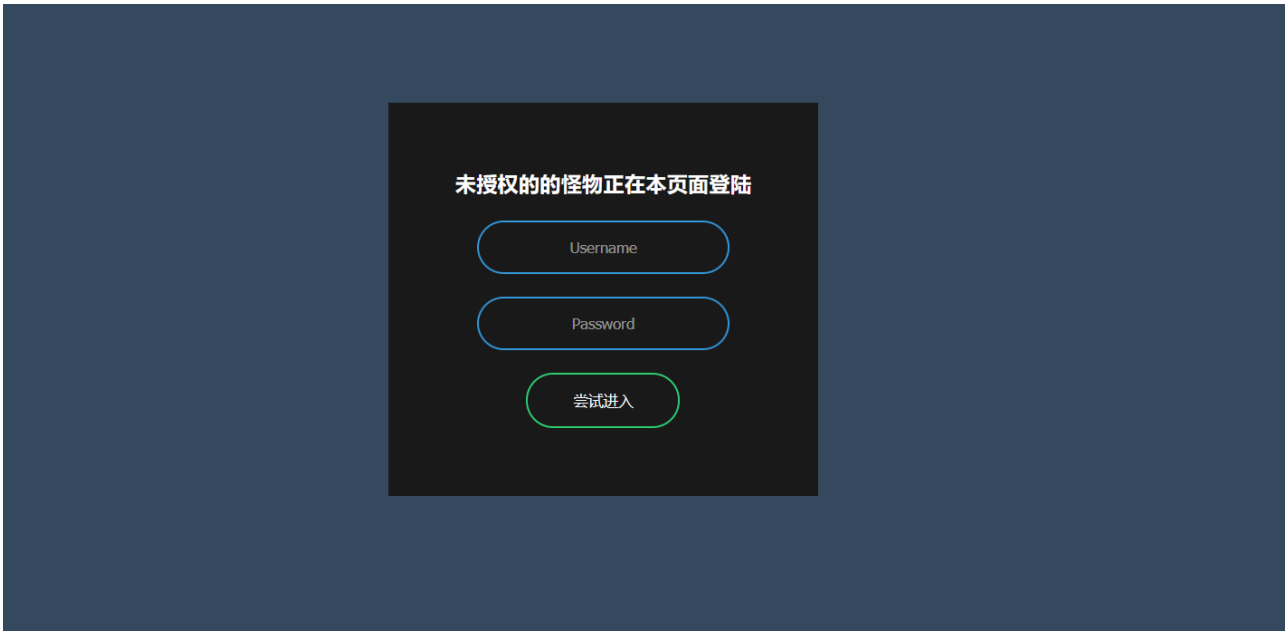
HGAME-Week2-Writeup

Web

1.LazyDogR4U

首先打开题目地址

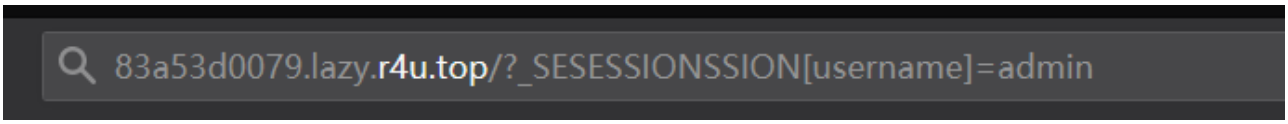


根据题目给的hint，在地址栏后面加上 **/www.zip** 获得网页源码

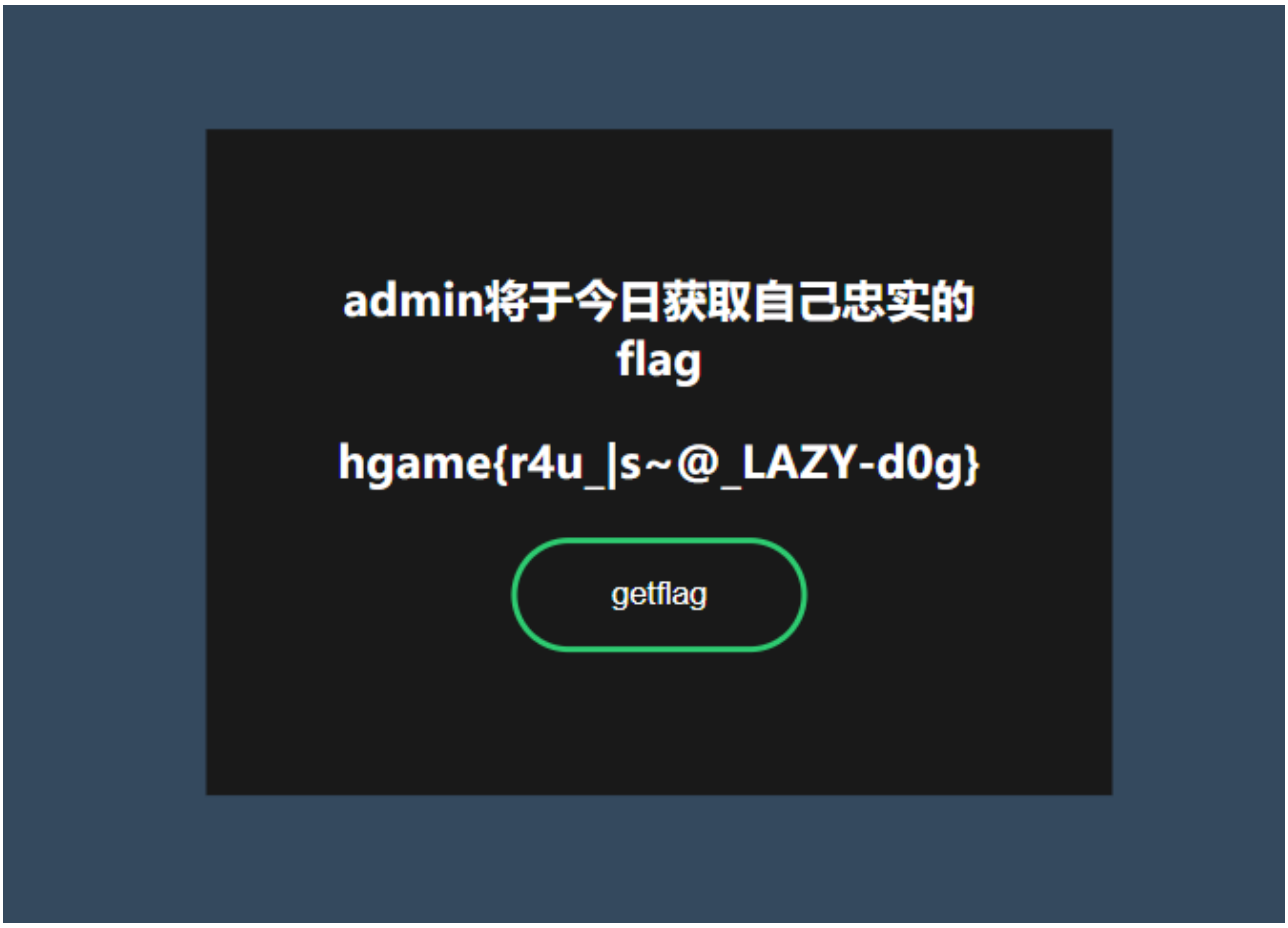
接着打开 lazy.php 和 flag.php 这两个php文件，再根据**变量覆盖**的提示找到漏洞点

```php
<?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;
    }
}


// 自动加载类，这样我也能少打字力，芜湖~
function auto($class_name){
    require_once $class_name . ".php";
}
spl_autoload_register('auto');
```

这里划线处的两个$符号容易导致变量覆盖，那么根据 flag.php 中得到flag的条件就可以进行构造，相当于：　　$_SESSSION[username]=admin



双写SESSION是因为上面str_replace的过滤



得到flag

2.Post to zuckonit

Attention: you can freely **post** your thoughts to this page. But this online editor is vulnerable to attack, so you can write down **XSS** sentences and **submit** them to bot backend, and CAPTCHA is necessary.

