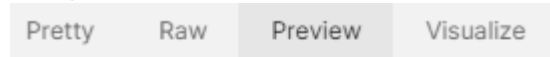


web

Hitchhiking_in_the_Galaxy

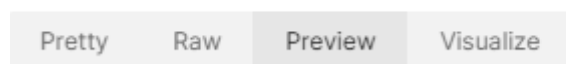
使用 post 提交请求后出现



只有使用"无限非概率引擎"(Infinite Improbability Drive)才能访问这里~

似乎是要改 UA 标识, 于是改 UA 之后出现这个

body Cookies Headers (8) Test Results



你知道吗? 茄子特别要求: 你得从他的Cardinal过来

那么继续伪造 Referer, 得



flag仅能通过本地访问获得

于是再添加 X-Forwarded-For 为127.0.0.1就得 flag

watermelon

方法一: 玩到2000分, 因为它这个游戏的物理引擎有一点bug, 因此可以轻松玩到2000分

方法二: 在源代码的第3436行和第3440行找到修改score的代码!

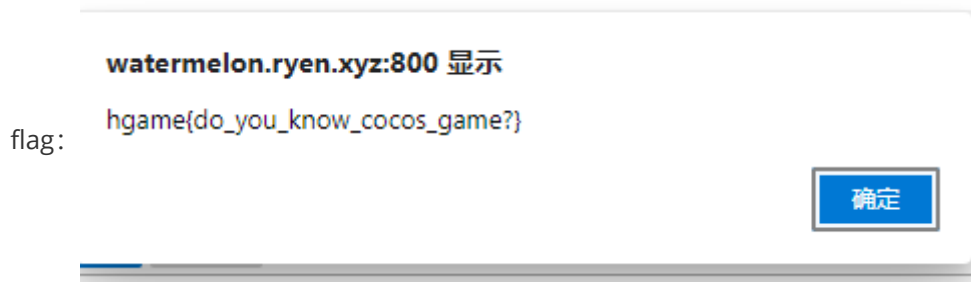
直接修改为+2000

```
3436onent("fruitData").getNumber() && (a.default.score += this.fruitNumber + 2000
3437
3438
3439efault.Instance.createFruitL(o.fruitNumber, n.node.position, n.node.width), i
3440 == t.node.getComponent("fruitData").getNumber() && (a.default.score += this.
3441
```

然后玩到游戏结束, 结果玩了半天一直结束不了, 于是去找控制游戏结束的代码

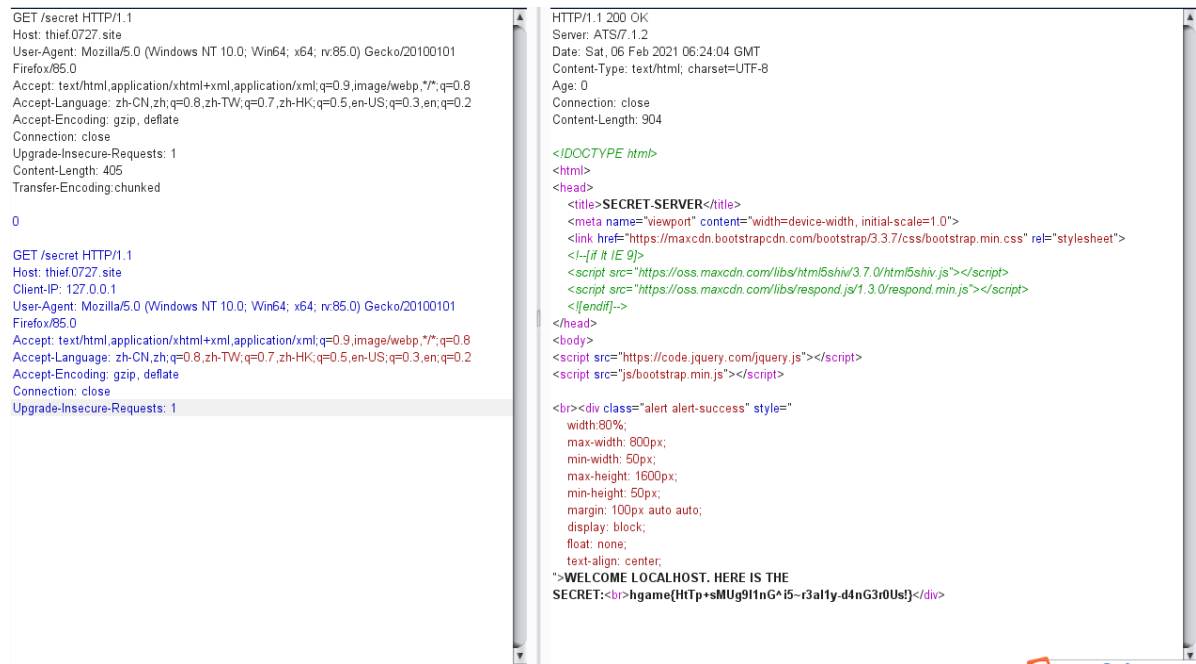
```
3412ianjieX), this.notTargetTime += e, this.returnNumber && (this.scheduleOnce(function () {
3413
3414e.width / 2 > cc.find("Canvas/lineNode").children[0].y && 0 == this.pengzhuangCount && this.endCtrl && 0 == this.endOne && this.testEndDJS > 3||true) {
3415
3416Component(cc.PhysicsCircleCollider), cc.find("Canvas/fruitNode").children[n].removeComponent(cc.RigidBody);
3417
```

