Week2-ChenMoFeiJin

Crypto

signin

描述

签到题 233

解题思路

注：由于 `a` `c` `p` 为服务器随机提供的值，故题解中不表面具体数值，只提供解题思路。

已知 `a` `c` `p`，且满足 $c \equiv a^p \times m \pmod p$，求 `m`。

因为 `p` 为质数，由费马小定理可得 $a^{p-1} \equiv 1 \pmod p$

所以原式等价于 $c \equiv a \times m \pmod p$

即求 $ax \equiv c \pmod p$ 的解

通过 `扩展GCD` 算法，可获得其最小正整数解 `m`

由于 `m = s2n(flag)`，所以 `flag = n2s(m)`

```python
from libnum import * # 提供 xgcd 和 n2s 函数

a, b, c = ...

x, y, g = xgcd(a, p) # 获取 ExGCD 提供的 ax + by = GCD(a,b) = g 的解
m = ((x * c // g) % p + p) % p # 得到 ax = c (mod p) 的解，并保证是正数
FLAG = n2s(m) # 将 m 还原回 FLAG

print(FLAG)
```

最终得到 Flag，`hgame{M0du1@r_m4th+1s^th3~ba5is-Of=cRypt0!!}`

gcd or more?

描述

GCD...?

解题思路

注：由于 `p` `q` `cipher` 为服务器随机提供的值，故题解中不表面具体数值，只提供解题思路。

由题意得 $flag^2 \equiv chiper \pmod p$，直接通过 `libnum` 提供的函数即可求解

```
from libnum import * # 提供 sqrtmod 和 n2s 函数

p, q, cipher = ...

for m in sqrtmod(cipher, {p: 1, q: 1}): // 由于可能存在多个解所以都输出，只有一个能解出 FLAG
    print(n2s(m))
```

最终得到 Flag，`hgame{3xgCd~i5_re4l1y+e@sy^r1ght?}`

WhitegiveRSA

描述

n = 882564595536224140639625987659416029426239230804614613279163
e = 65537
c = 747831491353896780365654517748216624798517769637260742155527

解题思路

已知 `n` `e` `c` 理论上是难以解除私钥 `d` 的，故直接暴力破解得：

```
p = 857504083339712752489993810777
q = 1029224947942998075080348647219
d = 121832886702415731577073962957377780195510499965398469843281
```

带入下程序

```
from libnum import * # 提供n2s函数
print(n2s(pow(c, d, n)))
```

最终得到 Flag，`hgame{w0w~yOU_kNoW+R5@!}`

The Password

描述

Tinmix和朋友一起去玩密室逃脱,但是由于突发情况,Tinmix被锁在了一间密室里,于是开始四处摸索,昏暗的灯光下,Tinmix发现密室有一块大圆盘,被人工分割成了7块小圆盘,但由于刚开始没注意,每个圆盘已经被旋转过了,但Tinmix记住了旋转的过程和结果

$$
\begin{gathered}
y_1 = x_1 \oplus n_1 \oplus (x_1 \ggg 7) \oplus (x_1 \lll 3) \\
y_2 = x_2 \oplus n_2 \oplus (x_2 \ggg 4) \oplus (x_2 \lll 9) \\
y_3 = x_3 \oplus n_3 \oplus (x_3 \ggg 2) \oplus (x_3 \lll 5) \\
y_4 = x_4 \oplus n_4 \oplus (x_4 \ggg 6) \oplus (x_4 \lll 13) \\
y_5 = x_5 \oplus n_5 \oplus (x_5 \ggg 8) \oplus (x_5 \ggg 16) \\
y_6 = x_6 \oplus n_6 \oplus (x_6 \ggg 5) \oplus (x_6 \lll 7) \\
y_7 = x_7 \oplus n_7 \oplus (x_7 \ggg 2) \oplus (x_7 \lll 5) \\
(y_1, n_1) = (15789597796041222200, 14750142427529922) \\
(y_2, n_2) = (8279663441787235887, 2802568775308984) \\
(y_3, n_3) = (9666438290109535850, 15697145971486341) \\
(y_4, n_4) = (10529571502219113153, 9110411034859362) \\
(y_5, n_5) = (8020289479524135048, 4092084344173014) \\
(y_6, n_6) = (10914636017953100490, 2242282628961085) \\
(y_7, n_7) = (4622436850708129231, 10750832281632461)
\end{gathered}
$$

定义

$\ggg$表示循环右移
$\lll$表示循环左移
$\oplus$ 表示异或运算

解题思路

由于题目中七个方程各不相干（没有相同的变量），所以各个方程可以分开求解

由于循环左移与循环右移可以相互转换：$x \ggg k = x \lll (t \times n - k), t \in \mathbb{Z}, n = 64$（$n$ 是从题目数据中分析出的）

