




信安实验室CTF writeup

原创

[开心星人](#)  于 2021-11-16 14:11:34 发布  247  收藏 1

文章标签: [安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_55675216/article/details/121352171

版权

文章目录

- 1、白给签到
- 2、Welcome
- 3、Get
- 4、Post
- 5、滴滴滴
- 6、每逢佳节
- 7、Bacon
- 8、古典变奏
- 9、affine
- 10、affine-revenge
- 11、Random_encrypt
- 12、easy_re
- 13、re1
- 14、ez_xor
- 15、maze
- 16、easy_php
- 17、easy_bypass
- 18、Autumn
- 19、easy_Cookie
- 20、[白给] 连上就给flag
- 21、小兔子
- 22、我在城楼观山景
- 23、Stream
- 24、如来の兽
- 25、PDF
- 26、QR code
- 27、Trytouse
- 28、Boringboss
- 29、esay-RSA

1、白给签到

白给&签到

分值: 10 已解答

2012080115-葛健

不排名-聂豪杰

2012080038-于晨彤

flag{SongFenTi}

提交

CSDN @开心星人

直接填入flag即可

2、Welcome

Welcome

分值: 30 已解答

2012080040-张澳韩

2012080008-陈章绮

欢迎来到HenuCTF! 你这flag挺能藏啊? 题目地址: <http://39.107.125.199:40011/>

提交

CSDN @开心星人

进入题目链接地址

右键查看源代码或者Fn+F12

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Welcome21</title>
</head>
<body>
  <!--flag{Thank_you_Meiyu}-->
  <div align="middle">
    
    <h1>Welcome to Henu CTF :)</h1>
  </div>
</body>
</html>
```

CSDN @开心星人

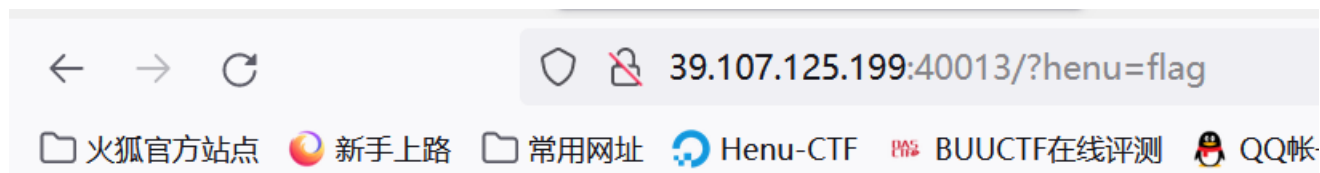
获得flag

3、Get

```
<?php
if ($_GET["henu"] == "flag") {
    echo($flag);
}
?>
```

使用Get传入参数henu

使得参数henu=flag即可回显flag



```
<?php
if ($_GET["henu"] == "flag") {
    echo($flag);
}
?>
flag{flag_is_flag}
```

CSDN @开心星人

4、Post

```
<?php
if ($_POST["henu"] == "fl@g") {
    echo($flag);
}
?>
```

和刚刚的Get几乎没有什么区别(注意这里是henu=fl@g，一般可以直接复制)，不过这一次是Post传入参数

Post传参需要借助插件HackerBar或其他类似的插件，我这里使用的Max HackerBar

右键点击“检查”，选中Max HackerBar插件

```
<?php
if ($_POST["henu"] == "fl@g") {
    echo($flag);
}
?>
flag{have_fun_ctfers}
```

查看器 控制台 调试器 网络 样式编辑器 性能 内存 存储

SQL Error Based WAF XSS LFI LDAP VARIABLES Byp

Load URL http://39.107.125.199:40015/

Split URL

Execution

Post Data Referrer REVERSE HEX

SHA256 ROT13

Post Data

henu=fl@g

CSDN @开心星人

先Load URL，勾选上Post Data，这时候下面会出现Post Data输入框，将要传入的参数写上henu=fl@g，最后点击Execution（执行），即可获得flag

5、滴滴滴

题目描述： .--/./.-./-.-/---/--/--/---/-../.-./-.-/---/---
注意：提交形式为flag{xxx}

打开附件，可以看到很明显是摩斯电码

复制摩斯电码

打开一个在线摩斯电码转换器

英文字母:

WELCOMTOCRYPTO

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)

. --/. /. -. /-. -. /---/--/--/--/-. -. /-. -. /-. --/. --. /-/--

CSDN @开心星人

又根据题目描述是小写, 所以将flag小写即可

6、每逢佳节

每逢佳节

分值: 100 已解答

2012080115-葛健 2012080097-张开斌 2012080040-张澳韩

解密字符串: ZmxhZ3t0aGVfZmxhZ19vZl90aGlzX3N0cmIuZ30=

提交

CSDN @开心星人

解密字符串以"="号结尾, 由大写英文字母、小写英文字母和数字组成

base64的特征

- 1.标准base64只有64个字符(英文大小写、数字和+、/)以及用作后缀等号;
- 2.base64是把3个字节变成4个可打印字符, 所以base64编码后的字符串一定能被4整除(不算用作后缀的等号);
- 3.等号一定用作后缀, 且数目一定是0个、1个或2个。这是因为如果原文长度不能被3整除, base64要在后面添加\0凑齐3n位。为了正确还原, 添加了几个\0就加上几个等号。显然添加等号的数目只能是0、1或2;

所以推测是base64加密

```
flag(the_flag_of_this_string)
```

```
ZmxhZ3t0aGVfZmxhZ19vZl90aGlzX3N0cmlyZ30=
```

☐ 多行

Base64编码

Base64解码

CSDN@开心星人

7、Bacon

王果 王果 王木 王木 王果 王果 王木 王木 王果 王果 王木 王木 王果 王果 王果 王果 王木 王木

得到一个字符串，又由题目名称得知这题使用的是Bacon密码

分析字符串可知是由“王果”和“王木”组成

可以将“王果”当作A，“王木”当作B

```
str1="王果 王果 王木 王木 王果 王果 王木 王木 王果 王果 王木 王木 王果 王果 王果 王果 王木 王木"
str2=""
for i in range(len(str1)):
    if str1[i]=="果":
        str2+="A"
    elif str1[i]=="木":
        str2+="B"
print(str2)
```

得到AABBAABBBAABBBAAAABB

（一般如果不行把A和B互换位置，因为不知道王果和王木哪个代表A和B）

打开在线Bacon解密工具

培根密码

Baconian Cipher

AABBAABBBAAABBBAAAABB

加密

good

CSDN @开心星人

即可获得flag

8、古典变奏

古典变奏

分值: 150 已解答

2012080040-张澳韩

2012080115-葛健

2112080124

小明要去猪圈，但是他没有钥匙，于是他上到6楼找到凯撒，并向凯撒要钥匙，凯撒指着楼下的栅栏说：钥匙就在栅栏下面。注意：提交字母均为小写，形式为flag{xxx}。

附件.jpg

提交

CSDN @开心星人

W ^ > C F J F O ^ W O O F < W F J A W L W > < O

打开附件发现一串奇怪的字符

由经验的积累或者题目的提示“小明要去猪圈”得知这是一个猪圈密码



加密的内容:

W^ >C F J F O ^ W O O F < W F < W F J F W L L W > X O

解密的内容:

kztdrjiezkenrukrukraqklktue

回退 清空

CSDN @开心星人

猪圈密码在线解密即可得到一个字符串

由题目提示“他上6楼找凯撒”推测这是一个凯撒密码，经过移六位得到的

kztdrjiezkenrukrukraqklktue

位移 6 加密 解密

etnxdcyteyhloelolukefenoy

CSDN @开心星人

凯撒密码在线解密得到一个字符串

由题目提示“钥匙就在栅栏下面”推测是栅栏密码

etnxldcyteyhloeloelukefenoy

每组字数

excellentlyooufondtheekey
CSDN @开心星人

但是题目没有给出每组字数，需要自己不断去尝试，最后得到一个有含义的字符串，即为flag

9、affine

```
a,b,m=generate_key()
print(a,b,m)
#35 9 52
cipher=encrypt(flag,a,b,m)
assert flag==decrypt(cipher,a,b,m)
print(cipher)
#UTWH{YUDp_Dp_j_qTGr_SjpDB_TWBGroYDFW}
```

CSDN @开心星人

```
assert flag==decrypt(cipher,a,b,m)
```

python断言是声明其布尔值必须为真的判定

所以flag一定等于cipher使用a,b,m解密后得到的字符串

分析可知加密和解密都是使用的a,b,m，这题的公钥和私钥都是a,b,m，且这题的a,b,m已经明确给出

cipher是flag进行加密后得到的，且cipher已知

那么现在就可以直接通过cipher反解出flag即可

```
50 a, b, m = 35, 9, 52
51 cipher = 'UTWH{YUDp_Dp_j_qTGr_SjpDB_TWBGroYDFW}'
52 flag = decrypt(cipher, a, b, m)
53 print(flag)
54
```

控制台

```
henu{this_is_a_very_basic_encryption}
程序运行结束
```

CSDN @开心星人

10、affine-revenge

```
a,b,m=generate_key()
#print(a,b,m)

cipher=encrypt(flag,a,b,m)
assert flag==decrypt(cipher,a,b,m)
print(cipher)

#ZovM{RZo_lIbIAoRobW_UW_Hobs_WAIFF_RZIR_Co_yIv_Rbs_IFF}
```

CSDN @开心星人

这一题和上面一题唯一的区别就是这一题的钥匙a,b,m没有明确给出，需要自己去找

```
50 cipher = 'ZovM{RZo_lIbIAoRobW_UW_Hobs_WAIFF_RZIR_Co_yIv_Rbs_IFF}'
51 for i in range(1000):
52     a, b, m = generate_key()
53     flag = decrypt(cipher, a, b, m)
54     if flag[0] == 'h' and flag[1] == 'e':
55         print(flag)
56
```

控制台

```
henu{the_parameters_is_very_small_that_we_can_try_all}
henu{the_parameters_is_very_small_that_we_can_try_all}
程序运行结束
```

CSDN @开心星人

这里我直接使用爆破

11、Random_encrypt