在第3414行, 整句if只要有一个条件不满足游戏就不结束, 于是直接在后面||true, 游戏结束, 拿到



宝藏走私者

看资料，是有关http请求走私的，于是简单学习了一下，构造如下请求，拿到flag



但为什么第五题做不出来呢？

智商检测鸡

做定积分题。。。直接拿手机的微软数学扫，做了几道题发现太慢了，于是去看资料，原来是要用爬虫的，于是用自己蹩脚的 python 和现学的 BeautifulSoup 写了个爬虫：

```
import requests
import json
from bs4 import BeautifulSoup
from sympy import *
x = Symbol('x')

def cal(html):
    soup = BeautifulSoup(html)
    a=int(soup.math.mrow.msubsup.mrow.mn.string)
    if str(soup.math.mrow.msubsup.mrow.mo) != 'None':
        a=-a
    b=int(soup.math.mrow.msubsup.mrow.next_sibling.mn.string)

    if(str(soup.math.mrow.msubsup.mrow.next_sibling.mo) != 'None'):#.string != 'None':
        b=-b
    soup.math.mrow.msubsup.extract()
    print(soup.text)
    if str(soup.math.mrow.mo.next_element.string) != '-':
        c=int(soup.math.mrow.mn.string)
    else:
        c=-int(soup.math.mrow.mn.string)
    soup.mn.extract()
    d=int(soup.math.mrow.mn.string)
    if str(soup.math.mrow.mn.previous_element.string) == '+':
        f=c*x+d
```

```

else:
    f=c*x-d
    answer=integrate(f, (x, a, b))
    return answer

s = requests.Session()
for i in range(101):
    s.get("http://r4u.top:5000/")
    cook=s.cookies
    getq=s.get("http://r4u.top:5000/api/getQuestion",cookies=cook)
    print("第%d个: "%i)
    print(getq.text)
    gets=s.get("http://r4u.top:5000//api/getStatus",cookies=cook)
    print(gets.text)
    if i==100:
        a=s.get("http://r4u.top:5000//api/getFlag",cookies=cook)
        print(a.text)
        break
    html=getq.text
    answer=cal(html)
    datas = {'answer':float(answer)}
    header={"Host": "r4u.top:5000",
"User-Agent":"Mozilla/5.0 (windows NT 10.0; win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0",
"Accept":"application/json, text/javascript, */*; q=0.01",
"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",
"Content-Type": "application/json;charset=utf-8",
"Origin": "http://r4u.top:5000",
"Connection": "close",
"Referer": "http://r4u.top:5000/",}

    posta=s.post("http://r4u.top:5000/api/verify",headers=header,data=json.dumps(datas))
    print(posta.text)

```

取得flag

```

第100个:
{"question":"all have done!"}
{"solving":100}
{"flag":"hgame {3very0ne_H4tes_Math}"}
Press any key to continue . . .