根据异或运算的归零率（$a \oplus a = 0$）和交换律（$a \oplus b = b \oplus a$）

方程可化为 $y \oplus n = x \oplus (x \ggg p) \oplus (x \ggg q)$

将 $y \oplus n$ 记作 y，方程简化为 $y = x \oplus (x \ggg p) \oplus (x \ggg q)(1)$

两边同时循环右移 p 位，得 $y \ggg p = (x \ggg p) \oplus (x \ggg 2p) \oplus (x \ggg (p+q))(2)$

两边同时循环右移 q 位，得 $y \ggg q = (x \ggg q) \oplus (x \ggg (p+q)) \oplus (x \ggg 2q)(3)$

$(1) \oplus (2) \oplus (3)$ 得，$y \oplus (y \ggg p) \oplus (y \ggg q) = x \oplus (x \ggg 2p) \oplus (x \ggg 2q)$

简化得 $y' = x \oplus (x \ggg p') \oplus (x \ggg q')$，其中 $y' = y \oplus (y \ggg p) \oplus (y \ggg q), p' = 2p, q' = 2q$

该方程与原方程同解，但 p 和 q 变为原方程的两倍，通过这种变换，我们可以将 p 和 q 变换为 $p \times 2^k$ 和 $q \times 2^k$，$k \in \mathbb{Z}$

而 $x \ggg (t \times n) = x, t \in \mathbb{Z}$ 所以只需将上述变换操作 8 次，即可得
$y^{(8)} = x \oplus (x \ggg p \times 64) \oplus (x \ggg q \times 64) = x$

下面是编程实现

```python
from libnum import *  # 提供n2s函数

def move(n, k):  # 循环位移函数, k > 0 往左移, k < 0 往右移
    s = bin(n)[2:].zfill(64)  # 将数据转成 64 位二进制字符串
    k &= 63  # 防止下标越界
    return int(s[k:] + s[:k], 2)  # 左移 k 位后返回

def calc(y, p, q, k):
    return y if k == 0 else calc(y ^ move(y, p) ^ move(y, q), p << 1, q << 1, k - 1) # 实现上述逻辑

y = [15789597796041222200, 8279663441787235887, 9666438290109535850, 10529571502219113153, 8020289479524135048,
     10914636017953100490, 4622436850708129231]
n = [14750142427529922, 2802568775308984, 15697145971486341, 9110411034859362, 4092084344173014, 2242282628961085,
     10750832281632461]
p = [-7, -4, -2, -6, -8, -5, -2]  # 左移为正，右移为负
q = [3, 9, 5, 13, -16, 7, 5]

x = [calc(ty ^ tn, tp, tq, 8) for ty, tn, tp, tq in zip(y, n, p, q)]  # 计算所有的 x
flag = "".join([str(n2s(tx))[2:-1] for tx in x])  # 将所有 x 转化为字符串并拼接起来

print(flag)
```

优化

若能让 $p \times 2^k = t \times n$，那么 $x \ggg (p \times 2^k) = x$，这样就可以减少迭代的次数

那么方程最终变换为 $y^{(k)} = x \oplus (x \ggg (p \times 2^k)) \oplus (x \ggg (q \times 2^k)) = x \ggg (p \times 2^k)$

解得 $x = y^{(n)} \lll (q \times 2^k) = y \oplus [\oplus_{i=1}^{k} y \ggg (i \times p)] \oplus [\oplus_{i=1}^{k} y \ggg (i \times q)] \lll (q \times 2^k)$

下面是编程实现

```python
from libnum import * # 提供n2s函数

def move(n, k): # 循环位移函数，k > 0 往左移，k < 0 往右移
    s = bin(n)[2:].zfill(64) # 将数据转成 64 位二进制字符串
    k &= 63 # 防止下标越界
    return int(s[k:] + s[:k], 2) # 左移 k 位后返回

def calc(y, p, q):
    if p & 63 != 0 and q & 63 != 0: # 判断 p 或 q 是否被 64 整除
        return calc(y ^ move(y, p) ^ move(y, q), (p << 1) & 63, (q << 1) & 63) # 不满足条件继续迭代
    return move(y, -q if p & 32 == 0 else -p) # 满足条件返回答案

y = [15789597796041222200, 8279663441787235887, 9666438290109535850, 10529571502219113153, 8020289479524135048, 10914636017953100490, 4622436850708129231]
n = [14750142427529922, 2802568775308984, 15697145971486341, 9110411034859362, 4092084344173014, 2242282628961085, 10750832281632461]
p = [-7, -4, -2, -6, -8, -5, -2] #  左移为正，右移为负
q = [3, 9, 5, 13, -16, 7, 5]

x = [calc(ty ^ tn, tp, tq) for ty, tn, tp, tq in zip(y, n, p, q)] # 计算所有的 x
flag = "".join([str(n2s(tx))[2:-1] for tx in x]) # 将所有 x 转化为字符串并拼接起来

print(flag)
```

