%&BP-OJFL1LMA/YTI*6
M(\BM$DHQ"_%!X@<ZA]J.X6ZZ+12`L3_(`=[4WES8_H-2;M??ON!ROT<7\Y)=
MI@[/83["4@ASC1LR<\V6*P8%;D]PR2)D936G.JN?82'H'2&M@WI&+NB]+&I^
M2*Y<4SJ0K"TVKS&4[S$SI?P@U5CX>HAMBM#UPYA[!H@C3%1I"W!9G`OC\Y9E
```

将flag1.txt丢进kali进行文件分离，发现应该是个7z的文件

```
└─$ binwalk -D=".*" flag1.txt

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             uuencoded data, file name: "flag.7z", file perm
issions: "600"
```

因此将后缀`txt`改为`7z`，发现存在一个压缩包`flag.7z`的压缩包，同样进行了加密

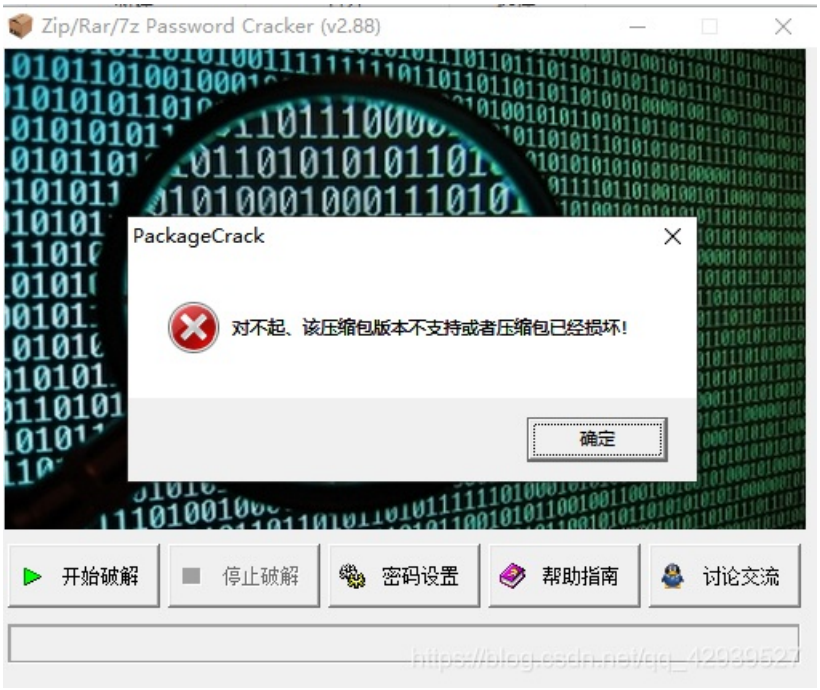




猜测`password.7z`里的`password.txt`文件存在解压`flag.7z`的密码

7z压缩包破解

根据题目hint，password.7z压缩包的密码是6-8位纯数字，使用`PackageCrack`破解提示不支持！



更换方法，采用`hashcat`进行破解

生成hash值

下载John the Ripper: https://github.com/openwall/john
下载perl: https://strawberryperl.com/

使用`john`的脚本`7z2john.pl`计算出`password.7z`的hash值

```
Tools\john\run>perl 7z2john.pl password.7z
ATTENTION: the hashes might contain sensitive encrypted data. Be careful when
 sharing or posting these hashes
password.7z:$7z$2$19$0$$16$abc477f84f711f5530432e64418c8392$3167568243$16$12$
40a31f0f88ac7b9a9acdc6cbb7d23f23$8$00
```

得到password.7z的hash值`$7z$2$19$0$$16$abc477f84f711f5530432e64418c8392$3167568243$16$12$40a31f0f88ac7b9a9acdc6cbb7d23f23$8$00`

hashcat密码爆破

采用hashcat进行6位纯数字的密码爆破

`-m` 指定文件类型，7z是11600，可以通过`hashcat --help`查看

`-a` 指定攻击类型 3为掩码模式，后接文件hash值

`?d` 指当前位置为数字类型

`-O` 采用自动优化方式

`-w` 指定电力消耗

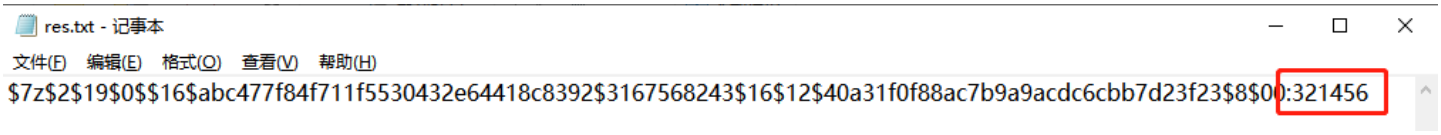
`-o` 指定结果输出文件

`--force` 忽略warning强制执行