```

reverse

apacha

elf文件，于是IDA打开，进入主函数，加密函数很容易找，就是sub_1447这个函数。进入后发现逻辑十分复杂，简单的重命名后变成这个样子

```
1  v4 = &input[a2 - 1];
2  v5 = *v4;
3  v6 = 0;
4  do
5  {
6      v6 -= 1640531527;
7      v7 = v6 >> 2;
8      if ( a2 == 1 )
9      {
10         v9 = 0;
11     }
12     else
13     {
14         v8 = 0LL;
15         do
16         {
17             v5 = input[v8]
18                 + (((v5 >> 5) ^ (4 * input[v8 + 1])) + ((16 * v5) ^ (input[v8 + 1] >> 3))) ^ (((_DWORD *)a3 + 4LL * (((unsigned __int8)v8 ^ (unsigned __int8)v7) & 3)) ^ v5)
19                 + (input[v8 + 1] ^ v6));
20             input[v8++] = v5;
21         }
22         while ( v8 != (unsigned int)(a2 - 2) + 1LL );
23         v9 = a2 - 1;
24     }
25     result = (16 * v5) ^ (*input >> 3);
26     v5 = *v4
27         + ((((_DWORD *)a3 + 4LL * ((v9 ^ (unsigned __int8)v7) & 3)) ^ v5) + (*input ^ v6)) ^ (((4 * *input) ^ (v5 >> 5))
28             + result));
29     *v4 = v5;
30 }
31 while ( v6 != -1640531527 * (52 / a2) - 1253254570 );
```

仔细分析后发现第27行为主要的加密操作，v5是每次加密后得到的结果，并且这个计算过程是原数据加上原数据和下一个数据和上一个v5经过一堆计算后得出的结果（第一个数据的v5是最后一个数据），这样其实所有数据都是已知的，可以直接计算再减回去就好了。并且总共加密要循环好几次，由v6决定，于是写解密代码（大部分内容直接复制即可）：

```
#include<stdio.h>
#include<windows.h>
void fun(__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;
    DWORD input[35] = {
        0x0E74EB323,0x0B7A72836,0x59CAFE2,0x967CC5C1,0x0E7802674,
        0x3D2D54E6,0x8A9D0356,0x99DCC39C,0x7026D8ED,0x6A33FDAD,
        0x0F496550A,0x5C9C6F9E,0x1BE5D04C,0x6723AE17,0x5270A5C2,
        0x0AC42130A,0x84BE67B2,0x705CC779,0x5C513D98,0x0FB36DA2D,
        0x22179645,0x5CE3529D,0x0D189E1FB,0x0E85BD489,0x73C8D11F,
        0x54B5C196,0x0B67CB490,0x2117E4CA,0x9DE3F994,0x2F5AA1AA,
        0x0A7E801FD,0x0C30D6EAB,0x1BADDC9C,0x3453B04A,0x92A406F9 };

    for (v6 = -1640531527 * (52 / 35) - 1253254570; v6 != 0; v6 += 1640531527)
    {
        DWORD* p = &input[33];
        v5 = *p;
        v7 = v6 >> 2;
        result = (16 * v5) ^ (*input >> 3);
        input[34] -= ((((*(_DWORD*)a3 + 4 * ((34 ^ (unsigned __int8)v7) & 3)) ^ v5)
+ (*input ^ v6)) ^ (((4 * *input) ^ (v5 >> 5)) + result));
        v8 = 33LL;
        p--;
        v5 = *p;
        do
        {
            if (v8 == 0)
            {
                v5 = input[34];
```

```

        if (v6 - 1640531527 == 0)
        {
            v5 = '}';
        }
    }
    input[v8] -= (((v5 >> 5) ^ (4 * input[v8 + 1])) + ((16 * v5) ^
(input[v8 + 1] >> 3))) ^ ((* (DWORD*)(a3 + 4LL * ((unsigned __int8)v8 ^
(unsigned __int8)v7) & 3)) ^ v5)
        + (input[v8 + 1] ^ v6));
    p--;
    v5 = *p;
    v8--;
} while (v8 != -1);
}
for (int i = 0; i < 35; i++)
{
    printf("%c", input[i]);
}
}
int main()
{
    int a[] = { 1,2,3,4,0.3 };
    fun((__int64)a);
}