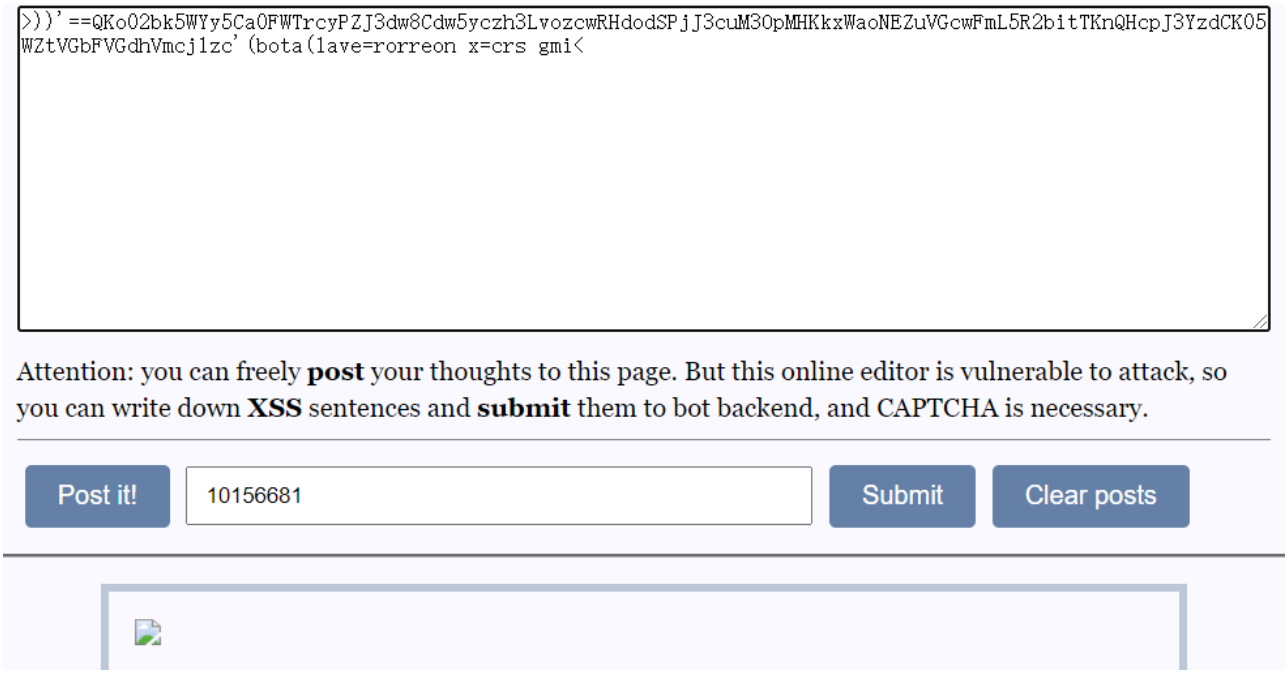
点开后发现是留言板，题目也很明确的指出是**xss**，那么就先尝试输入 ，，

输入后发现不管大小写 都会被过滤掉，小写的 会被过滤，则不会被过滤

之后尝试输入，让错误的图片导致弹窗

```
>)1(trela = rorreon 1=crs gmi<
```

输出了这么个东西，把这个输入再 post 就会弹窗，成功弹窗之后就是找一个 xss 平台来接受 cookie 信息

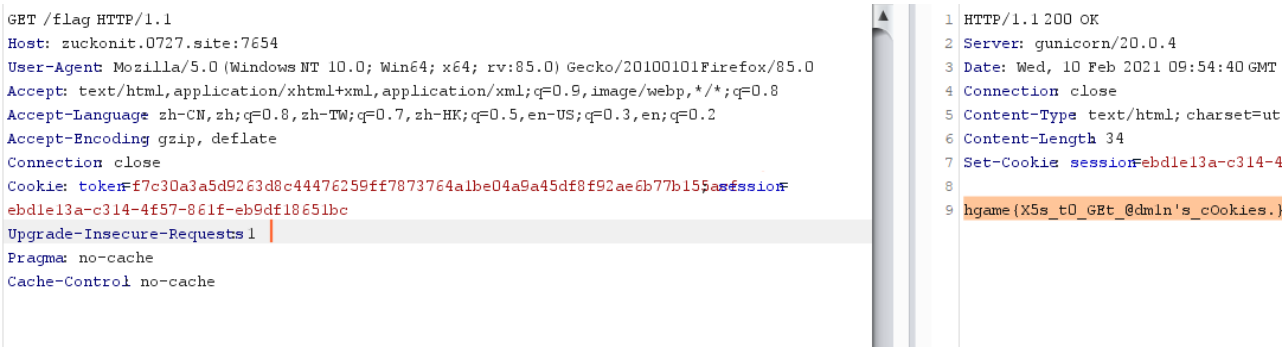


根据之前的规律 post 平台给的语句，并用脚本得出 md5 验证码提交，接收到 cookie 信息

脚本：

```
import hashlib
def md5(s):
    return hashlib.md5(s.encode()).hexdigest()
def get_code():
    code = 'XXXXXX'
    for i in range(10000000, 99999999):    #8位md5
        if md5(str(i)).startswith(code):
            return str(i)

print(get_code())
```

接着就伪造 admin cookie 得到flag



Re

1.ezApk

安卓逆向，用 jadx 反编译为 java，找到 onClick 事件

```
public final void onClick(View view) {
    MainActivity mainActivity;
    String str;
    MainActivity mainActivity2 = this.f1427a;
    T t = this.f1428b.f1325a;
    c.d.b.a.b(t, "pass");
    MainActivity.s(mainActivity2, t.getText().toString());
    c.d.b.a.b(this.f1427a.getApplicationContext().getString(R.string.flag), "applicationContext.getString(R.string.flag)");
    MainActivity mainActivity3 = this.f1427a;
    T t2 = this.f1428b.f1325a;
    c.d.b.a.b(t2, "pass");
    if (c.d.b.a.a(MainActivity.s(mainActivity3, t2.getText().toString()), this.f1427a.getApplicationContext().getString(R.string.flag))) {
        mainActivity = this.f1427a;
        str = "Good Job!";
    } else {
        mainActivity = this.f1427a;
        str = "Again?";
    }
    Toast.makeText(mainActivity, str, 1).show();
}
```