最终得到 flag，`hgame{l1ne0r_a1gebr0&is@1mpor10n1^1n$crypto}`

MISC

Tools

描述

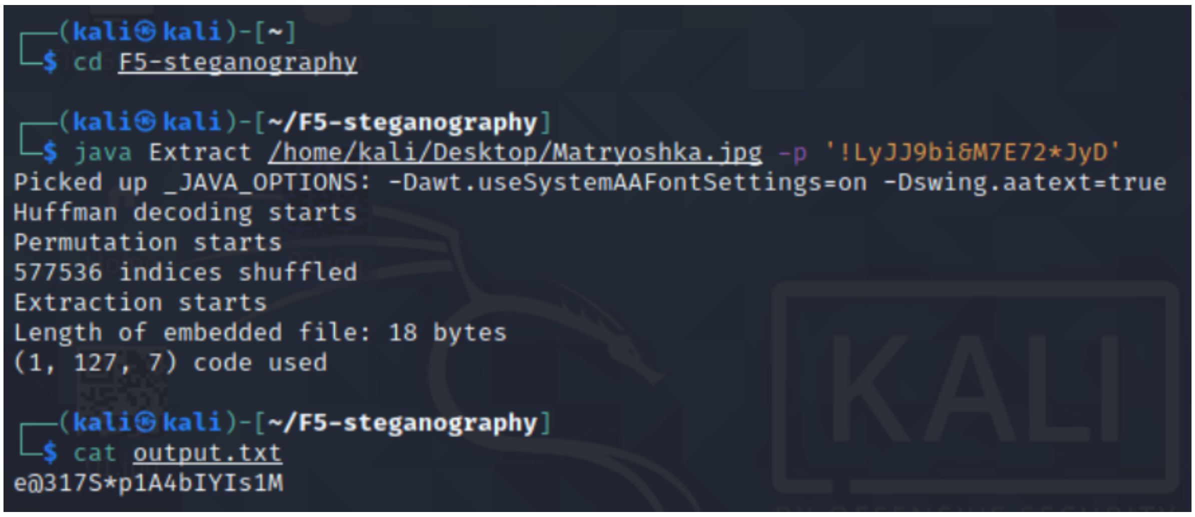
工欲善其事，必先利其器。

解题思路

打开压缩包，里面有一张图片和一个名字为 `F5` 的压缩包，压缩包有密码

搜索 `F5隐写` 得到解密方法，图片有备注 `!LyJJ9bi&M7E72*JyD` 为隐写的秘钥

借助 `F5-steganography` 工具解密，在文件 `output.txt` 中得到第一个压缩包的密码 `e@317S*p1A4bIYIs1M`

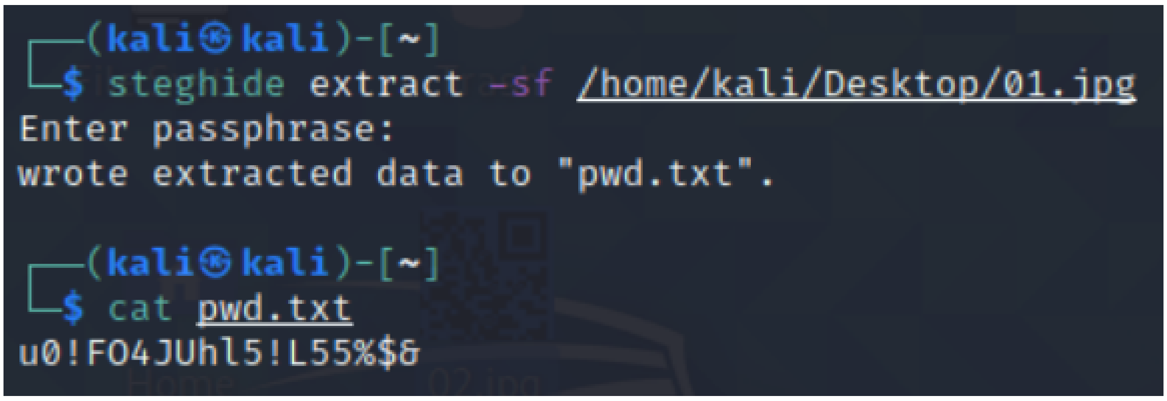


```
┌──(kali㉿kali)-[~]
└─$ cd F5-steganography

┌──(kali㉿kali)-[~/F5-steganography]
└─$ java Extract /home/kali/Desktop/Matryoshka.jpg -p '!LyJJ9bi&M7E72*JyD'
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Huffman decoding starts
Permutation starts
577536 indices shuffled
Extraction starts
Length of embedded file: 18 bytes
(1, 127, 7) code used

┌──(kali㉿kali)-[~/F5-steganography]
└─$ cat output.txt
e@317S*p1A4bIYIs1M
```