```
import base64
import random
from secret import flag

def Random_Encrypt(msg):
    for _ in range(10):
        r=random.randint(1,100)
        if r % 3 == 0:
            msg = base64.b64encode(msg)
        elif r % 3 == 1:
            msg = base64.b32encode(msg)
        elif r % 3 == 2:
            msg = base64.b16encode(msg)
    return msg

cipher=Random_Encrypt(flag)

print(cipher)
#R1JCREtOS1ZJRTJEUU5KUkdVM1RLTVpVSVUyRUVOS0JHUTNESU9CVUc0M1RNTkJUR1UyVE1RS1VJRTJ
```

CSDN @开心星人

可以看到这里最后的cipher是经过flag随机使用base64或base32或base16加密十次后得到的结果

(刚开始我任何可以在python代码中让随机解码十次,但因为base64、base32、base16编码组成不同

base64>base32>base16, base64的编码几乎不可以用base16和base32解码,但base16和base32编码却一定可以用base64解码,所以在进行解码时候要先判断它们可能是哪种编码,再随机使用该编码进行解码,发现这样难度系数增加),我直接使用人工判断,进行解码

Base64:

包含大写字母(A-Z),小写字母(a-z),数字(0-9)以及+;

Base32:

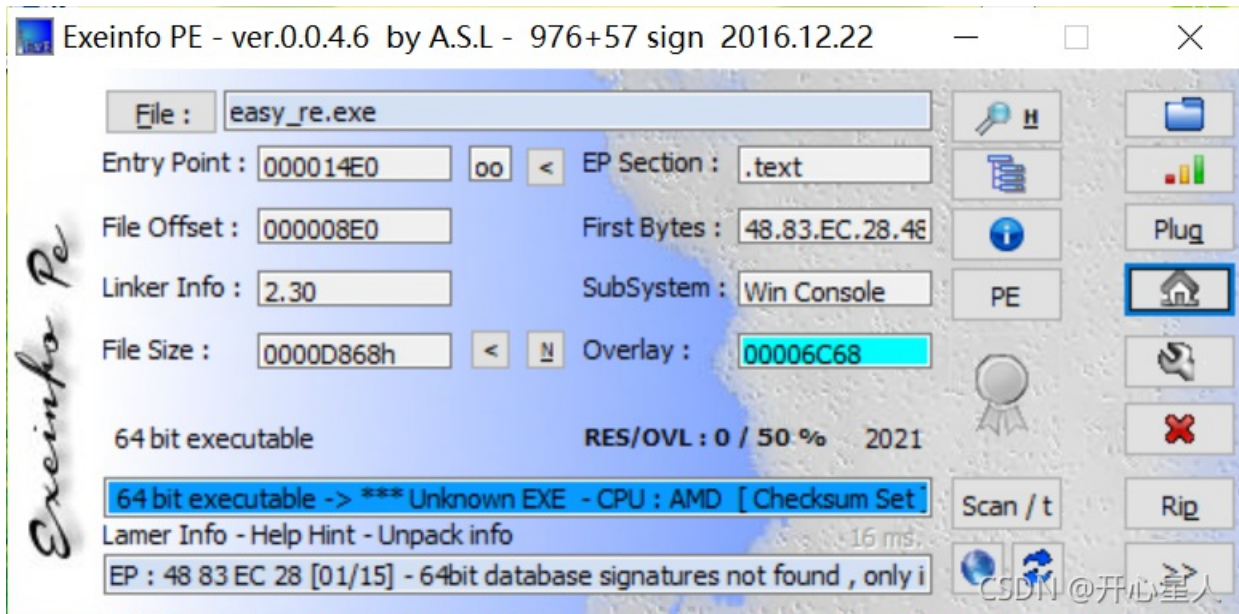
而Base32中只有大写字母(A-Z)和数字234567;

Base16:

而Base16就是16进制,他的范围是数字(0-9),字母(ABCDEF);

12、easy_re

拖入exeinfo中查看exe基本信息



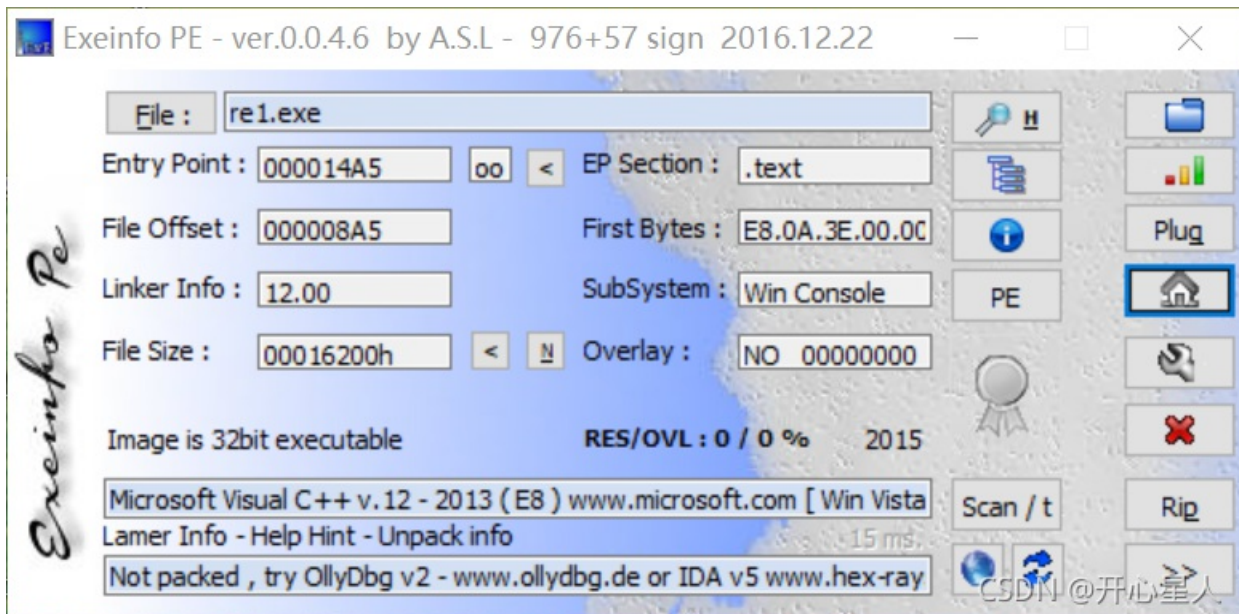
可以看到是64可执行文件

拖入64位ida中

```
.text:000000000401550 ; DATA XREF: .pdata:00000000040506C1o
.text:000000000401550
.text:000000000401550 var_8 = dword ptr -8
.text:000000000401550 var_4 = dword ptr -4
.text:000000000401550
.text:000000000401551 push rbp
.text:000000000401551 mov rbp, rsp
.text:000000000401554 sub rsp, 30h
.text:000000000401558 call __main
.text:00000000040155D mov [rbp+var_4], 5
.text:000000000401564 mov [rbp+var_8], 0
.text:00000000040156B lea rax, [rbp+var_8]
.text:00000000040156F mov rdx, rax
.text:000000000401572 lea rcx, Format ; "%d"
.text:000000000401579 call scanf
.text:00000000040157E mov eax, [rbp+var_8]
.text:000000000401581 cmp [rbp+var_4], eax
.text:000000000401584 jnz short loc_401594
.text:000000000401586 lea rcx, Buffer ; "henu{S0_9oOd,th1s_Is_EaSyRE!}"
.text:00000000040158D call puts
.text:000000000401592 jmp short loc_4015A0
.text:000000000401594 ; -----
.text:000000000401594 loc_401594: ; CODE XREF: main+341j
.text:000000000401594 lea rcx, aS0rryY0uCanT9e ; "s0rry,y0u_can't_9et_The_flag!"
.text:00000000040159B call puts
```

flag直接给出了

13、re1



拖入32位ida,找到main函数, F5查看伪代码

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v3; // eax
4     __m128i v5; // [esp+0h] [ebp-44h] BYREF
5     int v6; // [esp+1Ch] [ebp-28h]
6     char v7[32]; // [esp+20h] [ebp-24h] BYREF
7
8     v5 = _mm_loadu_si128((const __m128i *)&xmmword_413E34);
9     LOWORD(v6) = 0;
10    printf(aHenuctf, v5.m128i_i64[0], v5.m128i_i64[1], 2099344488, 0, 0, v6);
11    printf("这是一道很可爱很简单的逆向题呦\n");
12    printf("输入flag吧:");
13    scanf("%s", v7);
14    v3 = strcmp(v5.m128i_i8, v7);
15    if ( v3 )
16        v3 = v3 < 0 ? -1 : 1;
17    if ( v3 )
18        printf(aFlag_0);
19    else
20        printf(aFlagGet);
21    system("pause");
22    return 0;
23}

```

CSDN @开心星人

首先我们可以看到代码的最后的if-else语句会进行v3作为条件进行判断, 如果v3为真, 会printf(aFlag_0);否则 printf(aFlagGet);然后往上看到 v3 = strcmp(v5.m128i_i8, v7);将输入的v7和v5进行比较, 如果v5不等于v7直接就printf(aFlag_0);(这里就可以判断出来了真正的flag是aFlag_0,否则aFlagGet输出过于容易了)

所以v5就显得格外关键了，找跟v5相关的

```
v5 = _mm_loadu_si128((const __m128i *)&xmmword_413E34);
```

可以看到_mm_storeu_si128()，对其进行分析发现它类似于memset(),将xmmword_413E34的值赋值给v5，所以，我们可以得到正确的flag应该在xmmword_413E34中，然后，我们双击413E34进行跟进

```
.rdata:00413E2C a1Qnan          db '1#QNaN',0          ; DATA XREF: _$I10_OUTPUT:loc_40F13
.rdata:00413E33                align 4
.rdata:00413E34 xmmword_413E34  xmmword 3931725F6537615F7530597B756E6568h
.rdata:00413E34                ; DATA XREF: _main+10↑r
.rdata:00413E44 qword_413E44   dq 7D217468h          ; DATA XREF: _main+27↑r
.rdata:00413E4C ; const char aHenuctf[15]
.rdata:00413E4C aHenuctf       db '欢迎来到henuCTF'  ; DATA XREF: _main+1A↑r
.rdata:00413E5B                db 0D3h
```

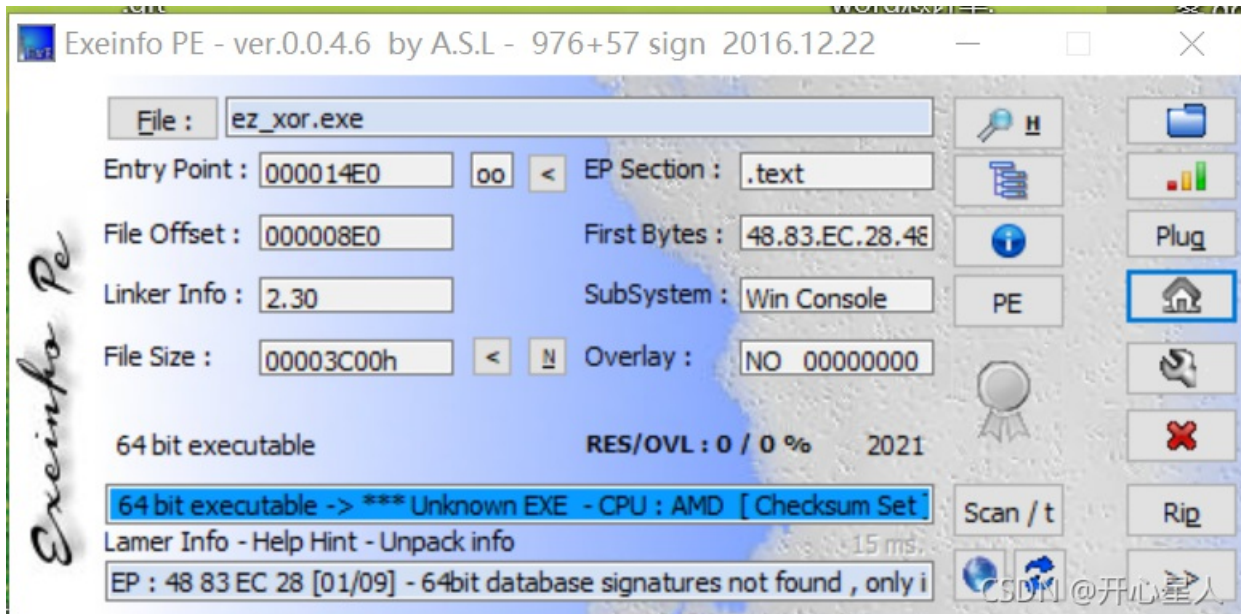
可以看到它的值是十六进制的数，将十六进制的数转换成字符串（选中之后按R键）

```
.rdata:00413E33                align 4
.rdata:00413E34 xmmword_413E34  xmmword '91r_e7a_u0Y{uneh'
.rdata:00413E34                ; DATA XREF: _main+10↑r
.rdata:00413E44 qword_413E44   dq '}!th'
```