```

得到flag

```
hgame(100ks_like_y0u_f0Und_th3_t34)
```

helloRe

IDA 打开，是 C++ 逆向，幸亏还稍微学过一点。。。不然连 cin 都看不懂。

首先来到 sub_140001290 函数，看起来似乎像是加密函数，但为什么传的参数这么奇怪？

```

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 cout(std::cout, (__int64)" .");
}

```

最后发现这个函数原来就是 sleep，真正的加密函数就在下方

```
27 printwrong:
28     printWrong();
29     v6 = v15;
30     v7 = (void **)input[0];
31     do
32     {
33         v8 = input;
34         if ( v6 >= 16 )
35             v8 = v7;
36         if ( *((_BYTE *)v8 + v3) ^ (unsigned __int8)sub_140001430() ) != byte_140003480[v3] )
37             goto printwrong;
38         ++v3;
39     }
40     while ( v3 < 22 );
```

看来就是输入的一串字符串和某个东西异或之后得到 byte_140003480。那么进到 sub_140001430 函


数看看

```
1 __int64 sub_140001430()
2 {
3     CloseHandle((HANDLE)0xC001CAFEi64);
4     sub_140001290(50);
5     return (unsigned __int8)byte_140005044--;
6 }
```

是 byte_140005044 上的值与输入异或，并且依次减一。于是写解密代码：

```
#include<stdio.h>
int main()
{
    int a = 0xFF;
    char ch[] = {0x97,0x99, 0x9C, 0x91, 0x9E, 0x81, 0x91, 0x9D, 0x9B, 0x9A,
0x9A, 0x0AB, 0x81, 0x97, 0x0AE, 0x80, 0x83, 0x8F, 0x94, 0x89, 0x99, 0x97,0};
    for (int i=0;ch[i]!=0;i++,a--)
        putchar(ch[i] ^ a);
}
```

得flag



```
hgame{hello re player}
```

pypy

打开附件，是 python 的字节码，于是去 python 官网找到说明，大概看懂了，翻译一下：

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

于是解密

```

flag="30466633346f59213b4139794520572b45514d61583151576638643a"
ch=[]
for i in range(int(len(flag)/2)):
    ch.append(chr(int(flag[i*2:i*2+2],16)^i))
for i in range(int(len(ch)/2)):
    ch[2*i],ch[2*i+1]=ch[2*i+1],ch[2*i]
for i in range(len(ch)):
    print(ch[i],end="")

```

```
G00dj0&_H3r3-I$Y@Ur_$L@G!~!~
```

把这个看上去不像 flag 的东西包上 hgame{} 后竟然是正确的。。。

Crypto

まひと

打开文件，是莫斯密码，于是在线莫斯密码解密得

```

86 109 108 110 90 87 53 108 99 109 85 116 84 71 108 114 87 84 112 57 86 109 116 116 100 107 112 108 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 88 84 86 87 79 110
65 61 61]

```

每个数字都不大于 127，用 ASCII 码输出试试：

```

PS D:\vsc\test2> cd d:\vsc\test2\ ; if ($?) { clang main.c -o main.exe
Vm1nZW5lcmUtTGlraTp9VmttdkpiITFYdEF4ZSFocE0xe00rOXhxnJUTV90an5jUmc0eA==
PS D:\vsc\test2>

```

于是 base64 解密

```

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

```

有提示是 vigenere 加密，估计 Liki 是密钥，在线 vigenere 解密得

}KccnYt!1N1Ppu!zeE1{C+9pfrhLB_Fz~uGy4n，有大括号了！

于是栅栏解密（一开始是每组4，大括号刚好在两边，结果不正确）

```

}KccnYt!1N1Ppu!zeE1{C+9pfrhLB_Fz~uGy4n

```

每组字数

```

}!!Ch~K1z+LucNe9BGclEp_ynP1ff4Yp{rzntu

```

这看起来逆序以后应该就是 hgame 的形式，于是逆序之后做 vigenere 的已知字符串破解得 flag

Transformer

打开文件是一堆 enc 和 ori 文件，并且数量相同，应该是一一对应关系，于是使用正则表达式进行格式匹配，得到对应的字符表

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

, 于是对密文进行解密得:

```
The lift bridge console system has only used password login since 2003, the  
password is "hgame{ ea5y_f0r_fun ^ 3nd & he11o_ }",Don't forget to add the year  
at the end.
```