打开压缩包，里面有一张图片和一个名字为 `Steghide` 的压缩包，压缩包有密码

搜索 `Steghide` 得到解密方法，图片有备注 `A7SL9nHRJXLh@$EbE8` 为隐写的秘钥

借助 `steghide` 工具解密，在文件 `pwd.txt` 中得到第二个压缩包的密码 `u0!FO4JUhl5!L55%$&`

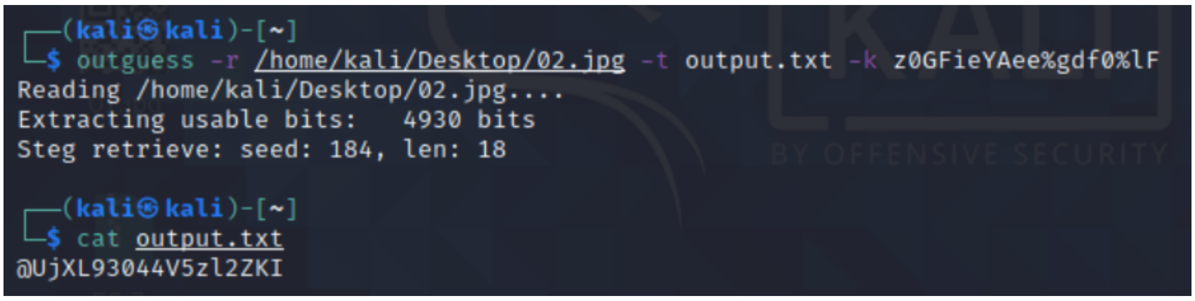


```
┌──(kali㉿kali)-[~]
└─$ steghide extract -sf /home/kali/Desktop/01.jpg
Enter passphrase:
wrote extracted data to "pwd.txt".

┌──(kali㉿kali)-[~]
└─$ cat pwd.txt
u0!FO4JUhl5!L55%$&
```

打开压缩包，里面有一张图片和一个名字为 `Outguess` 的压缩包，压缩包有密码

搜索 `Outguess` 得到解密方法，图片有备注 `z0GFieYAee%gdf0%lF` 为隐写的秘钥

借助 `Outguess` 工具解密，在文件 `output.txt` 中得到第三个压缩包的密码 `@UjXL93044V5zl2ZKI`

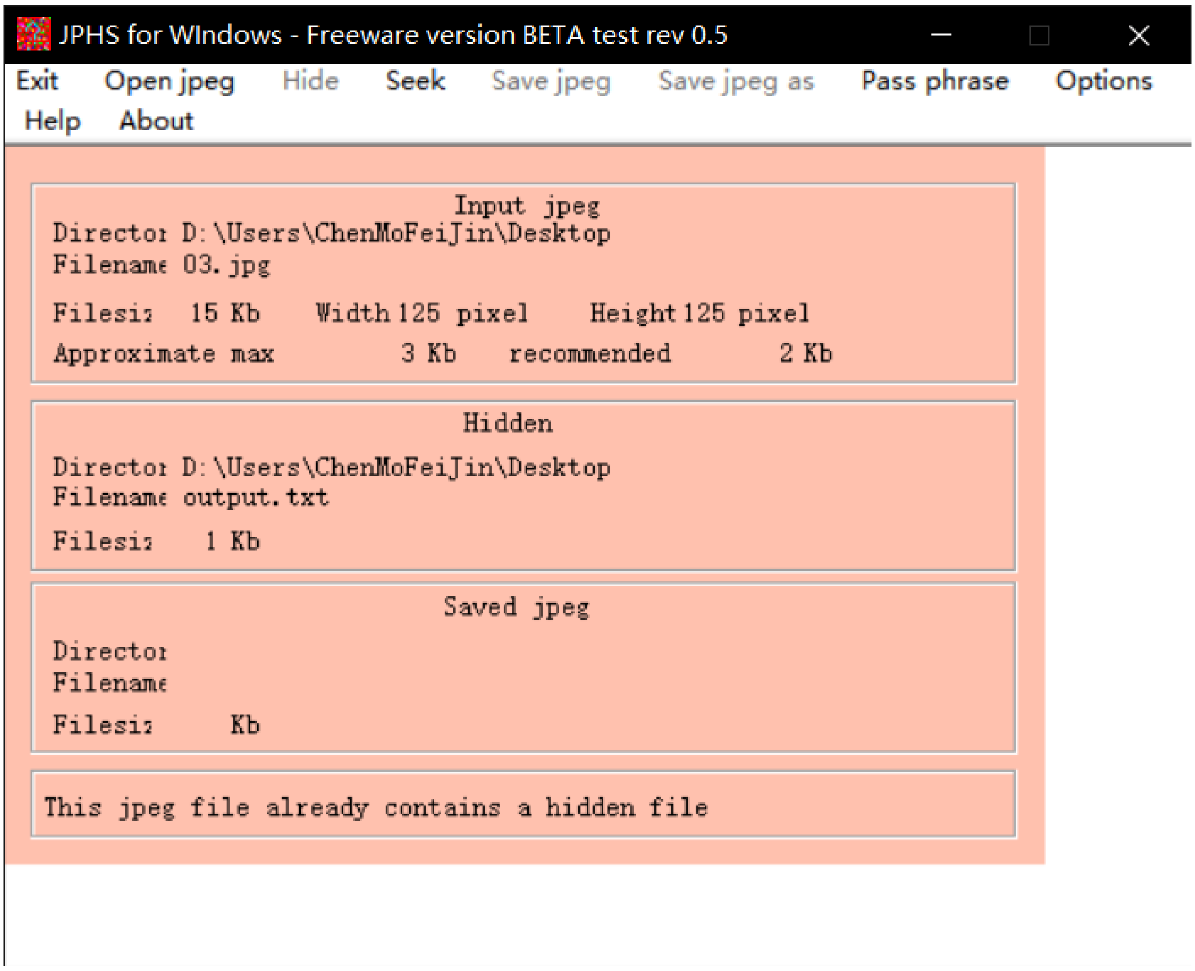


```
┌──(kali㉿kali)-[~]
└─$ outguess -r /home/kali/Desktop/02.jpg -t output.txt -k z0GFieYAee%gdf0%lF
Reading /home/kali/Desktop/02.jpg....
Extracting usable bits:   4930 bits
Steg retrieve: seed: 184, len: 18

┌──(kali㉿kali)-[~]
└─$ cat output.txt
@UjXL93044V5zl2ZKI
```

