HGAME 2021 WP

WEB

Hitchhiking_in_the_Galaxy

进入题目。点击 "我要搭顺风车"，发现又跳转回当前网址(index.php)，观察右下角实际链接地址:

hitchhiker42.0727.site:42420/HitchhikerGuide.php

复制链接，使用接口调试工具 Talend API Tester， 按惯例先发个 POST 请求，提示要使用"无限非概率引擎"，显然要求加上UA，

```
User-Agent: Infinite Improbability Drive
```

再发包，提示:

```
    你知道吗？<a href="https://github.com/wuhan005">茄子</a>特别要求：你得从他的<a href="https://cardinal.ink/">Cardinal</a>过来
```

在headers里加上

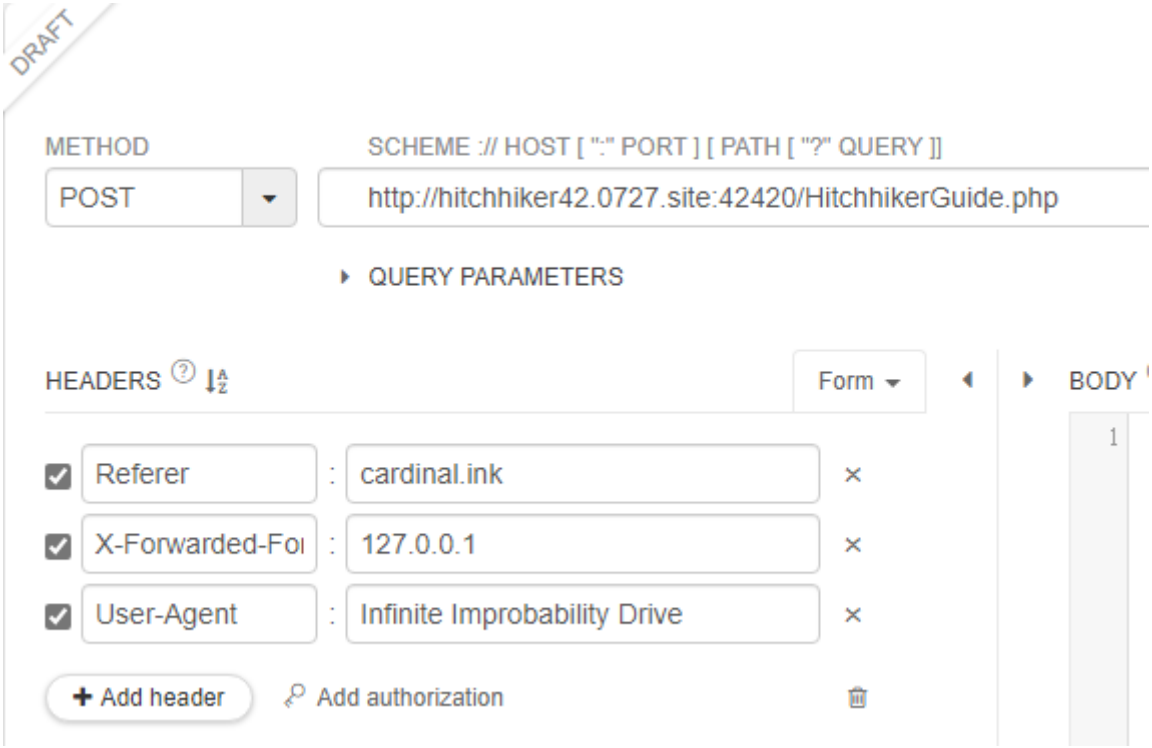
```
Referer: cardinal.ink
```

即可。返回页面要求本地访问， 百度"本地访问 ctf"， 可以知道要添加 XFF 参数伪装本地:

```
X-Forwarded-For：127.0.0.1
```

拿到 flag。

最终测试界面:



watermelon

使用了非常规方法: 直接玩到 2000分。

下面介绍一些提分技巧:

1. 使用电脑 chrome 系浏览器, 进入开发者工具, 使用模拟手机的功能, 拉长界面以获得更大的空间。
2. 计算下一个出现水果的类型调用了 js 的 `math.random` 函数,