```
C:\VE99\Tools\hashcat-4.1.0\hashcat-4.1.0>hashcat64.exe -m 11600 -a 3 $7z$2$1
9$0$$16$abc477f84f711f5530432e64418c8392$3167568243$16$12$40a31f0f88ac7b9a9ac
dc6cbb7d23f23$8$00 ?d?d?d?d?d?d -O -w 4 -o res.txt --force
hashcat (v4.1.0) starting...

OpenCL Platform #1: Intel(R) Corporation
========================================
* Device #1: Intel(R) UHD Graphics 620, 1612/3225 MB allocatable, 24MCU

./OpenCL/m11600-optimized.cl: Optimized OpenCL kernel requested but not neede
d - falling back to pure OpenCL kernel
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Applicable optimizers:
* Zero-Byte
* Single-Hash
* Single-Salt
* Brute-Force

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Initialized device kernels and memory...      https://blog.csdn.net/qq_42939527
```

```
[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit =>

Session..........: hashcat
Status...........: Running
Hash.Type........: 7-Zip
Hash.Target......: $7z$2$19$0$$16$abc477f84f711f5530432e64418c8392$316...3$8$
00
Time.Started.....: Wed Jun 02 15:33:03 2021 (2 secs)
Time.Estimated...: Wed Jun 02 20:14:57 2021 (4 hours, 42 mins)
Guess.Mask.......: ?d?d?d?d?d?d [6]
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:       59 H/s (98.74ms) @ Accel:128 Loops:32 Thr:256 Vec:1
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 0/1000000 (0.00%)
Rejected.........: 0/0 (0.00%)
Restore.Point....: 0/100000 (0.00%)
Candidates.#1....: 123456 -> 176464
HWMon.Dev.#1.....: N/A                    https://blog.csdn.net/qq_42939527
```

hashcat自动停止，说明破解成功，查看res.txt密码为`321456`



打开password.txt，得到flag.7z的解压密码`7324623c`

图片恢复

打开flag.7z，存在一张图片`challenge.png`与加密脚本`encode.py`
encode.py:

```
import numpy as np
import cv2
import sys
import random

def encode(image):
    i = random.randint(520,540)
    np.random.seed(i)
    # image = 1298 * 695
    to_hide = cv2.imread(image)
    to_hide_array = np.asarray(to_hide)

    for i in range(to_hide_array.shape[0]):
        np.random.shuffle(to_hide_array[i])

    gray = cv2.cvtColor(to_hide_array, cv2.COLOR_BGR2GRAY)
    cv2.imwrite('challenge.png', gray)
    print("encode!")

def main():
    if len(sys.argv) != 2:
        print('error!')
        exit(1)
    encode(sys.argv[1])

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

加密脚本分析

选取一个520-540的随机数，并确定为随机种子

```
i = random.randint(520,540)
np.random.seed(i)
```

导入待加密的图片，并将图片像素转换为矩阵(三维)

```
to_hide = cv2.imread(image)
to_hide_array = np.asarray(to_hide)
```

矩阵的形式是：695行，每一行1298列，每一列是一个3个元素的一维列表
下面的for循环是利用shuffle函数`随机扰乱每一行各列的相对顺序`，但是一开始设定了随机种子，因此随机扰乱实际上是`伪随机`，即使用确定的随机种子，对初始图片进行上述的扰乱，得到的结果是`一致的`

```
for i in range(to_hide_array.shape[0]):
 np.random.shuffle(to_hide_array[i])
```