注意这里是小端存储方式，要从后往前读（其实根据代码的含义也可以大概判断出来），flag就出来了

14、ez_xor

拿到ez_xor.exe附件直接丢进PE



可以看到是64位exe文件，丢进ida64

Shift+F12查看字符串（如果是笔记本电脑的话，F12自带热键，先按Fn，即Fn+Shift+F12）

Address	Length	Type	String
.rdata:00000019	00000019	C	Please input your flag:
.rdata:00000017	00000017	C	The flag is henu{%.}\n
.rdata:00000006	00000006	C	pause
.rdata:0000001F	0000001F	C	Argument domain error (DOMAIN)
.rdata:0000001C	0000001C	C	Argument singularity (SIGN)

.rdata:000000... 00000020	C	Overflow range error (OVERFLOW)
.rdata:000000... 00000025	C	Partial loss of significance (PLOSS)
.rdata:000000... 00000023	C	Total loss of significance (TLOSS)
.rdata:000000... 00000036	C	The result is too small to be represented (UNDERFLOW)
.rdata:000000... 0000000E	C	Unknown error
.rdata:000000... 0000002B	C	_matherr(): %s in %s(%g, %g) (retval=%g)\n
.rdata:000000... 0000001C	C	Mingw-w64 runtime failure:\n
.rdata:000000... 00000020	C	Address %p has no image-section
.rdata:000000... 00000031	C	VirtualQuery failed for %d bytes at address %p
.rdata:000000... 00000027	C	VirtualProtect failed with code 0x%x
.rdata:000000... 00000032	C	Unknown pseudo relocation protocol version %d.\n
.rdata:000000... 0000002A	C	Unknown pseudo relocation bit size %d.\n
.rdata:000000... 00000007	C	.pdata
.rdata:000000... 0000003F	C	GCC: (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0

CSDN @开心星人

一般在这里找有没有和flag相关的字符串，可以看到这里是有的，双击进入

```

.rdata:0000000000404000 ; DATA XREF: main+Ffo
.rdata:0000000000404019 ; const char aS[]
.rdata:0000000000404019 aS db '%s',0 ; DATA XREF: main+1Bfo
.rdata:000000000040401C ; const char aTheFlagIsHenuS[]
.rdata:000000000040401C aTheFlagIsHenuS db 'The flag is henu{%s}.\',0Ah,0 ; DATA XREF: main:loc_402C55fo
.rdata:000000000040401C ;
.rdata:0000000000404033 ; const char Command[]
.rdata:0000000000404033 Command db 'pause',0 ; DATA XREF: main+64fo
.rdata:0000000000404039 align 20h
.rdata:0000000000404040 ; const struct _EXCEPTION_POINTERS ExceptionInfo
.rdata:0000000000404040 ExceptionInfo _EXCEPTION_POINTERS <offset qword_407540, offset ContextRecord>
.rdata:0000000000404040 ; DATA XREF: sub_401720+B7fo
.rdata:0000000000404050 align 20h
.rdata:0000000000404060 off_404060 dq offset TlsCallback_0 ; DATA XREF: .rdata:off_404370lo
.rdata:0000000000404068 align 20h
.rdata:0000000000404080 TlsDirectory dq offset TlsStart
.rdata:0000000000404088 TlsEnd_ptr dq offset TlsEnd
.rdata:0000000000404090 TlsIndex_ptr dq offset TlsIndex
.rdata:0000000000404098 TlsCallbacks_ptr dq offset TlsCallbacks
.rdata:0000000000404A0 TlsSizeOfZeroFill dd 0
.rdata:0000000000404A4 TlsCharacteristics dd 0
.rdata:0000000000404A8 align 20h
.rdata:00000000004040C0 aArgumentDomain db 'Argument domain error (DOMAIN)',0 ; DATA XREF: sub_401940:loc_401971fo
.rdata:00000000004040C0 ;
.rdata:00000000004040DF aArgumentSingul db 'Argument singularity (SIGN)',0 ; DATA XREF: sub_401940:loc_4019E0fo
.rdata:00000000004040DF ;

```

点击进入该字符串在main方法中出现的位置

找到该字符串，点击上图所示，进入main方法

会进入流程图界面，按空格进入文本界面

可以看到汇编代码了，按F5（同理如果是笔记本记得按Fn+F5）反汇编，转换成C语言

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    __int64 v3; // rax
    char v5[40]; // [rsp+20h] [rbp-28h] BYREF
    char v5[40]; // [rsp+20h] [rbp-28h] BYREF
    sub_401600(argc, argv, envp);
    printf("Please input your flag: ");
    scanf("%s", v5);
    v3 = 0i64;
    while ( (char)(v3 ^ v5[v3]) == dword_403020[v3] )
    {
        if ( ++v3 == 32 )
        {
            printf("The flag is henu{%s}.\n", v5);
            system("pause");
            return 0;
        }
    }
    return 0;
}

```

CSDN @开心星人

现在就可以分析代码了，这里的C语言可能数据类型之类的会和我们平时的有点不一样
 比如说这里的v3=0i64，0i64表示int64_t类型的0，其实就基本上可以理解为0
 这里代码可以看到关键字或代码while ((char)(v3 ^ v5[v3]) == dword_403020[v3])

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    __int64 v3; // rax
    char v5[40]; // [rsp+20h] [rbp-28h] BYREF

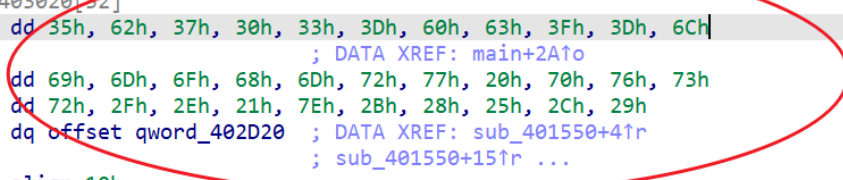
    sub_401600(argc, argv, envp);
    printf("Please input your flag: ");
    scanf("%s", v5);
    v3 = 0i64;
    while ( (char)(v3 ^ v5[v3]) == dword_403020[v3] )
    {
        if ( ++v3 == 32 )
        {
            printf("The flag is henu{%s}.\n", v5);
            system("pause");
            return 0;
        }
    }
    return 0;
}
```

双击进入该字符串进行查看



CSDN @开心星人

```
.data:0000000000403000 ;org 403000h
.data:0000000000403000 dword_403000 dd 0Ah ; DATA XREF: sub_401180:loc_40130F1w
.data:0000000000403004 align 20h
.data:0000000000403020 ; _DWORD dword_403020[32]
.data:0000000000403020 dword_403020 dd 35h, 62h, 37h, 30h, 33h, 3Dh, 60h, 63h, 3Fh, 3Dh, 6Ch ; DATA XREF: main+2A10
.data:0000000000403020 ; DATA XREF: main+2A10
.data:0000000000403020 dd 69h, 6Dh, 6Fh, 68h, 6Dh, 72h, 77h, 20h, 70h, 76h, 73h
.data:0000000000403020 dd 72h, 2Fh, 2Eh, 21h, 7Eh, 2Bh, 28h, 25h, 2Ch, 29h
.data:00000000004030A0 off_4030A0 dq offset qword_402D20 ; DATA XREF: sub_401550+41r
.data:00000000004030A0 ; sub_401550+151r ...
.data:00000000004030A8 align 10h
.data:00000000004030B0 db 0FFh
.data:00000000004030B1 db 0FFh
.data:00000000004030B2 db 0FFh
.data:00000000004030B3 db 0FFh
.data:00000000004030B4 db 0FFh
.data:00000000004030B5 db 0FFh
.data:00000000004030B6 db 0FFh
.data:00000000004030B7 db 0FFh
.data:00000000004030B8 db 0
.data:00000000004030B9 db 0
.data:00000000004030BA db 0
.data:00000000004030BB db 0
.data:00000000004030BC db 0
.data:00000000004030BD db 0
.data:00000000004030BE db 0
```



CSDN @开心星人

可以看到该字符串每个字符对应的ASCII码（这里按R键即可看到对应的字符）
 现在已知dword_403020和v3（v3就是0~31），逐个进行异或即可得到flag

写一个Python脚本

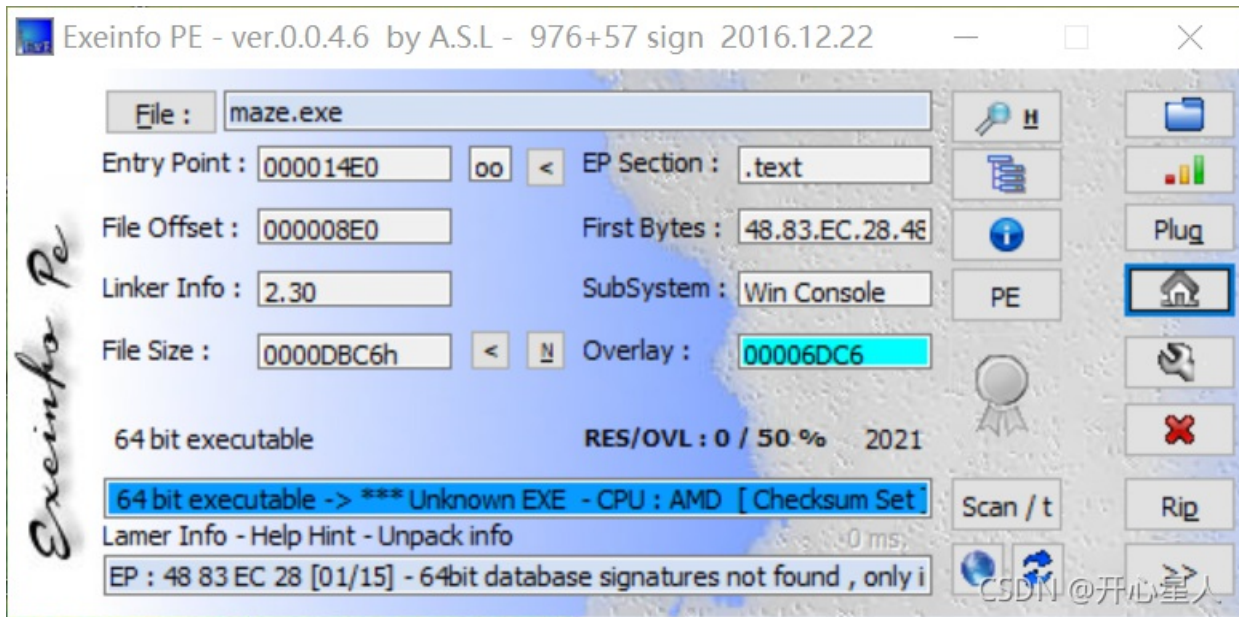
```

s=[0x35, 0x62, 0x37, 0x30, 0x33, 0x3D, 0x60, 0x63, 0x3F, 0x3D, 0x6C,0x69, 0x6D, 0x6F, 0x68, 0x6D, 0x72, 0x77, 0x
20, 0x70, 0x76, 0x73,0x72, 0x2F, 0x2E, 0x21, 0x7E, 0x2B, 0x28, 0x25, 0x2C, 0x29]
flag=[0 for i in range(32)] #从给出的代码很容易看到flag是32位的
for i in range(32):
    flag[i]=i^s[i]
print(flag)