这条判断语句是关键，在资源文件中找到加密后的 flag 和 key

```
<string name="flag">EEB23sI1Wd9Gvhvk1sgWyQZhjilnYwCi5au1guzOaIg5dMAj9qPA7lnIyVoPSdRY</string>
<string name="hide_bottom_view_on_scroll_behavior">com.google.android.material.behavior.HideBottom\
<string name="icon_content_description">Dialog Icon</string>
<string name="item_view_role_description">Tab</string>
<string name="key">A_HIDDEN_KEY</string>
<string name="material_slider_range_end">Range end.</string>
```

s函数用来加密输入的字符串，转到s函数分析

```
CharSequence charSequence;
String string = mainActivity.getApplicationContext().getString(R.string.key);
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);
Charset charset = c.g.a.f1337a;
if (str != null) {
    byte[] bytes = str.getBytes(charset);
    c.d.b.a.b(bytes, "(this as java.lang.String).getBytes(charset)");
    byte[] doFinal = instance.doFinal(bytes);
    int i = 0;
    String encodeToString = Base64.encodeToString(doFinal, 0);
    c.d.b.a.b(encodeToString, "encodeToString(byteResult, Base64.DEFAULT)");
    List asList = Arrays.asList("\n");
    c.d.b.a.b(asList, "ArraysUtilJVM.asList(this)");
    b bVar = new b(new c.g.b(encodeToString, 0, 0, new h(asList, false)), new i(encodeToString));
    StringBuilder sb = new StringBuilder();
    sb.append((CharSequence) "");
    Iterator it = bVar.iterator();
```

仔细分析后得知是**AES CBC模式**加密，密钥是之前的key经过**sha-256**加密得到的，iv向量则是key经过**MD5**加密得到的，最后将加密结果再进行一次 **base64** 加密得到之前的flag

尝试用在线网站解密，但是网站一般只支持 16 位 iv 值解密，没有别的办法只能东拼西凑写出 java 脚本来解密

```
package com.company;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.security.MessageDigest;
```

```
import java.security.NoSuchAlgorithmException;
import java.util.Base64;

public class Main {

    //md5,sha-256加密
    public static byte[] t(String str, String str2) {
        MessageDigest instance = null;
        try {
            instance = MessageDigest.getInstance(str);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        byte[] bytes = str2.getBytes();
        byte[] digest = instance.digest(bytes);
        return digest;
    }


    //解密
    public static String desEncrypt(String data, byte[] key) {

        byte[] ivString = t("MD5","A_HIDDEN_KEY"); //iv向量
        byte[] iv = ivString;

        try {
            byte[] encryp = Base64.getDecoder().decode(data);
            Cipher cipher =
Cipher.getInstance("AES/CBC/PKCS7Padding");
            SecretKeySpec keySpec = new SecretKeySpec(key, "AES");
            IvParameterSpec ivSpec = new IvParameterSpec(iv);
            cipher.init(Cipher.DECRYPT_MODE, keySpec, ivSpec);
            byte[] original = cipher.doFinal(encryp);
            return new String(original);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    public static void main(String[] args) {
```

```
        String data =
"EEB23sI1Wd9Gvhvk1sgWyQZhjilnYwCi5au1guzOaIg5dMAj9qPA7lnIyVoPSdRY";
    //密文
        byte[] key =t("SHA-256","A_HIDDEN_KEY");  //密钥
        String desencrypt = desEncrypt(data, key);
        System.out.println("解密后:"+desencrypt);
    }
}
```

因为要用到的是AES 256位解密，而 java 本身只支持 128位 AES 解密，所以就要做其他的一些配置才能解密

```
解密后:hgame{jUst_A_3z4pp_write_in_k07l1n}
```

2.helloRe2

拖进 IDA 分析，乍一看结构有点复杂，password1 的判断在后半部分，而password2 的判断在前半部分，先分析 password1

```
if ( strlen(xmmword_4043A0) == 16
  && (unsigned __int16)_mm_movemask_epi8(_mm_cmpeq_epi8(_mm_load_si128((const __m128i *)xmmword_4043A0), (__m128i)xmmword_4030F0)) == 0xFFFF )// 逆序
```