   在控制台输入 `Math.Random = function() {return 2}` 重写该函数，适时修改返回的值以调出合适的水果，2000分还是挺快的。

宝藏走私者

好像搭了顺风车， 访问网页直接拿到了flag。。。

我是被迫的！！！

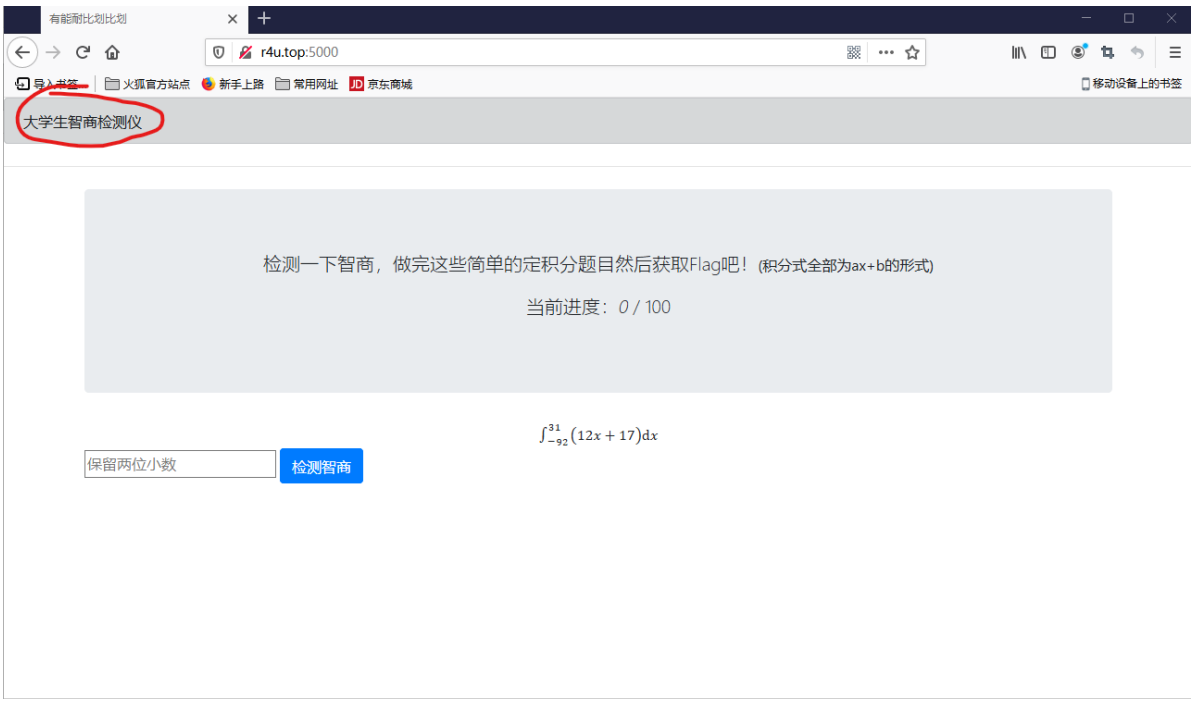
部分思路：第一个 hint 中提到了留意服务器信息，上 `Burp Suite`，在返回的包中可以找到