```

即可得出flag

15、maze



可以看到是64位可执行文件，拖入64位ida
找到主函数main，F5反汇编查看C语言代码

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    int result; // eax
    char Str[32]; // [rsp+20h] [rbp-70h] BYREF
    char v5[63]; // [rsp+40h] [rbp-50h]
    char v6; // [rsp+7Fh] [rbp-11h]
    int v7; // [rsp+84h] [rbp-Ch]
    int i; // [rsp+88h] [rbp-8h]
    int v9; // [rsp+8Ch] [rbp-4h]

    _main(argc, argv, envp);
    v5[0] = 5;
    v5[1] = 0;
    v5[2] = 0;
    v5[3] = 0;
    v5[4] = 0;
    v5[5] = 0;
    v5[6] = 0;
    v5[7] = 0;
    v5[8] = 1;
    v5[9] = 0;
    v5[10] = 0;
    v5[11] = 0;
    v5[12] = 0;

```

```
v5[12] = 0;
v5[13] = 0;
v5[14] = 0;
v5[15] = 0;
v5[16] = 1;
v5[17] = 1;
v5[18] = 1;
v5[19] = 0;
v5[20] = 0;
v5[21] = 0;
v5[22] = 0;
v5[23] = 0;
v5[24] = 0;
v5[25] = 0;
v5[26] = 1;
v5[27] = 0;
v5[28] = 1;
v5[29] = 1;
v5[30] = 1;
v5[31] = 0;
v5[32] = 0;
v5[33] = 0;
v5[34] = 1;
v5[35] = 0;
v5[36] = 1;
```

- List item

```
v5[37] = 0;
v5[38] = 1;
v5[39] = 0;
v5[40] = 0;
v5[41] = 0;
v5[42] = 1;
v5[43] = 0;
v5[44] = 1;
v5[45] = 0;
v5[46] = 1;
v5[47] = 0;
v5[48] = 0;
v5[49] = 0;
v5[50] = 1;
v5[51] = 1;
v5[52] = 1;
v5[53] = 0;
v5[54] = 1;
v5[55] = 0;
v5[56] = 0;
v5[57] = 0;
v5[58] = 0;
v5[59] = 0;
v5[60] = 0;
v5[61] = 0;
v5[62] = 1;
v6 = 5;
v9 = 0;
i = 0;
printf(Format);
scanf("%s", Str);
v7 = strlen(Str);
```

```
if ( v7 == 20 )
{
  for ( i = 0; i <= 19; ++i )
  {
    if ( Str[i] == 85 )
      v9 -= 8;
    if ( Str[i] == 74 )
      v9 += 8;
    if ( Str[i] == 72 )
      --v9;
    if ( Str[i] == 75 )
      ++v9;
    if ( !v5[v9] )
      break;
  }
  if ( v9 == 63 && v6 == 5 )
    puts("Congratulations!");
  else
    puts("Sorry,your flag is wrong!");
  system("pause");
  result = 0;
}
else
{
  puts("Sorry,your flag is wrong!");
  system("pause");
  result = 0;
}
return result;
}
```

```

5 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0
0 0 1 0 1 1 1 0
0 0 1 0 1 0 1 0
0 0 1 0 1 0 1 0
0 0 1 1 1 0 1 0
0 0 0 0 0 0 1

```

由 $v7=20$, 易知 flag 长度为 20

进行 20 次循环, 如果 $str[i]$ 的 ASCII 为 85, $v9-=8$, 表示向上走 (这也就是为啥要把迷宫画成 8 行 8 列)

如果 $str[i]$... 为 74, $v9+=8$, 向下走

... .. 为 72, $--v9$, 向左走

... .. 为 75, $++v9$, 向右走

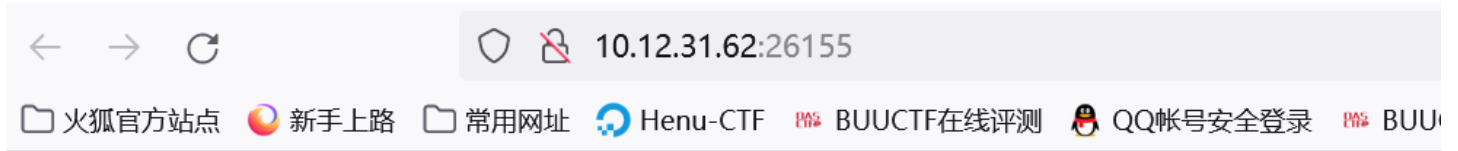
又由: $(!(v5[v9]))$... (必须只能 1 不能走 0, 否则就退出循环)

break;

迷宫的开头是 $v5[0]$, 结尾是 $v5[6]$

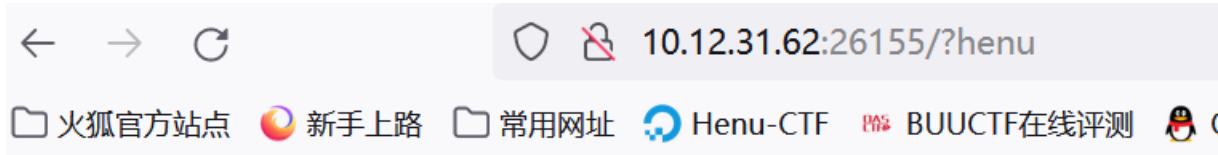
先写出迷宫表法
 对应 str ↓ ↓ → → ↓ ↓ ↓ ↓ → → ↑ ↑ ↑ → → ↓ ↓ ↓ ↓ →
 JJ KK JJ JJ KK UUU KK JJ JJ K

str 即为 flag



我get不到'henu'呀!!!

提示get不到henu，那get henu试一下



Wrong Wrong Wrong!!!

CSDN @开心星人

查看源码

```
<html>
<head>
<meta charset="utf-8" />
<title>你很弱唉</title>
</head>
<body>
<!--
$m1 = md5('s214587387a');
$a=@$_GET['henu'];
$m2 = @md5($a);
if(isset($a)) {
    if($a != 's214587387a' && $m2==$m1) {
        echo "flag{XXXXXXXXXXXXXXXX}";
    }else{
        echo "Wrong Wrong Wrong!!!";
    }
}else{
    echo "我get不到'henu'呀!!!";
}
-->
Wrong Wrong Wrong!!!</body>
</html>
```

CSDN @开心星人

这题考察的是：常见的MD5碰撞：md5值为0e开头

你需要get传参，使得参数的md5值和s214587387a的md5值相等，但又不能等于s214587387a

因为0e开头的md5的原值包括有s214587387a
因此只需要传递一个0e开头的md5的原值即可

0e开头的md5和原值:

QNKCDZO

0e830400451993494058024219903391

240610708

0e462097431906509019562988736854

s878926199a

0e545993274517709034328855841020

s155964671a

0e342768416822451524974117254469

s214587387a

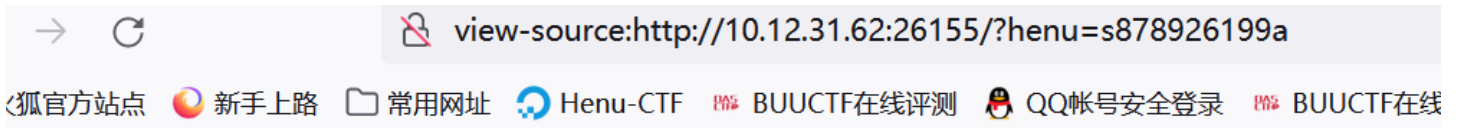
0e848240448830537924465865611904

s214587387a

0e848240448830537924465865611904

s878926199a

0e545993274517709034328855841020 [CSDN @开心星人](#)



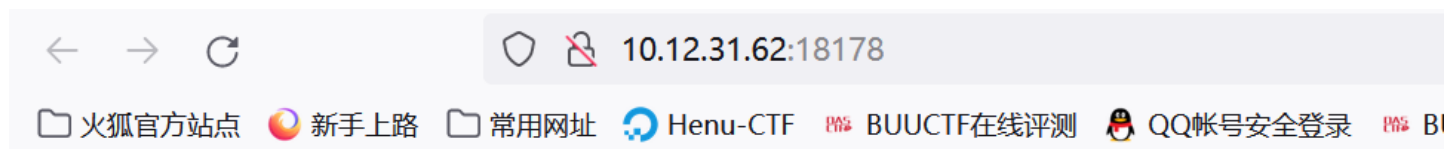
```
<html>
<head>
<meta charset="utf-8" />
<title>你很弱唉</title>
</head>
<body>
<!--
$m1 = md5('s214587387a');
$a=@$_GET['henu'];
$m2 = @md5($a);
if(isset($a)) {
    if($a != 's214587387a' && $m2==$m1) {
        echo "flag{XXXXXXXXXXXXXXXX}";
    }else{
        echo "Wrong Wrong Wrong!!!";
    }
}else{
    echo "我get不到'henu'呀!!! ";
}
-->
```