打开压缩包，里面有一张图片和一个名字为 `JPHS` 的压缩包，压缩包有密码

搜索 `JPHS` 得到解密方法，图片有备注 `rFQmRoT5lze@4x4^@0` 为隐写的秘钥

借助 `JPHS05` 工具的 `seek` 功能解密，在文件 `output.txt` 中得到第四个压缩包的密码 `xSRejK1^Z1Cp9M!z@H`



```
Input jpeg
Director D:\Users\ChenMoFeiJin\Desktop
Filename 03.jpg

Filesiz   15 Kb    Width 125 pixel    Height 125 pixel
Approximate max      3 Kb    recommended      2 Kb

Hidden
Director D:\Users\ChenMoFeiJin\Desktop
Filename output.txt

Filesiz    1 Kb

Saved jpeg

Director
Filename

Filesiz      Kb

This jpeg file already contains a hidden file
```

将得到的四个二维码碎片拼起来并扫描

最终得到 Flag，`hgame{Taowa_is_N0T_g00d_but_T001s_is_Useful}`

Telegraph：1601 6639 3459 3134 0892

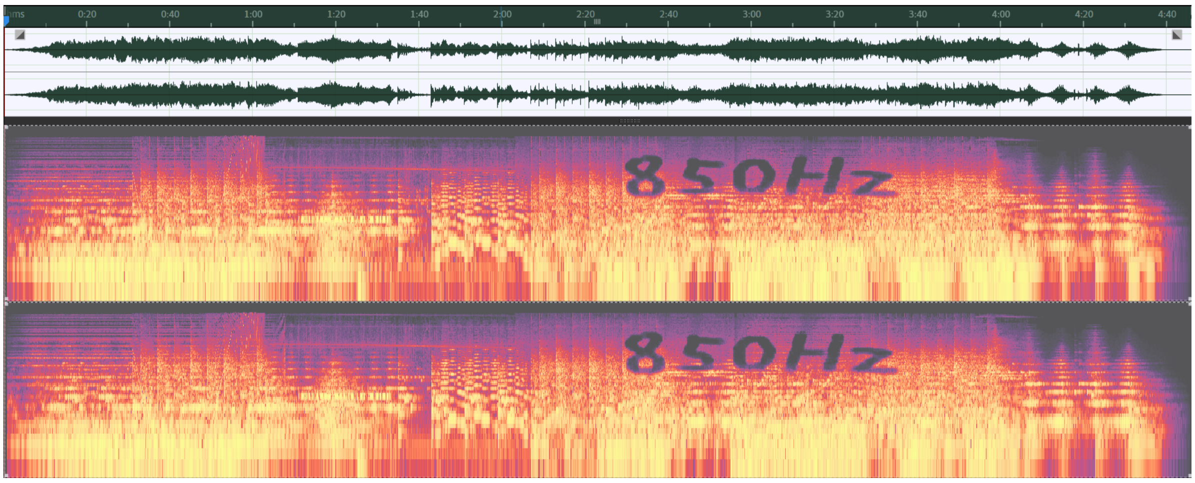
描述

他曾经最喜欢的曲师写的曲子，让人犹如漫步在星空之下，可如今他听见只觉得反胃。
请将flag以hgame{your_flag_here}形式提交，flag为全大写。