`Server: ATS/7.1.2` ，猜测与该版本 ATS 相关特性有关。拿到相关资料测试后 复现了 `CL-TE`

但是多次尝试走私都没有成功。

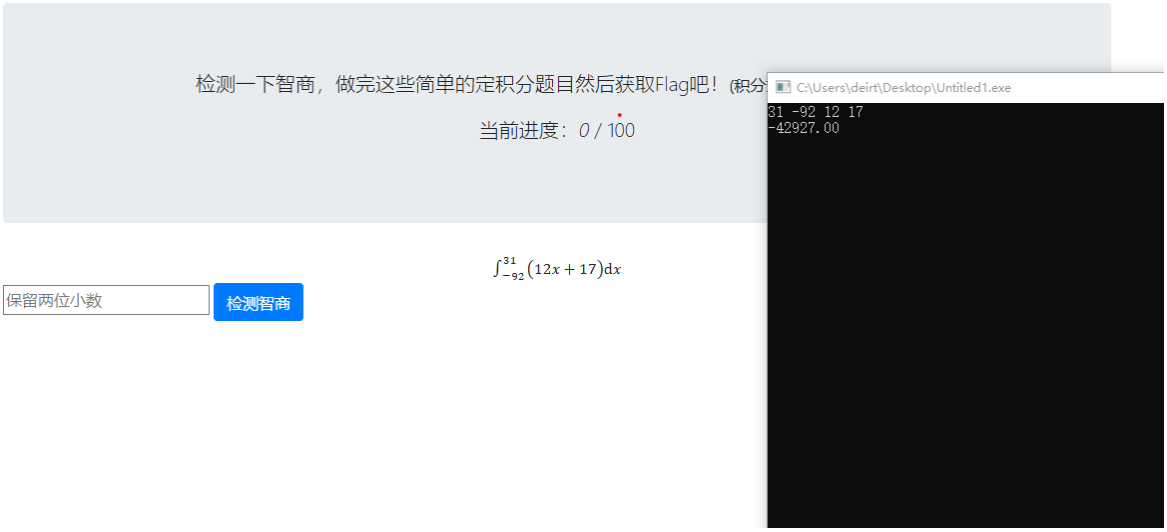
智商检测鸡

要求做完100道简单的高数题。



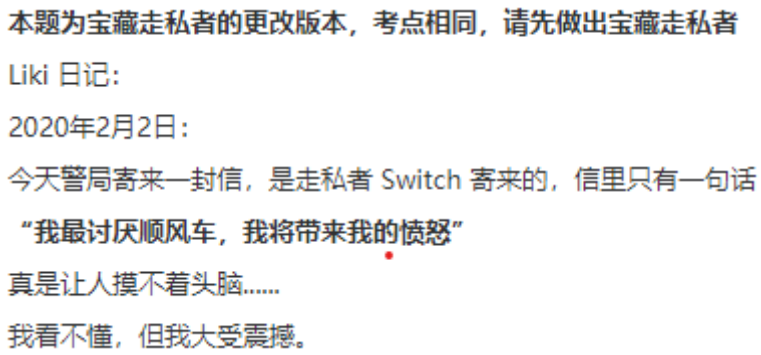
十分害怕，

这题第一考虑是设计 *Tampermonkey* 脚本全自动完成，然而智商不足（不会js），退求其次写了需要手动输参数，自动计算题给形式积分的c程序，大概是这样（代码见附件1）：



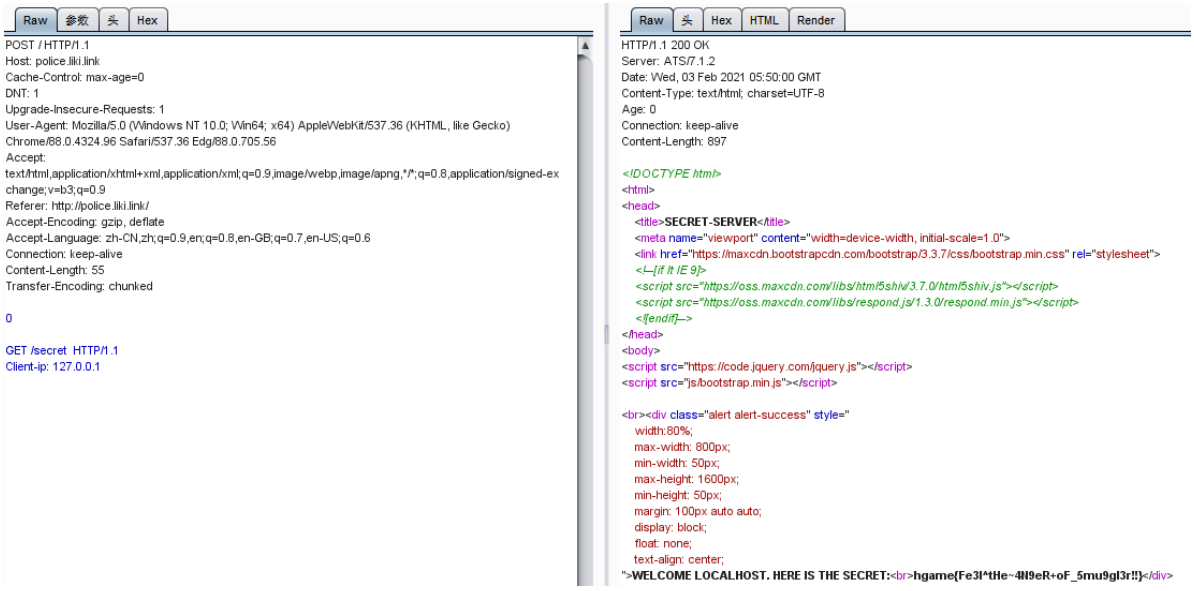
（做到一半增加了自动保存计算结果到剪贴板，希望能挽留我不存在的智商，以及颤抖的双手）