提交flag:

```
hgame{ ea5y_f0r_fun ^ 3nd & he11o_2020 }
```

Misc

Base全家福

对所给密文

```
R1k0RE10Wl dhRTNFSU5SVkc1QkRLTlpXR1VaVENOUlRHTVlETVJCV0dVMlVNTlpvR01ZREtSu1VIQTJE  
T01aVuDSQ0RHTVpWSVlaVEVNwlFHTVpER01kWE1RPT09PT09
```

一次 base64 解密:

```
GY4DMNZWGE3EINRVG5BDKNZWGUZTCNRTGMYDMRBWGU2UMNZUGMYDKRRUHA2DOMZUGRCDGMZVIYZTEMZQ  
GMZDGMJXIQ=====
```

再一次 base32 解密:

```
6867616D657B57653163306D655F74305F4847344D335F323032317D
```

再 base16 解密得:

```
hgame{we1c0me_t0_HG4M3_2021}
```

不起眼压缩包的培养的方法

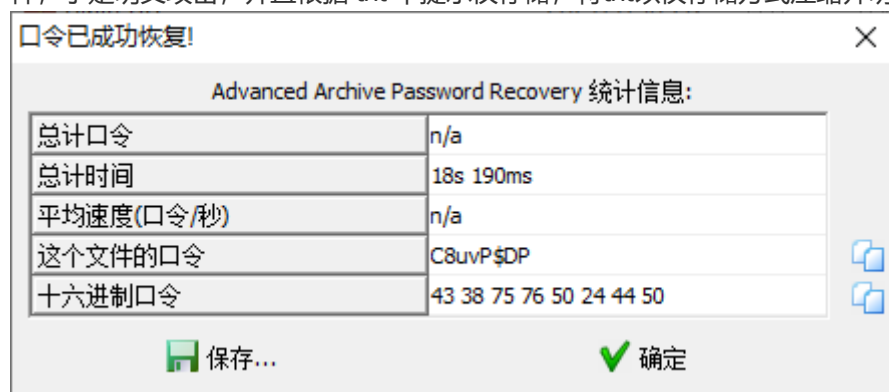
压缩包题，图片下载下来，到linux里binwalk出一个压缩包，有密码，看压缩包注释

注释

若密码是 picture ID (Up to 8 digits) 提示最多8位数字，暴力破解得



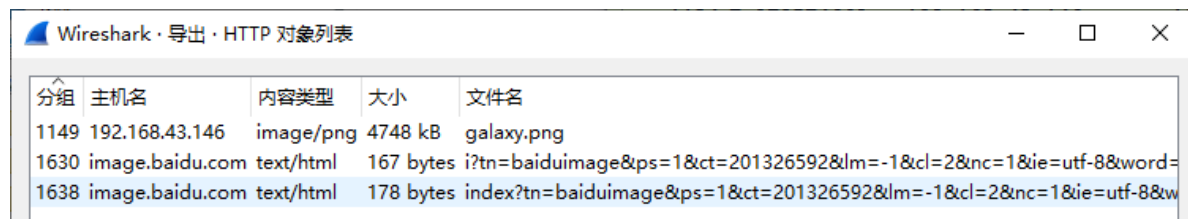
密码70415155，解压出来一个压缩包 plain 和一个 NO PASSWORD.txt，发现 plain 中也有该 txt 文件，于是明文攻击，并且根据 txt 中提示仅存储，将txt以仅存储方式压缩并明文攻击