关键部分是这条判断，判断输入的password长度和 password 经过一系列操作后的结果，用 ollydbg 调试并不断猜测，得知这么一系列操作就是判断输入的字符串是否是给出的密文的逆序

之后再分析 pwd2

```
77 6F 72 64 20 31 20 63  6F 72 72 65 63 74 20 21  word·1·correct·!
00 00 4F 00 62 00 6A 00  65 00 63 00 74 00 4C 00  ..O.b.j.e.c.t.L.
65 00 6E 00 67 00 74 00  68 00 00 00 42 00 6C 00  e.n.g.t.h...B.l.
6F 00 63 00 6B 00 4C 00  65 00 6E 00 67 00 74 00  o.c.k.L.e.n.g.t.
68 00 00 00 43 00 68 00  61 00 69 00 6E 00 69 00  h...C.h.a.i.n.i.
6E 00 67 00 4D 00 6F 00  64 00 65 00 00 00 4F 00  n.g.M.o.d.e...O.
70 00 61 00 71 00 75 00  65 00 4B 00 65 00 79 00  p.a.q.u.e.K.e.y.
42 00 6C 00 6F 00 62 00  00 00 41 00 45 00 53 00  B.l.o.b...A.E.S.
00 00 43 00 68 00 61 00  69 00 6E 00 69 00 6E 00  ..C.h.a.i.n.i.n.
67 00 4D 00 6F 00 64 00  65 00 43 00 42 00 43 00  g.M.o.d.e.C.B.C.
00 00 2A 00 2A 00 2A 00  2A 00 20 00 6D 00 65 00  ..*.*.*.*.·.m.e.
6D 00 6F 00 72 00 79 00  20 00 61 00 6C 00 6C 00  m.o.r.y.·.a.l.l.
```

应该又是AES CBC 模式加密，找到 iv 和 密钥，密钥是pwd1经过异或得到，密文是已经给出密文的逆序

```
memcpy(v14, &unk_40312C, *(size_t *)v47);// iv

{
  v30->m128i_i8[1] ^= 1u;
  v30->m128i_i8[2] ^= 2u;
  v30->m128i_i8[3] ^= 3u;
  v30->m128i_i8[4] ^= 4u;
  v30->m128i_i8[5] ^= 5u;
  v30->m128i_i8[6] ^= 6u;
  v30->m128i_i8[7] ^= 7u;
  v30->m128i_i8[8] ^= 8u;
  v30->m128i_i8[9] ^= 9u;
  v30->m128i_i8[10] ^= 0xAu;
  v30->m128i_i8[11] ^= 0xBu;
  v30->m128i_i8[12] ^= 0xCu;
  v30->m128i_i8[13] ^= 0xDu;
  v30->m128i_i8[14] ^= 0xEu;
  v30->m128i_i8[15] ^= 0xFu;
}
```

解密就用上一题的脚本

```
解密后:7a4ad6c5671fb313
```

```
if ( BCryptEncrypt(
       (BCRYPT_KEY_HANDLE)phKey.cb,
       v20,
       0x10u,
       0,
       v15,                // iv v15
       *(ULONG *)v47,
       0,
       0,
       &cbOutput,
       1u) >= 0 )
{
  v45 = cbOutput;
  v22 = GetProcessHeap();
  v23 = (UCHAR *)HeapAlloc(v22, 0, v45);
  if ( v23 )
  {
    v7 = (__m128i *)v23;
    if ( BCryptEncrypt(
           (BCRYPT_KEY_HANDLE)phKey.cb,
```

这里看着像是两次加密，但其实只有一次

```
Hello Re Player
input password 1:
2b0c5e6a3a20b189
password 1 correct !
input password 2:
7a4ad6c5671fb313
Good!
the flag is : hgame{2b0c5e6a3a20b189_7a4ad6c5671fb313}
```

3.fake_debugger beta

nc题目地址，发现只有在 ebx 经过一次转换后与 eax 相等才能继续调试，之后不断调试发现

```
--------------INFO--------------

eax: 21
ebx: 21
ecx: 12
zf: 1
--------------INFO--------------

eax: 71
ebx: 48
ecx: 13
zf: 0
--------------INFO--------------

eax: 71
ebx: 71
ecx: 13
zf: 1
--------------INFO--------------

Traceback (most recent call last):
  File "./challenge.py", line 36, in <module>
    dic['eax'] = ord(input[index]) ^ arr[index]
IndexError: string index out of range
```

转换前的 ebx 与 eax 异或后就是 flag 中的字符，那就一个个调试下来得到 flag

Crypto

1.gcd or more?

```
from libnum import *
from secret import FLAG

p = 852285650211289018533149345831290834419890452250225412985505704493898396090l9
q = 111614714641364911312915294479850549131835378046002423977989457843071188836271
n = p * q

cipher = pow(s2n(FLAG), 2, n)
print(cipher)
# 766500368283066645619389449101598964164785482664717787314198410720209908147598482
```