将扰乱后的矩阵转换为灰度图并恢复图片

```
gray = cv2.cvtColor(to_hide_array, cv2.COLOR_BGR2GRAY)
cv2.imwrite('challenge.png', gray)
```

解密脚本编写

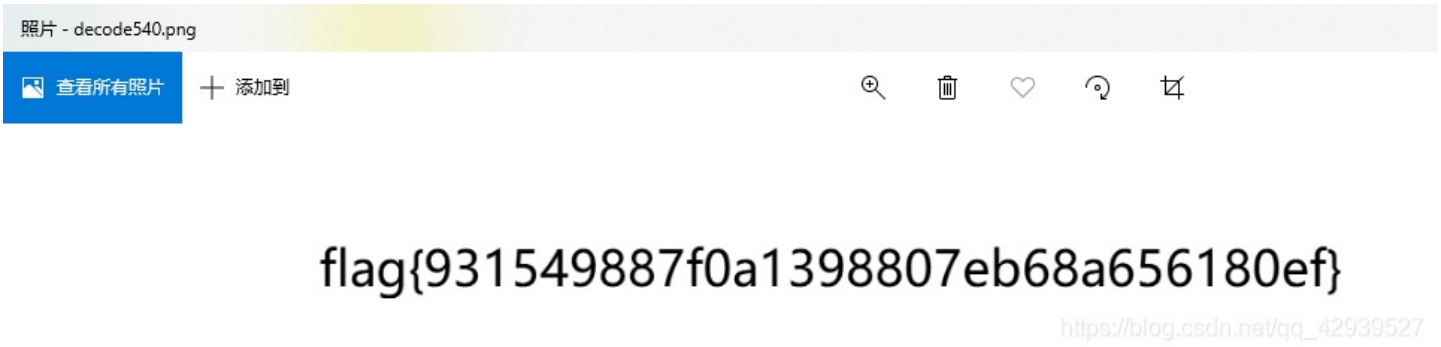
根据加密脚本的分析，假设确定随机种子为520，那么可以构造一个与二维的`695*1298`矩阵`tmp`，`即一共695行，每一行有1298列，并且每一行都是0-1297`。这样，在利用shuffle函数对其进行扰乱时，`每一行中各列的相对位置就与加密图片的扰乱方式一致，那么就能够根据tmp的每一行各列的值来还原加密图片矩阵的每一行`。
由于不确定加密时随机种子具体的值，但知道其范围，因此`对每一个随机种子都进行一次还原`，再从还原出来的图片中去寻找flag

decode.py:

```
import numpy as np
import cv2
import sys
import random

def decode(sd,image):
    np.random.seed(sd)
    to_hide = cv2.imread(image)
    to_hide_array = np.asarray(to_hide)

    # 构造695*1298的二维矩阵，且每一行均为0-1297
    tmp = [list(range(to_hide_array.shape[1]))]*to_hide_array.shape[0]
    tmp = np.asarray(tmp)

    # 对tmp矩阵进行随机种子为sd下的列扰动
    for i in range(to_hide_array.shape[0]):
        np.random.shuffle(tmp[i])

    # 构造一个与加密图片维度一样的矩阵，即695*1298*3，用来存放还原的数据
    re = [[[0,0,0]]*to_hide_array.shape[1]]*to_hide_array.shape[0]
    re = np.asarray(re)

    # tmp的每一行的每一列的值，代表着加密图片矩阵当前行，当前列原本的位置
    # eg：tmp[0][0] = 5 ， 那么to_hide_array中第一行，第一列是从第一行，第五列经过伪随机扰动过来的
    for i in range(to_hide_array.shape[0]):
        for j in range(to_hide_array.shape[1]):
            re[i][tmp[i][j]]=to_hide_array[i][j]

    # 恢复图片
    cv2.imwrite('decode'+str(sd)+'.png', re)
    print("decode!")


for i in range(520,541):
    decode(i,'challenge.png')
```

获得flag

查看生成的20张图片，发现随机种子为540时，还原出了flag



flag{931549887f0a1398807eb68a656180ef}