得 flag 压缩包，没提示了，试一试 zip 伪加密，把两个加密位都改为未加密可正常解压得 flag

Galaxy

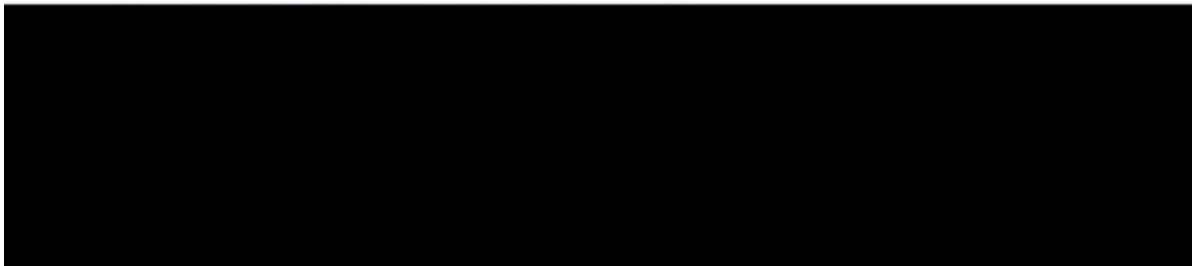
流量分析题，用 wireshark 打开，导出http对象



得到galaxy图片，修改图片高度



hgame{Wh4t_A_W0nderfu1_Wallpaper}



Word RE:MASTER

两个word文件，其中一个 word 有密码，打开无密码word发现没有啥有用的信息，去看2019年的wp发现有个word的xml隐写，于是用7z打开first.docx

名称	大小	压缩
media	28 028	2
theme	8 398	
_rels	950	
document.xml	10 103	
fontTable.xml	2 415	
password.xml	284	
settings.xml	3 353	
styles.xml	29 326	
webSettings.xml	4 282	

发现 password.xml , 打开发现一串奇怪的密码, 直接百度得出解密网站,

DOYOUKNOWHIDDEN?

Text to Ook!

Text to short Ook!

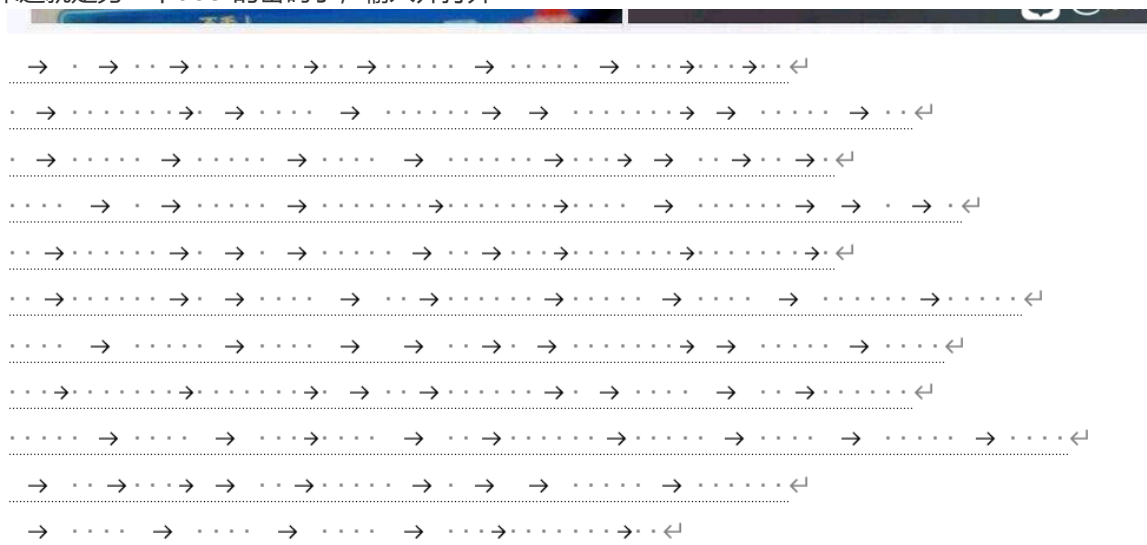
Ook! to Text

Text to Brainfuck

Brainfuck to Text

解密后是 DOYOUKNOWHIDDEN?

看来这就是另一个docx的密码了, 输入并打开



查了好久资料才知道这是snow加密, 于是下载小程序解密得解码得

```
hgame{Challen9e_whit3_P4ND0R4_P4R4D0XXX}
```