看着像是RSA，但是e = 2。百度后查到低指数爆破，之后又看到Rabin算法，特征是e = 2，那么就用这个脚本解题

```
import gmpy2
import libnum

e = 2
p = 852285650211289018533149345831290834419890452250225412985505704493898396090l9
q = 111614714641364911312915294479850549131835378046002423977989457843071188836271
n = p * q
c = 766500368283066645619389449101598964164785482664717787314198410720209908147598482780600728783047289961681808090727660674446745344590892305497539362350953 9

#c= int(open('./flag.enc','rb').read().encode('hex'),16)
mp = pow(c, (p + 1) // 4, p)
mq = pow(c, (q + 1) // 4, q)
yp = gmpy2.invert(p, q)
yq = gmpy2.invert(q, p)
r = (yp * p * mq + yq * q * mp) % n
rr = n - r
s = (yp * p * mq - yq * q * mp) % n
ss = n - s
```

```
print(libnum.n2s(int(r)))
print(libnum.n2s(int(rr)))
print(libnum.n2s(int(s)))
print(libnum.n2s(int(ss)))
```

2.WhitegiveRSA

WhitegiveRSA[已完成]

描述

N = 882564595536224140639625987659416029426239230804614613279163

e = 65537

c = 747831491353896780365654517748216624798517769637260742155527

题目地址 https://www.baidu.com

基准分数 150

当前分数 150

完成人数 177

已知N，那就拿到在线网站里分解出大素数 q 和 p，剩下来都已知就用脚本解密

```
>>> import gmpy2
>>> import binascii
>>>
>>> n = 882564595536224140639625987659416029426239230804614613279163
>>> p = gmpy2.mpz(857504083339712752489993810777)
>>> q = gmpy2.mpz(1029224947942998075080348647219)
>>> e = gmpy2.mpz(65537)
>>> phi_n = (p - 1) * (q - 1)
>>> d = gmpy2.invert(e, phi_n)
>>> c = gmpy2.mpz(747831491353896780365654517748216624798517769637260742155527)
>>>
>>> m = pow(c, d, n)
>>> print("10 : %s" % m)
10 : 255997447193686133225069560189674983138058671722772982207 7
>>> m_hex = hex(m)[2:]
>>> print("16: %s" % (m_hex))
16: 6867616d657b7730777e794f555f6b4e6f572b523540217d
>>> m_hex = m_hex.strip('\n')
>>> ''.join(m_hex)
'6867616d657b7730777e794f555f6b4e6f572b523540217d'
>>> print("ascii:\n%s" % (binascii.a2b_hex(m_hex).decode("utf8"),))
ascii:
hgame{w0w~yOU_kNoW+R5@!}
>>>
```

Misc

1.Tools

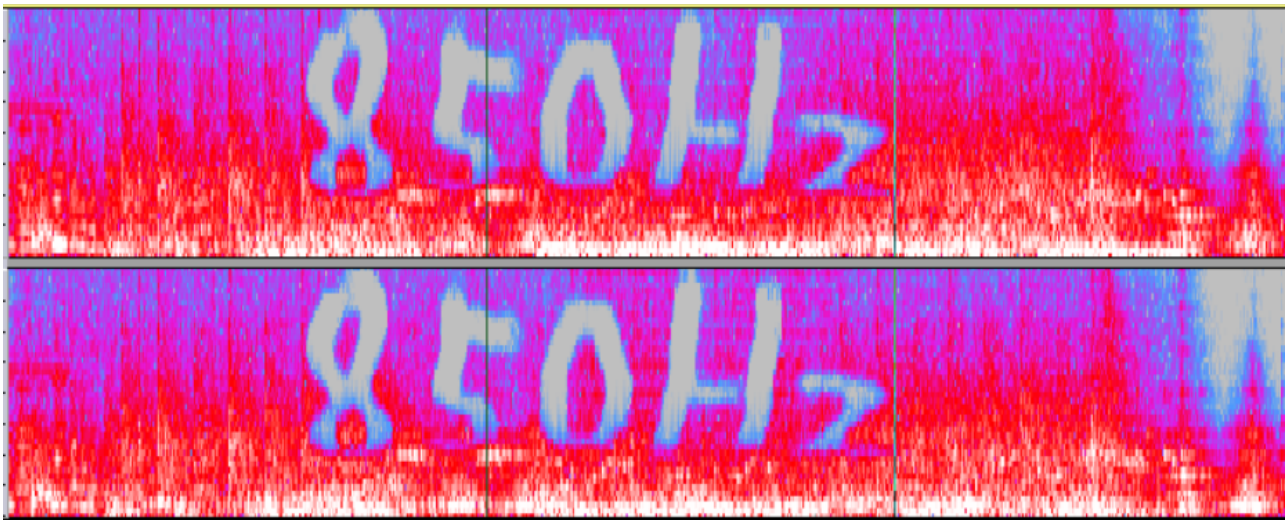
下载解压得到一个压缩包和一张图片，根据题目的 tools 和压缩包名 f5 尝试用 f5 工具破解压缩包密码，破解所需的密码在图片的详细信息中