```
flag {henu:xIBtPERmM9WXqwzy0NdLsVv5GcpF1hai}
</body>
</html>
```

CSDN @开心星人

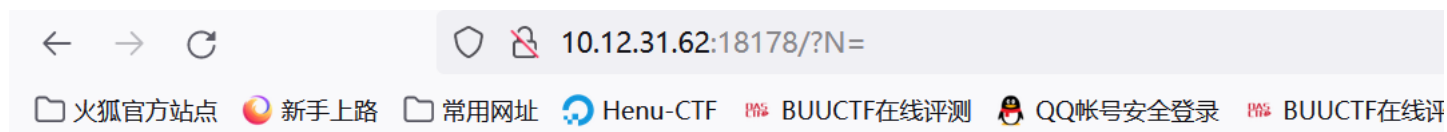
17、easy_bypass



好像少了个参数N呢...

CSDN @开心星人

他提示说少了一个参数N，那就给他传一个参数N



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>easy_php</title>
  </head>
  <body>
    <?php
error_reporting(0);
if(isset($_GET['N'])){
    highlight_file(__FILE__);
    $number = $_GET['N'];
    if(preg_match("/[0-9A-Za-z]/", $number)){
        die("怎么拿不到flag呢?");
    }
    if(intval($number)){
        include("fllluggggg.php");
        echo $flag;
    }
}
else{
    echo "好像少了个参数N呢..." ;
}
?>
```

</body>
</html>

看到了include("fillagggggg.php")文件包含;
那进去看看吧



得到一串字符有数字0~9和字母A-F组成, 所有推测可能是base16编码

```
5A6D78685A33746F5A5735314F6E513153474A4F55444D324D48647552325A736158684C5A31704A656C4E56536B3835636A46725155553366516F3D
```

编码 解码 清空

```
ZmxhZ3toZW51OnQ1SGJOU DM2MHduR2ZsaXhLZ1pJe1NVSk85cjFrQUU3fQo=
```

解码后又得到一串编码以等号结尾, 由大写字母、小写字母和数字组成, 推测应该是base64

```
ZmxhZ3toZW51OnQ1SGJOU DM2MHduR2ZsaXhLZ1pJe1NVSk85cjFrQUU3fQo=
```

清空 加密 解密 解密为UTF-8字节流

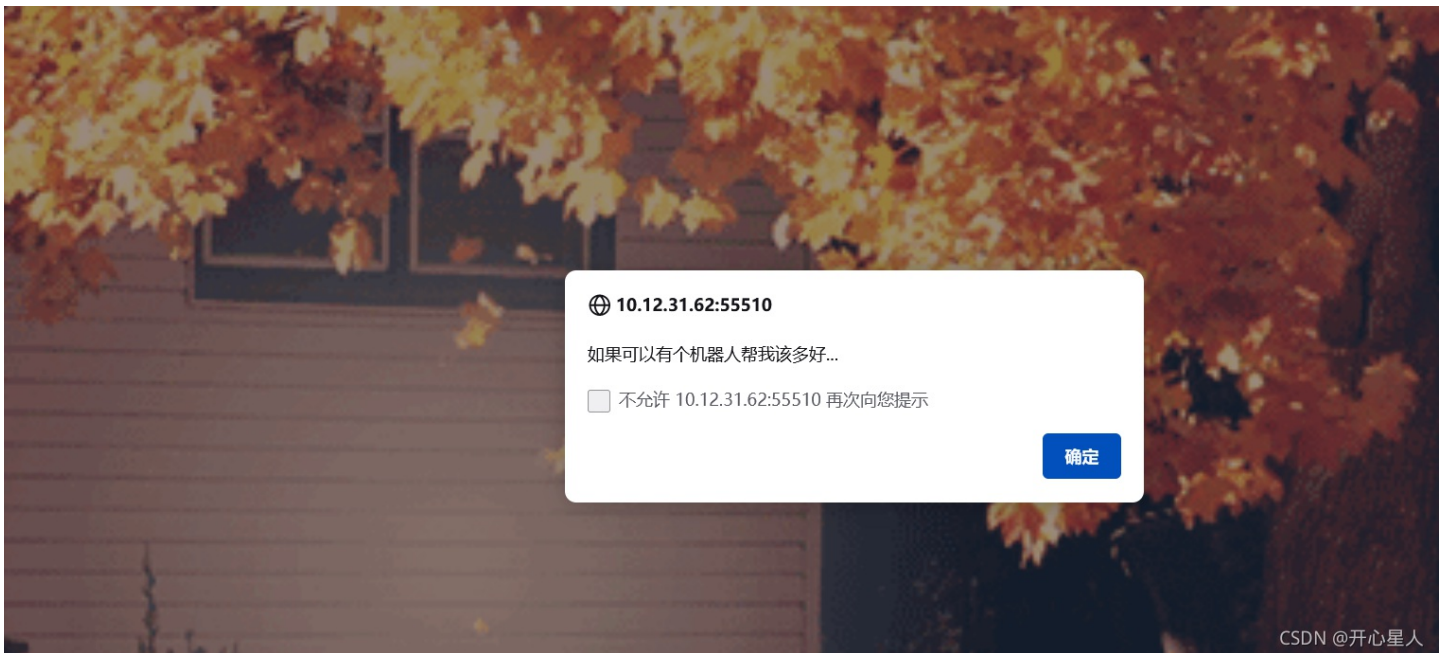
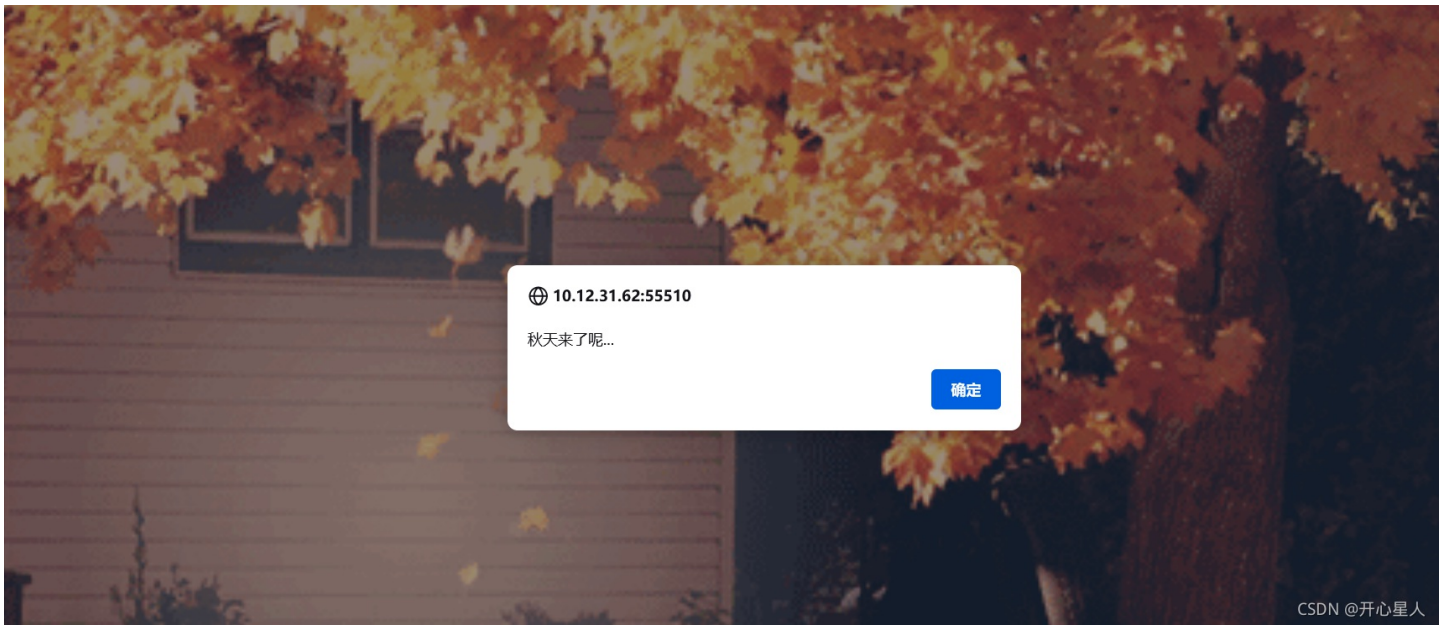
```
flag {henu:t5HbNP360wnGf1ixKgZIZSUJ09r1kAE7}
```

得到flag

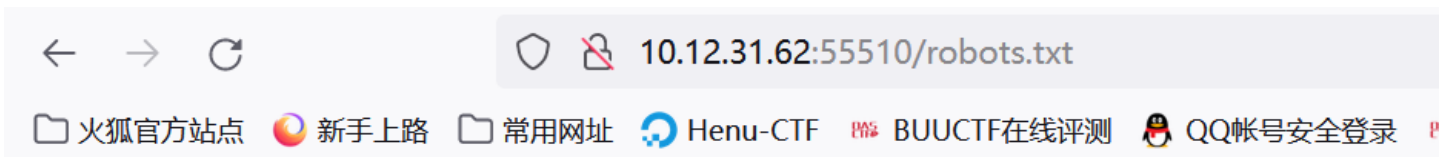
18、Autumn

题目提示: 爱、死亡和机器人
进入网址





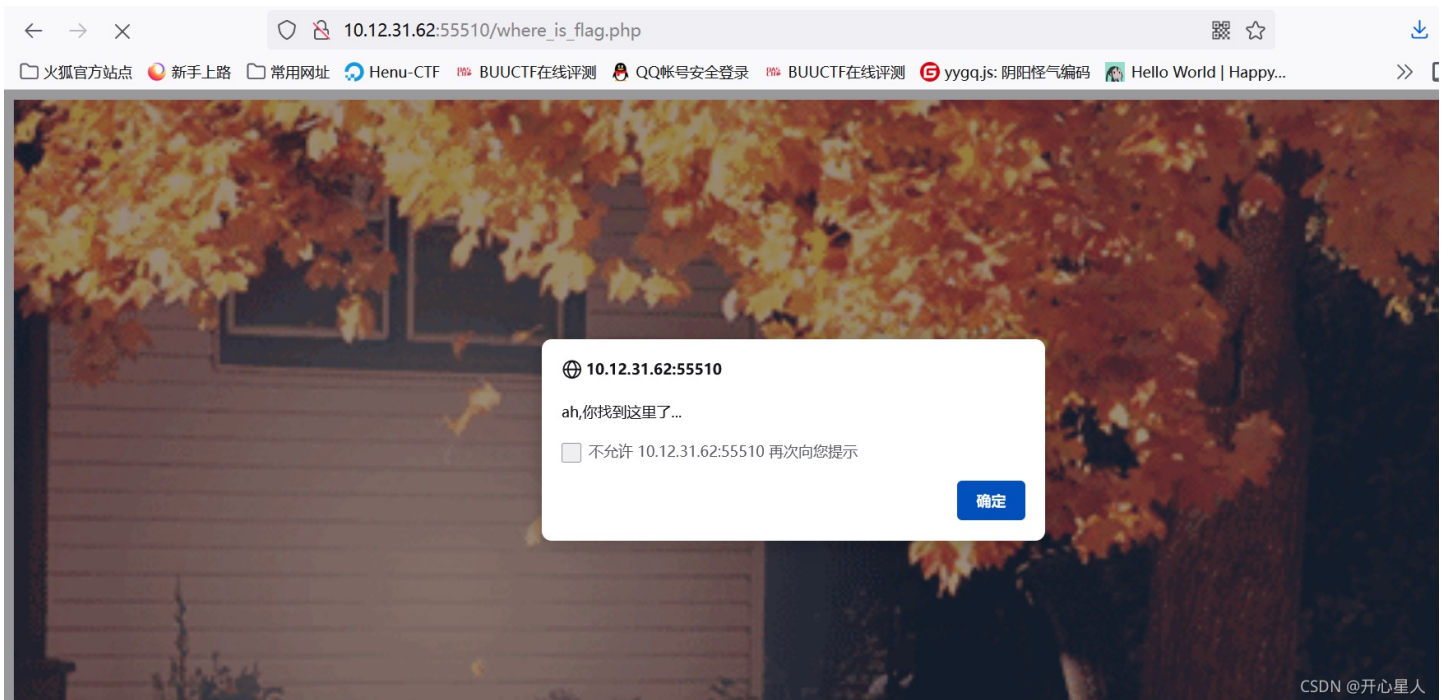
后面都是重复的了，直接关闭dialog，题目多次提示机器人，很容易就让人想到robots.txt,那进去看看吧



```
User-agent: *  
Disallow: where_is_flag.php
```