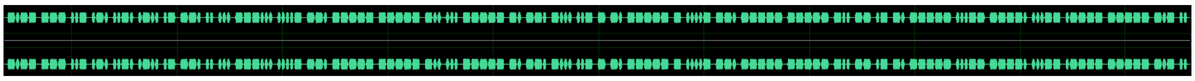
解题思路

用百度搜索题目包含的数字，得到关键字 `中文电报码`，使用在线工具解码可得关键字 `带通滤波器`

用 `Au` 打开题目提供的音频文件，并打开频谱视图，得到提示 `850Hz`



使用 带通滤波器 分离在 850Hz 附近的音频，得到一段莫斯电码



```
-.-- --- ..- .-. ..-.  .-.. .- --. .. ... ---... ....- --. ----- ----- -.. ... ----- -. --. -... ..- - -. ----- - ....- --. ----- ----- -.. -- .- -. ----- ...-- ----. ...-- .---- ----- -.- ..
```

解码后得到

```
YOURFLAGIS4G00DS0NGBUTN0T4G00DMAN039310KI
```

最终得到 Flag，`hgame{4G00DS0NGBUTN0T4G00DMAN039310KI}`

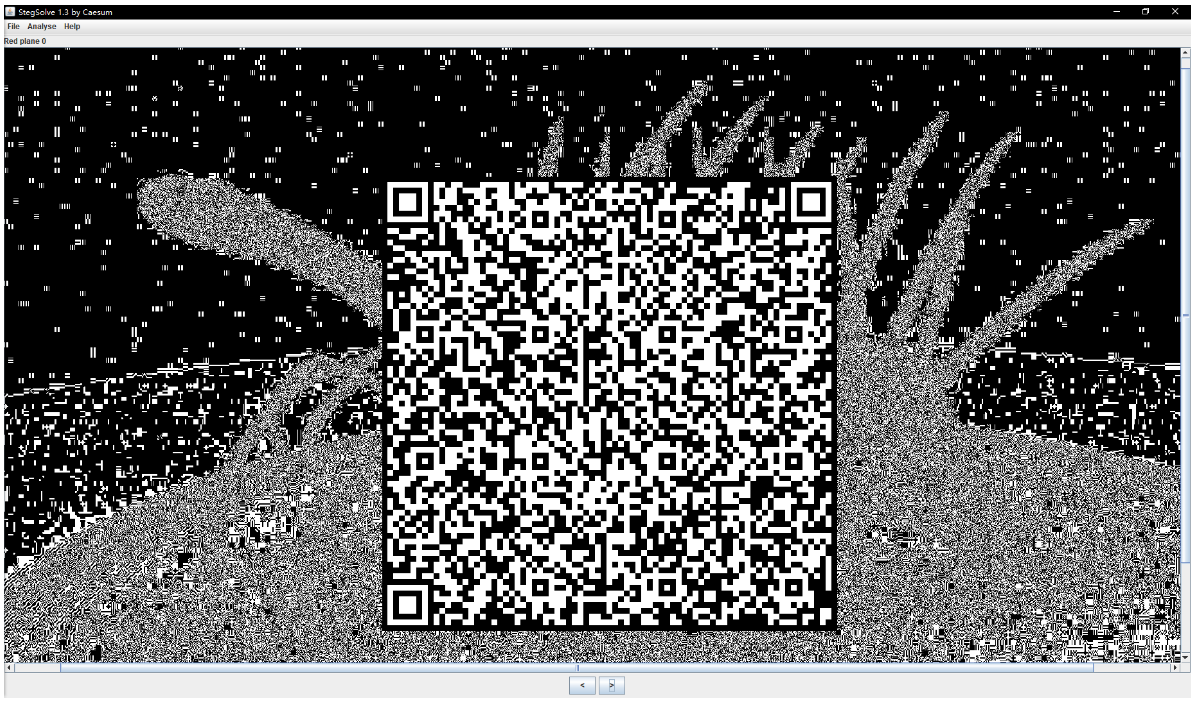
Hallucigenia

描述

“我们不仅弄错了他的上下，还颠倒了它的左右。”

解题思路

用 `Stegsolve` 打开，在 RGB 图层 0 比特平面，均能发现一个二维码



扫码后得到一段字符

```
gmBCrkRORUkAAAAA+jrgsWajaqOBeC3IQhCEIQhCKZw1MxTzSlNKnmJpivW9IHVPrTjvkkuI3sP7b
WAEdIHWCbDsGsRkZ9IUJC9AhfZFbpqrmZBtI+ZvptWC/KCPrLOgFeRPOcI2WyqjndfUWlNj+dgWpe
1qSTEcdurXzMRAc5EihsEflmIN8RzuguWq61JWRQpSI51/KHHT/6/ztPZJ33SSKbieTa1C5koONbL
cf9aYmsVh7RW6p3SpASnUSb3JuSvpUBKxscbyBjiOpOTq8jcdRsx5/IndXw3VgJV6iO1+6jl4gjVp
WouViO6ih9ZmybSPkhaqyNUxVXpV5cYU+Xx5sQTfKystDLipmqaMhxIcgvplLqF/LWZzIS5PvwbqO
vrSlNHVEYchCEIQISICSZJijwu5OrRQHDyUpaF0y///p6FEDCCDFsuW7YFoVEFESTOBAACLgLOrAA
AAAggUAAAAtAAAAFJESEkNAAAAChoKDUdOUIk=
```