打开加密压缩包后发现一部分二维码和压缩包，接着就是像之前那样找到压缩包名里的工具解密，最终集齐4张二维码碎片，用win10自带的画图工具拼合，扫码得flag

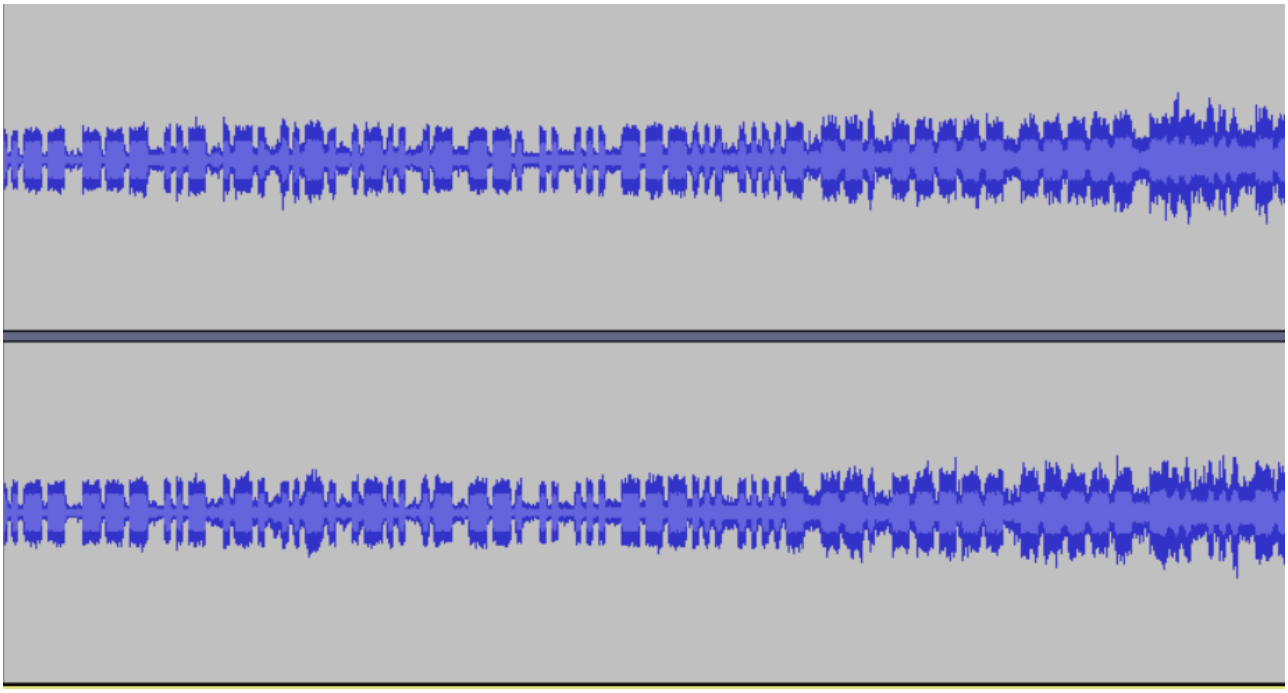
2.Telegraph：1601 6639 3459 3134 0892

题目为音频隐写，又提到听着反胃，于是猜测是**摩斯电码**。听了之后，中间有一段嘀嗒声，这一段就是需要摩斯解密的部分，接着把音频拖到 **Audacity** 里面分析