CSDN @开心星人

又出现了新的提示，where_is_flag.php
进去看看



除了提示框的内容不一样和刚开始界面没有什么不同，现在没有什么提示信息，查看源码，一直往下翻，得到flag

```

<script>alert('ah,你找到这里了...');</script>
<script>alert('可是...');</script>
<script>alert('好像不在这里呢...');</script>

<script>alert('ah,你找到这里了...');</script>
<script>alert('可是...');</script>
<script>alert('好像不在这里呢...');</script>

<script>alert('ah,你找到这里了...');</script>
<script>alert('可是...');</script>
<script>alert('好像不在这里呢...');</script>

<script>alert('ah,你找到这里了...');</script>
<script>alert('可是...');</script>
<script>alert('好像不在这里呢...');</script>

<script>alert('ah,你找到这里了...');</script>
<script>alert('可是...');</script>
<script>alert('好像不在这里呢...');</script>
<!--flag{henu:TUJMzSjNDUxwhtoy0fgmXFLiAQ972KPb}-->
</body>
</html>
```

html > body > img

CSDN @开心星人

19、easy_Cookie



Hi, do you need some cookies?

CSDN @开心星人

题目各种提示cookie，那么我们就看看cookie里面到底有什么

名称	值	Domain	Path	Expires / Max-Age	大小	HttpOnly	Secure
henuctf	Wm14aFozdFVhR1ZmWTI5dmEybGxj...YzE5a1pXehBZMmx2ZFhOOQ%3D%3D	39.107.125.199	/	Tue, 16 Nov 2021 08:58:22 GMT	67	false	false

henuctf: "Wm14aFozdFVhR1ZmWTI5dmEybGxj...YzE5a1pXehBZMmx2ZFhOOQ%3D%3D"
Domain: "39.107.125.199"
Expires / Max-Age: "Tue, 16 Nov 2021 08:58:22 GMT"
HostOnly: true
HttpOnly: false
Path: "/"
SameSite: "None"
Secure: false
创建于: "Tue, 16 Nov 2021 07:58:22 GMT"
大小: 67
最后访问: "Tue, 16 Nov 2021 07:58:22 GMT"

调用web服务时，参数中=变成%3D

所以这里将%3D换成=

猜测为Base64编码

解码后得到

CSDN @开心星人

Wm14aFozdFVhR1ZmWTI5dmEybgxjMT1wYzE5a1pXeHBZMmx2ZFh00Q==

解密为UTF-8字节流

ZmxhZ3tUaGVfY29va21lc19pc19kZWxpY21vdXN9

CSDN @开心星人

解码之后发现结果仍然是一串编码

再次base64解码即可得到flag

20、[白给] 连上就给flag

The screenshot shows a CTF challenge interface with a dark blue background. At the top, the challenge title is "[白给] 连上就给flag" in yellow. To the right, it says "分值: 100" and "已解答". Below the title, there are three user avatars with their IDs: "2012080149-林安康", "2024030005-李金刚", and "2012080140-彭立". The main text of the challenge reads: "z学长ida打开附件，两眼放光，心想这不是我做的最熟练的经典nc题吗？急忙打开虚拟机，反手一个nc,ls,cat flag,一系列操作行云流水，拿到了flag。温馨提示：1.虚拟机建议装ubuntu18或20；2.什么是nc? ### 远程地址：39.103.198.196:10008". Below the text, there is a button labeled "whitegive_connect" and a large input field. At the bottom right, there is a "提交" (Submit) button. The CSDN @开心星人 watermark is visible in the bottom right corner.

根据题目提示nc即可得到flag

在terminal界面

```
nc 39.103.198.196 10008
```

```
ls #显示当前目录下文件信息
```

```
然后发现里面有个flag
```

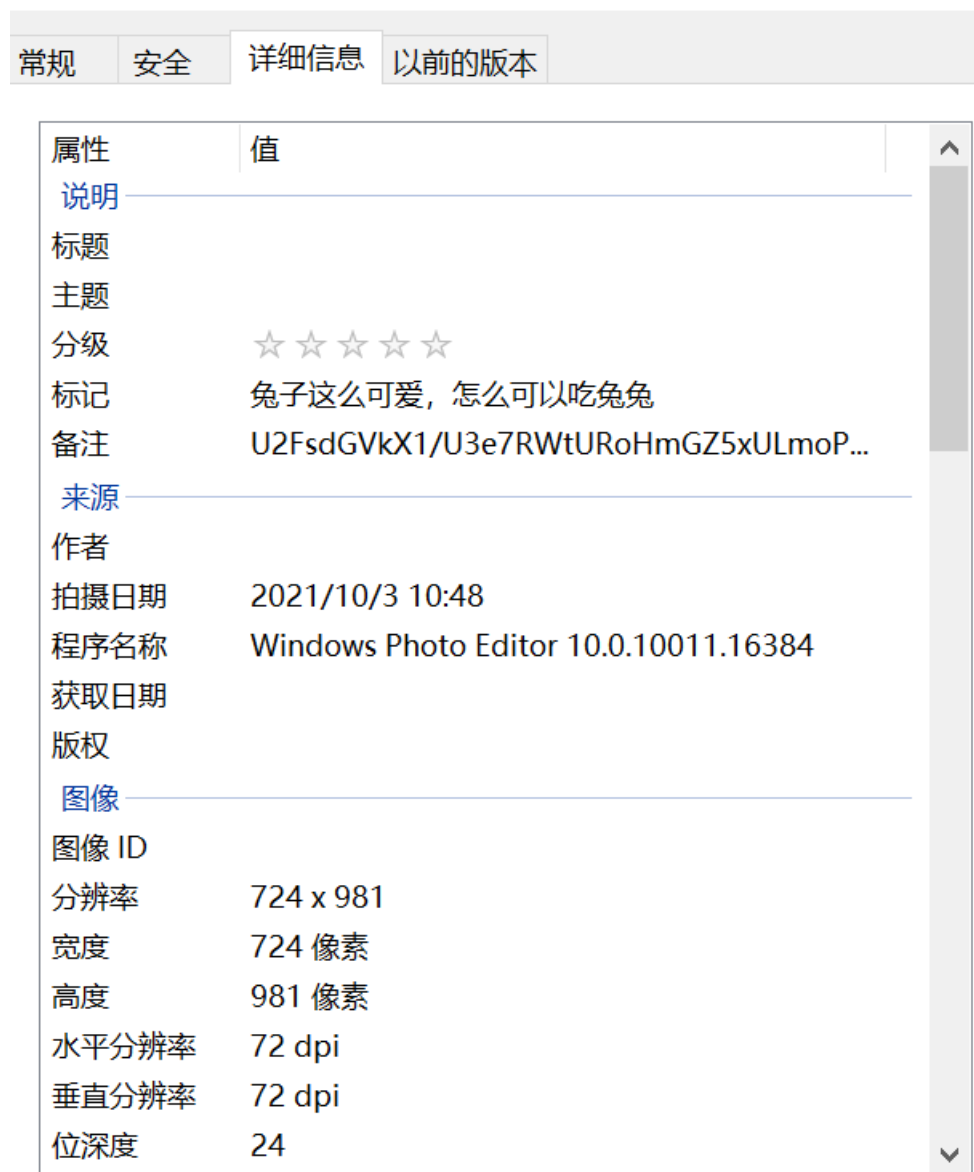
```
cat flag
```

```
即可得到flag
```

21、小兔子

拿到附件图片

一般拿到附件图片，先查看图片的属性



[删除属性和个人信息](#)

CSDN @开心星人

很明显这里的备注有异样内容，又由文件名和图片名小兔子，猜测是兔子密码

解密即可得到flag

22、我在城楼观山景

打开附件，发现是一堆乱码

時NG

□

IHDR □8 □!□□ n?h □sRGB 鑽? □gAMA 皖□默□ pHYs □% □%□IR\$? DATx^ 堯&M?h}gc?态TI@w?棘□恁^?桌四?
 ?\馥棗mW囡m?左笏飡?□控 □箴p p跽 ?_y齋蟬罽饅场^鞞韃z 鏞?禘?廿廬~鳴_ 椽yd?鮪帑吵搽蕙莞no痲鹿洮塚w枸+n? c雉w [箒瘠
 瘁<轍□B d陳?溱麒□y溼勳擲x鯁?蓋鏤 !{?□啣? 圻戩鯨?餗 樟榛 鞞?>o瘡澧77□? 你佩碾展?燭□冀鏡驟??謹lh积R躄措□蓄體/像
 鷗S?壞崆嶺??樹?吹黛y耐?)Y
 g p{荃e□陆.礪%拜O螢碧蝌g `?紧涓 ?咪麥KC€>q報□拮/維n?w)S m贏|/. ?□Xy4衿旱釘z~□筋 鄔?!鑄□□銜1]?m>|骨 o□呷□?戌`!
 騰_H闖 s€}|<□戌□S ? ?ht葵鎬>埃?俚^4{>pG□□v孰闕?0溏□ :!橡S 獠?
 劍□亨-衛 0r@遊 桃蘆允纏班x□厦蚱?&mq 顛墻\$-江Q憂 轆i□□M邨溥轄 V □壳菓^焮□橙□鳴#譌/?m□鬚凰隍檢o?□玳□?□鶉~? ?\?'!
 欠杞k你? 3紉胸蹠0罐?v礪??O`?.?皴峯□ _fS燦? 馱潔d?it ~鑄紮?軹b潼殆鍊荫D # |g黠F港\R/ 蟻趁?p鋤捆 砧?;埤5縱□x叢陰b覆切{
 登冻河硿筓譬mF Cr ?P 炅O鯊7哨赫□<菹?蝥~勳植a莹□嫉鶻?c齧>钱~:t{寢羈F援2嶋?€? □<F 遠'? ?爵 Y焱華 ?巖戾蝮b?>俱
 ,p孑鷲 ;庠模.婁; >楔O滂滴鷄b□rs弑洗□□??魁 ?蠱?&諛□V諛闹宓s肩-词爵sl嘍?桴6縫G揆9<?眈No?u璫&f?>囿孽嬉 l儗jg々鐸種
 Sgg2? w_??kd? lannu等滬5