观察发现好像使用的是 BASE64 编码，解码后却发现是一段乱码

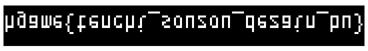
```
`BDNEI□□□□:○j ┌x-Â┼!◘B)53¶JSJbi uO8□↗m`┘t:읤dgҕ$/@Enm#耗ᒉ┴脵
¶[*ᆩZSɑ�1Lv꜡Ās"_b
L项챓E
R# (qxl')MB抎5Š핼t┌)ᆪı+鎊Ʊ–8I頂7↔Fȩ̈
ⱦz�9x5iZ□ɴ┬ü1UzU曻|y┘╤-↕Le. -fs!.O–꺤ҕs◀!◘B┼!"┐lb♂ɪɪPL<t̡D ┬+콬
TADI=┌        ┐◘¶        RDHI
□□□
→

GNP
```

但仔细观察，译文后有 GNP 字样，倒过来发现是 PNG，于是猜测是一段倒过来的 PNG 数据

编写代码将其翻转并导出到 flag.png 文件中

```python
import base64

s =
'gmBCrkRORUkAAAAA+jrgsWajaqOBeC3IQhCEIQhCKZw1MxTzSlNKnmJpivW9IHVPrTjvkkuI3sP7
bWAEdIHWCbDsGsRkZ9IUJC9AhfZFbpqrmZBtI+ZvptWC/KCPrLOgFeRPOcI2WyqjndfUWlNj+dgWp
e1qSTEcdurXzMRAc5EihsEflmIN8RzuguWq61JWRQpSI51/KHHT/6/ztPZJ33SSKbieTa1C5koONb
Lcf9aYmsVh7RW6p3SpASnUSb3JuSvpUBKxscbyBjiOpOTq8jcdRsx5/IndXw3VgJV6iO1+6jl4gjV
pWouViO6ih9ZmybSPkhaqyNUxVXpV5cYU+Xx5sQTfKystDLipmqaMhxIcgvplLqF/LWZzIS5Pvwbq
OvrSlNHVEYchCEIQISICSZJijwu5OrRQHDyUpaF0y///p6FEDCCDFsuW7YFoVEFESTOBAACLgLOrA
AAAAggUAAAAtAAAAFJESEkNAAAAChoKDUdOUIk='

open("flag.png", mode='wb').write(base64.b64decode(s)[::-1])
```

得到以下图片，顺时针旋转 180，即可得到 Flag



最终得到 Flag，`hgame{tenchi_souzou_dezain_bu}`

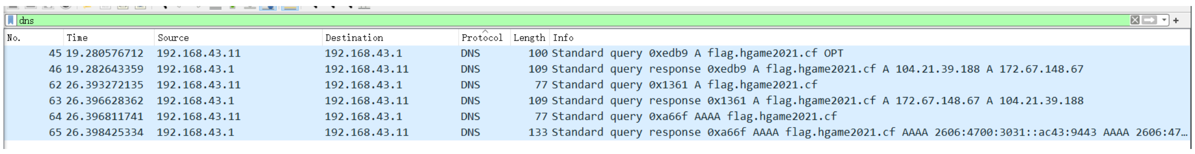
DNS

描述

A significant invention.