首先看一下频谱图



得到**850hz**的提示，之后尝试翻译摩斯电码，发现中间有一段无法辨认就想到要滤波，用的是上面的提示



窄的是. 宽的是- ，翻译后解密就得到flag

英文字母：

YOURFLAGIS4GOODSONGBUTNOT4GOODMAN039310KI

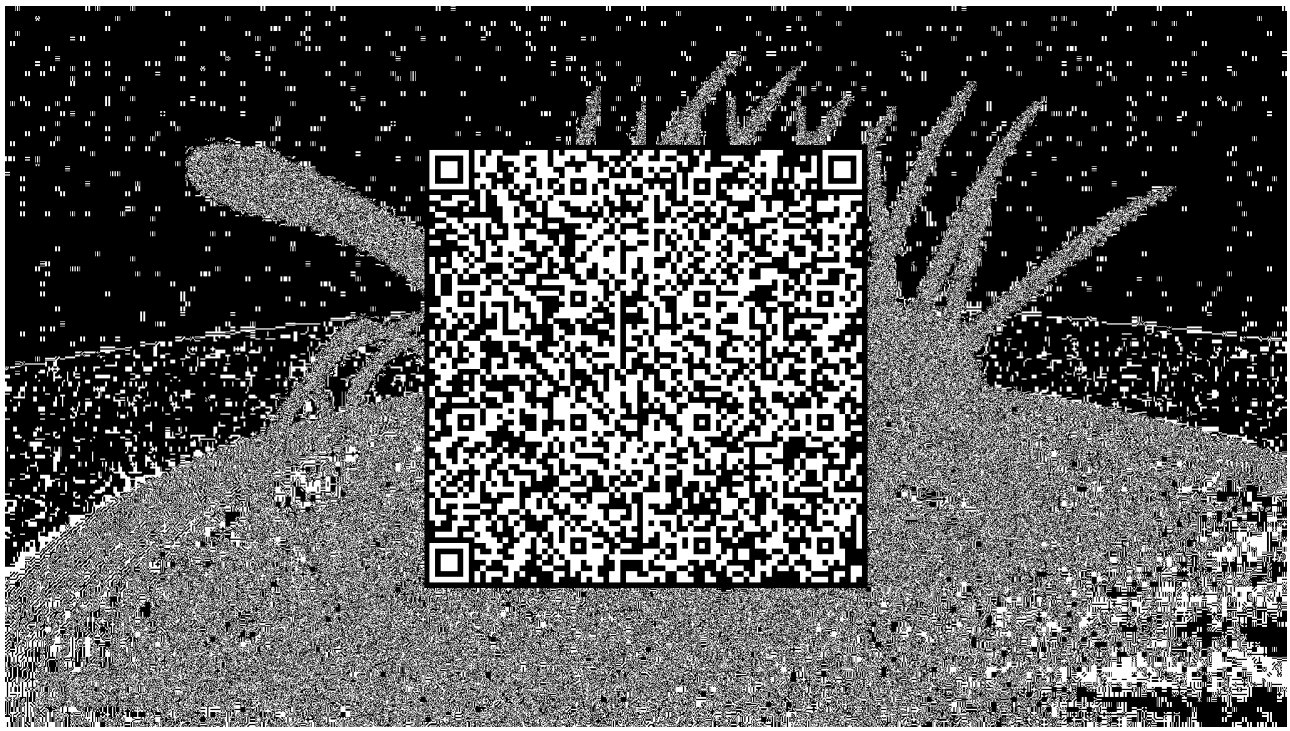
转换为摩斯电码 清除 生成摩斯代码的分隔方式： 空格分隔 单斜杠/分隔

摩斯电码：（格式要求：可用空格或单斜杠/来分隔摩斯电码，但只可用一种，不可混用）

```
-. --/---/.. -/. -. /.. -. /. -.. /. -/--. /.. /... /---. .. /.... -/--. /-----/----
-/-.. /... /-----/-. /--. /-... /.. -/-/-. /-----/-/.... -/--. /-----/----
-/-.. /--/. -/-. /-----/... --/----. /... --/. ----/-----/-. -/../
```

3.Hallucigenia

只给了一张图，那么就先用 Stegsolve 左右调一下得到二维码



扫码得到一串 base64 解码发现最后是 GNP，根据题目中“我们不仅弄错了他的上下，还颠倒了它的左右。”逆序处理后粘贴到winhex保存为图片

发现又是上下左右颠倒，倒回来就得flag

4.DNS

流量包分析，筛选 dns 得到一串域名并访问

抓包得

```
NEL: {"max_age":604800,"report_to":"cf-nel"}
Server: cloudflare
CF-RAY: 61d9e0110866e502-LAX

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

查一下 SPF 知道是一种 **dns 记录**，那就用在线网站查一下dns记录，得到flag

总结：

这周比较摸鱼，web 能做出两道是因为给了具体的考点，参照往年 writeup 一点点做下来。 re 做了很久，因为之前没有接触过 apk 逆向，对 aes 加密也不太了解，最后能全部做出来真是太好了，逆向的时候果然是要抓重点，解完题后才发现其实挺简单的。 密码学是真不太行，这周 pwn 也基本没看，Misc 倒是做的挺顺畅的，下周要试着把之前的知识捡起来了