猜测是文件类型错误

把文件放入010 Editor中查看文件的真实类型（其实这里通过文件开头的NG就可以推测出文件的类型是png）

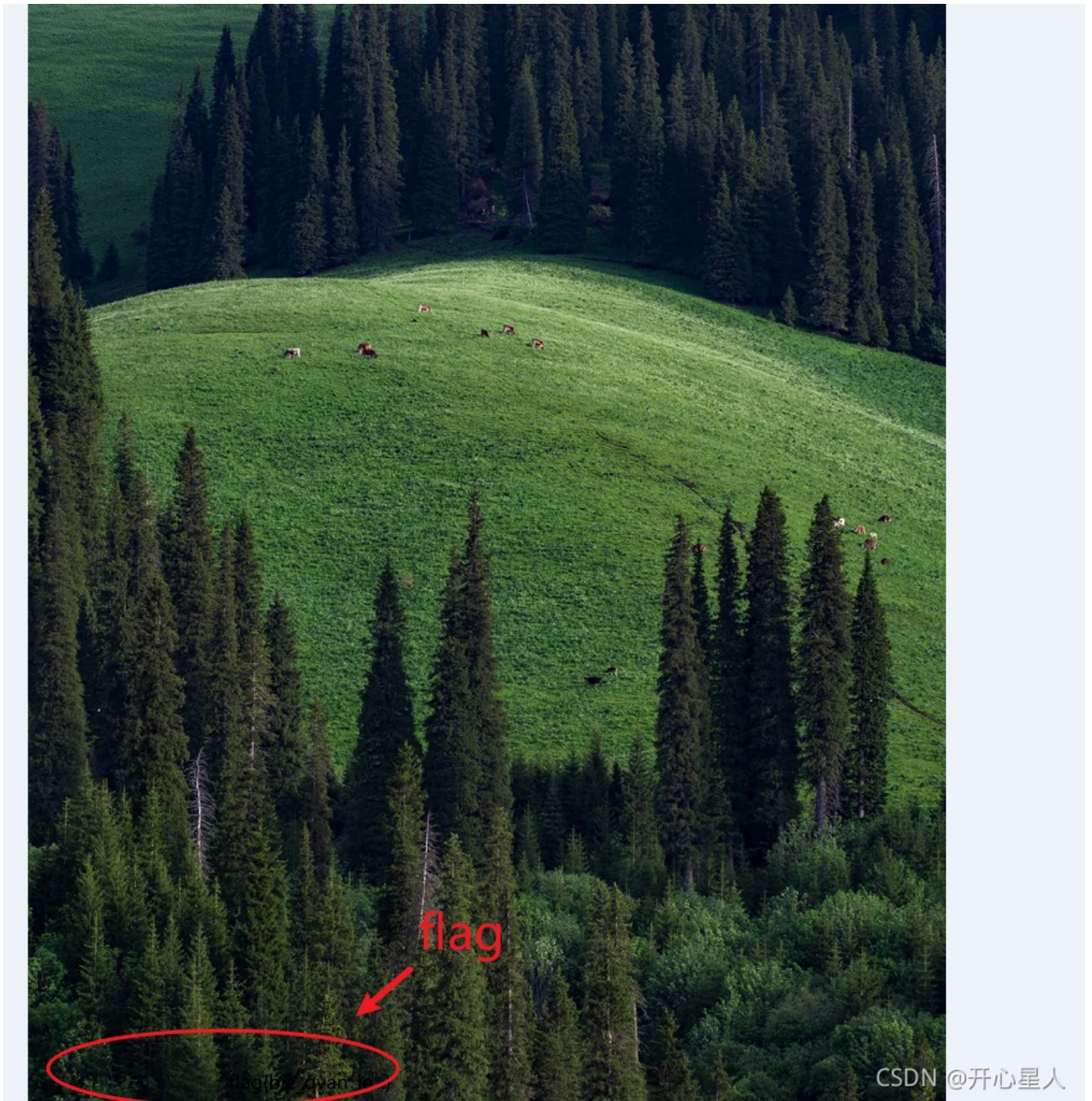


- JPEG (jpg), 文件头: FFD8FF
- PNG (png), 文件头: 89504E47
- GIF (gif), 文件头: 47494638
- ZIP Archive (zip), 文件头: 504B0304
- RAR Archive (rar), 文件头: 52617221
- XML (xml), 文件头: 3C3F786D6C
- HTML (html), 文件头: 68746D6C3E
- MS Word/Excel (xls.or.doc), 文件头: D0CF11E0
- Adobe Acrobat (pdf), 文件头: 255044462D312E
- ZIP Archive (zip), 文件头: 504B0304
- RAR Archive (rar), 文件头: 52617221

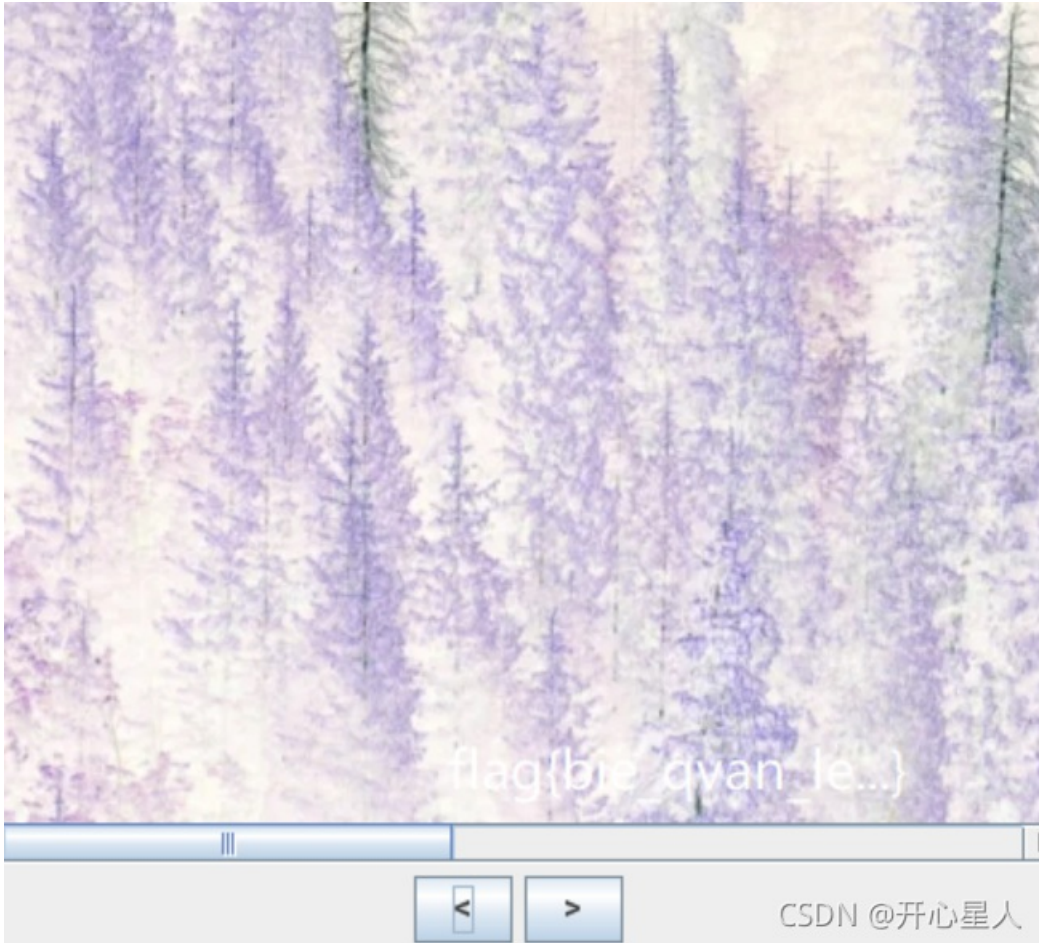
可以看到文件头为89504E47，所以真实类型是png

将文件后缀名改为.png

得到一张这样的图片可以看到图片下方有字样，就是我们的flag（记得将亮度调到最大）



看不出来也没有关系，使用看图工具stegsolve，这次可以清楚地看出flag了



23、Stream



注意看hacker is everywhere



打开附件看到提示说这题需要使用wireshark工具

直接把you_are_hacker.pcapng拖入wireshark中

The image shows the Wireshark interface with a filter applied: `(ip.addr eq 10.211.55.2 and ip.addr eq 10.211.55.15) and (tcp.port eq 55535 and tcp.port eq 80)`. The packet list shows several TCP connections between 10.211.55.2 and 10.211.55.15. Packet 8 is selected, showing a TCP ACK from 10.211.55.2 to 10.211.55.15. The packet details pane shows the Ethernet II header and the source address: `Parallel_00:00:08 (00:1c:42:00:00:08)`. The raw packet bytes are displayed in hexadecimal and ASCII, with the ASCII portion showing `..B.... B....E`.

点击文件->导出对象->HTTP

保存后，因为hacker is everywhere

我当时随便打开了一个PHP文件，发现一串长得像Base64编码的字符，解码之后就得到了flag

24、如来の兽

打开附件

如是我闻：時逝豆恤须念贤精排戏創守親貧足牟开恤求曳百伊足難山重灭曳矜知遠焰亦曳者依哈殿閱藥功倒想

由附件名称(Buddha)和经验推测这是与佛论禅编码

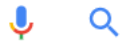
进入与佛论禅在线解密工具

并没有见过这个编码

进入Google搜索



CTF 这 就 这 不会吧？ 不会吧？ 不会吧？ 就这 不



https://www.cxymm.net › article › weixin_44145452

ctf密码学特殊的编码和解密_落雪wink的博客-程序员秘密

最近做密码学的题，遇到好多没见过的解密题，还是太菜了，可恶密码题解密传送门：1个T的种子请自备纸巾1.阴阳怪气编码题目:就这 就这 不会吧？ 就这 不会吧？ 就这 ...