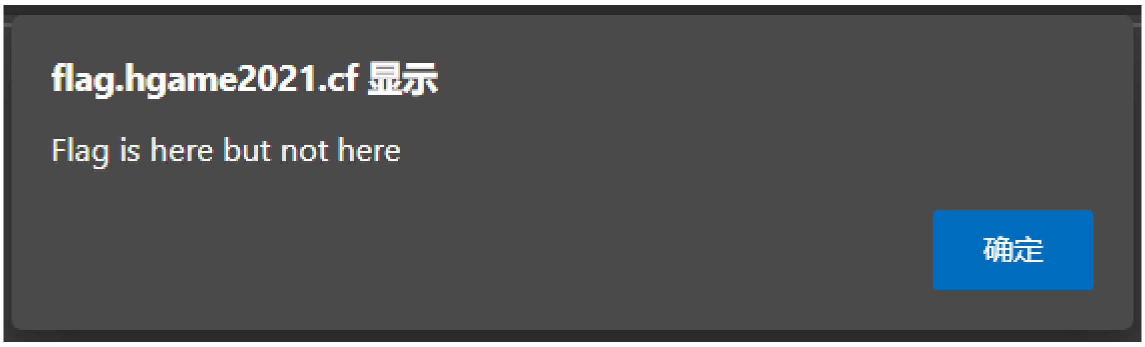
解题思路

用 `wiresharp` 打开 `.pcapng`，筛选包含 `DNS` 的记录，我们得到一个网址 `flag.hgame2021.cf`



```
dns
No.   Time          Source          Destination     Protocol Length Info
   45 19.280576712  192.168.43.11   192.168.43.1    DNS        100 Standard query 0xedb9 A flag.hgame2021.cf OPT
   46 19.282643359  192.168.43.1    192.168.43.11   DNS        109 Standard query response 0xedb9 A flag.hgame2021.cf A 104.21.39.188 A 172.67.148.67
   62 26.393272135  192.168.43.11   192.168.43.1    DNS         77 Standard query 0x1361 A flag.hgame2021.cf
   63 26.396628362  192.168.43.1    192.168.43.11   DNS        109 Standard query response 0x1361 A flag.hgame2021.cf A 172.67.148.67 A 104.21.39.188
   64 26.396811741  192.168.43.11   192.168.43.1    DNS         77 Standard query 0xa66f AAAA flag.hgame2021.cf
   65 26.398425334  192.168.43.1    192.168.43.11   DNS        133 Standard query response 0xa66f AAAA flag.hgame2021.cf AAAA 2606:4700:3031::ac43:9443 AAAA 2606:47…
```

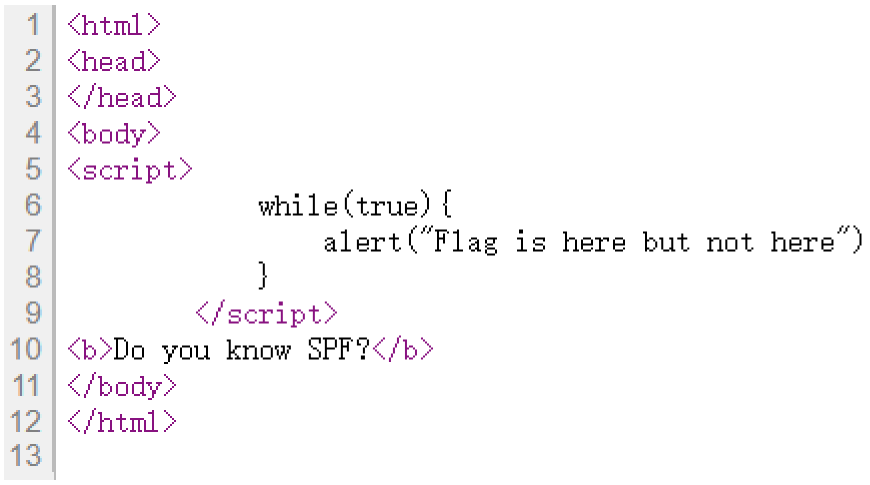
访问后得到一个关不掉的弹框，写着 `Flag在这但不在这`



flag.hgame2021.cf 显示

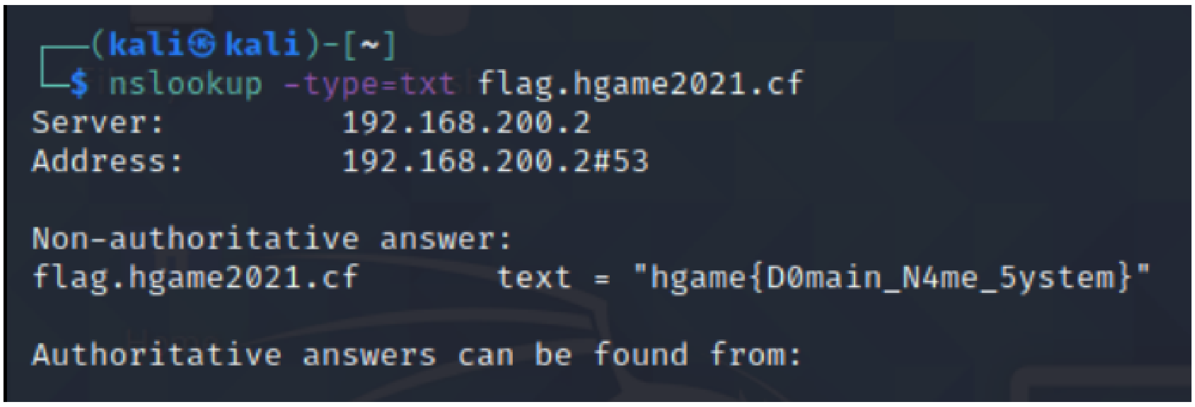
Flag is here but not here

使用 `view-source` 查看网页源代码，获得关键字 `SPF`



```
<html>
<head>
</head>
<body>
<script>
            while(true){
                alert("Flag is here but not here")
            }
        </script>
<b>Do you know SPF?</b>
</body>
</html>

```

百度得知使用 `nslookup -q=txt url` 可以查看邮件服务器的 `SPF` 配置，执行后看到 Flag



```
┌──(kali㉿kali)-[~]
└─$ nslookup -type=txt flag.hgame2021.cf
Server:         192.168.200.2
Address:        192.168.200.2#53

Non-authoritative answer:
flag.hgame2021.cf       text = "hgame{D0main_N4me_5ystem}"

Authoritative answers can be found from:
```

最终得到 Flag，`hgame{D0main_N4me_5ystem}`