CSDN @开心星人

OK，可以看到是阴阳怪气编码

在网上搜索到一个在线解密工具<https://mmdjiji.gitee.io/yygq.js/>

阴阳怪气编码

welcometohenu

编码 >>

<< 解码

！ 不 云 吧 ！ 就 这 ！ 就 这 ！ 不 会 吧 ？ 不 会 吧 ？ 不 会 吧 ？ 不 会 吧 ？
吧 ？ 就 这 ！ 就 这 ！ 不 会 吧 ？ 不 会 吧 ？ 不 会 吧 ？
就 这 ！ 不 会 吧 ？ 就 这 ！ 就 这 ！ 就 这 ！ 就 这 ！ 不 会
吧 ？ 不 会 吧 ？ 就 这 ！ 不 会 吧 ？ 不 会 吧 ？ 不 会
吧 ？ 不 会 吧 ？ 就 这 ！ 就 这 ！ 不 会 吧 ？ 不 会 吧 ？
就 这 ！ 不 会 吧 ？ 就 这 ！ 就 这 ！ 就 这 ！ 就 这 ！ 就 这
！ 不 会 吧 ？ 不 会 吧 ？ 就 这 ！ 就 这 ！ 不 会 吧 ？ 就
就 这 ！ 不 会 吧 ？ 就 这 ！ 就 这 ！ 不 会 吧 ？ 不 会 吧 ？
就 这 ！ 不 会 吧 ？ 不 会 吧 ？ 不 会 吧 ？ 就 这 ！ 就 这
！ 就 这 ！ 不 会 吧 ？ 不 会 吧 ？ 不 会 吧 ？ 就 这 ！ 不
会 吧 ？ 就 这 ！ 不 会 吧 ？

阴阳怪气编码 —— 不会吧？ 就这 ！

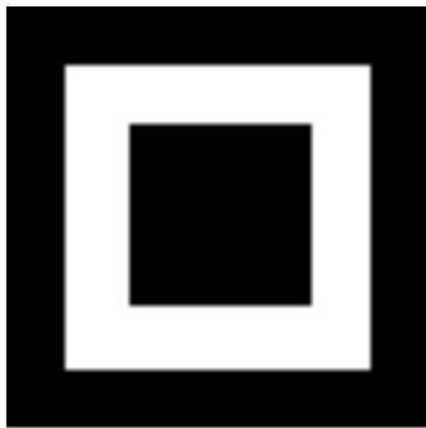
CSDN @开心星人

得到flag

26、QR code

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-SYad2ou0-1637033025190)(C:\Users\张开斌\AppData\Roaming\Typora\typora-user-images\image-20211115085102106.png)]

题目是直接丢过来一个二维码，缺少二维码定位符，所以搜索一下二维码定位符



CSDN @开心星人

截取图片，去除白色背景

（这一题用PowerPoint更简单容易）

Trytouse

分值: 200 已解答

2012080040-张澳韩

2014010038-涂多范

2012080059-田凯

你能够掌握寂静之眼么。

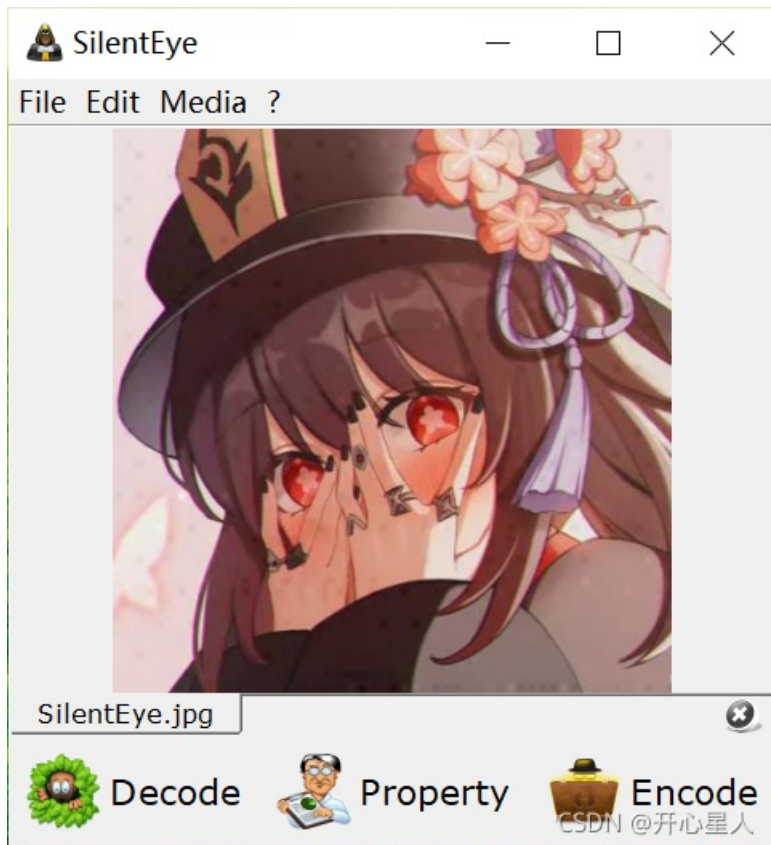
SilentEye.jpg

提交

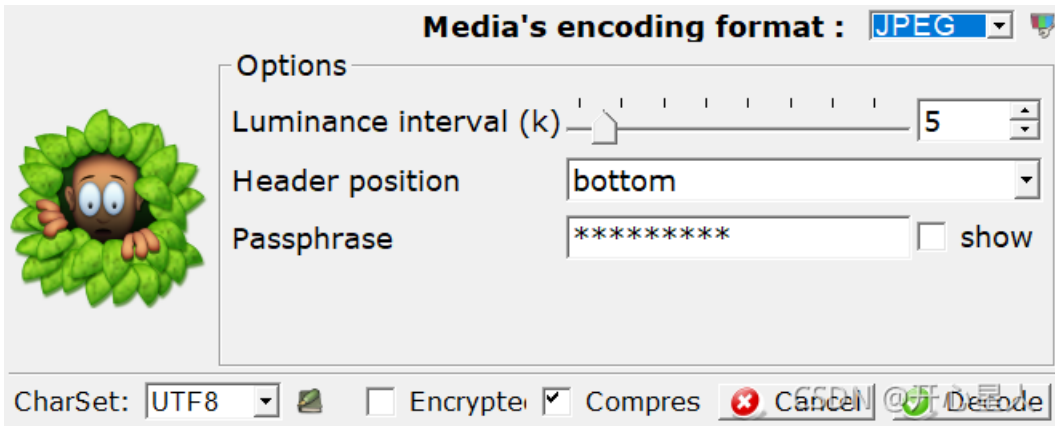
CSDN @开心星人

推测该题目是考察寂静之眼的使用

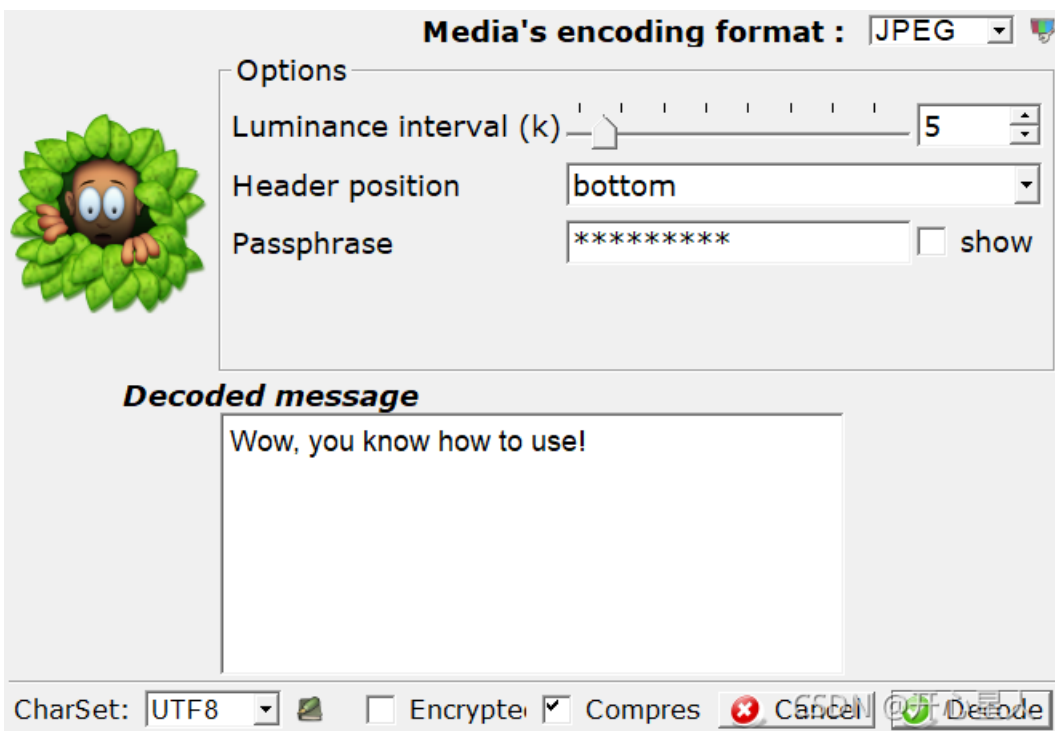
下载附件拖入寂静之眼



点击Decode



再次点击Decode



即可获得flag

28、Boringboss


```

def encrypt(msg, e, n):
    m = bytes_to_long(msg)

    return pow(m, e, n)

hint, (e, n) = generate_key(256)
cipher = encrypt(flag, e, n)

print('hint:{}'.format(hint))
print('public:{}'.format([e, n]))
print('cipher:{}'.format(cipher))

```

CSDN @开心星人

题目代码如下

更改后的主要代码

```

from gmpy2 import *
from Crypto.Util.number import *

e = 0x10001

def gcd(a, b):
    if a % b == 0:
        return b
    else:
        return gcd(b, a % b)

def generate_key(bits):

    p, q = getPrime(bits), getPrime(bits)
    phi = (p-1)*(q-1)
    assert gcd(e, phi) == 1

    return p+q, (e, p*q)

def ext_gcd(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, gcd = ext_gcd(b, a % b) # 递归直至余数等于0(需多递归一层用来判断)
        x, y = y, (x - (a // b) * y) # 辗转相除法反向推导每层a、b的因子使得gcd(a,b)=ax+by成立
    return x, y, gcd

```

```

def encrypt(msg, e, n):
    m = bytes_to_long(msg)
    return pow(m, e,n)

def decrypt(msg, d, n):
    return pow(msg,d,n)

def computed(T, e):
    (x, y, r) = ext_gcd(T, e)
    #y maybe < 0, so convert it
    if y < 0:
        return T + y
    return y

cipher = 3721706087701608031528143109887388627081325867064279646928798190372280395374522547064987799846969351893
973138918258025444740621243876030264735011810425237
hint = 180542641103954693101505542753644053708448862377824438761565169103024337070352
N = 796380896728355800172729827191551126221938467424353819838766498432912746132171662834435146796647856681545278
7078495773626878495336572392243183878207924327
T = N-hint+1 # 这是这一题的关键, 要通过T计算密钥 (p-1)*(q-1)=T
E = 65537
D = computed(T, E)
flag = decrypt(cipher, D, N)
flag = long_to_bytes(flag)
print(flag)

```

```

b'henu{you_really_know_how_to_solve_rsa}'
>>>

```

这题明文已知，公钥(N,E)已知

通过hint和N可以求出T

再通过 $DE \bmod T=1$ 计算出密钥D（要使用扩展欧几里得）

然后用私钥解密，即可求出密文

但注意要long_to_bytes！