走私者的愤怒



我看不懂，但我也大受震撼，

解题过程：上同宝藏走私者，在用 burp 发包测试的过程中突然出现了一个flag，疑似顺风车，附图：



Crypto

Transformer

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 附件 md5:0340142700c8f63546368fa14fd6fb24

由题目容易知道需要恢复两份文件并比较, 获得文件的加密方式以解密得到 flag。

打开压缩包，源文件和加密后的文件分别放在两个文件夹里，用命令行简单拼接一下：

```
for /l %i in (0 1 240) do type part_%i >> a.txt
for /l %i in (0 1 240) do type enc_%i >> b.txt
```

拼接顺序有些混乱，问题不大，继续下一步。

观察文档，猜测标点符号和数字没有被改变和移位，

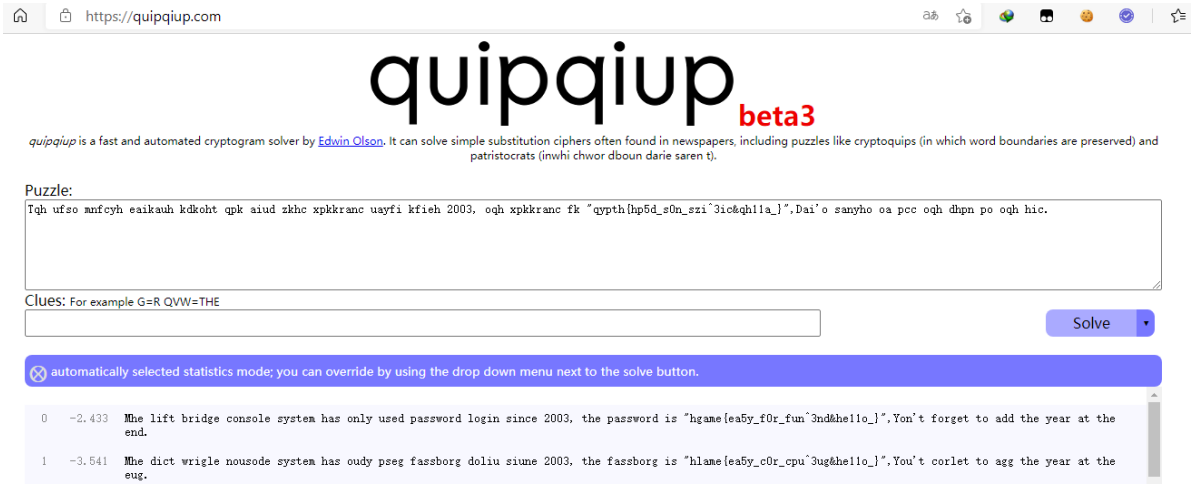
部分语句的比较图：

```
hkh fi 2015, fo qpk mhhi aih as oqh tako uavhc xnaynpttko rfoq azn CRLF
ese in 2015, it has been one of the most loved programmhe frontrunnCRLF
CRLF
e in y from go to rust at the end of 2019 so thahe browser and our CRLF
hpt qp snat ya oa nzko po oqh hic as 2019 ka oqpofvh xuposant ufmnpCRLF
```

分析后猜测为26字母一一对应替换的加密，接下来可以进行观察，手工对应后得到对应关系，

本题文本量足够大，应该也可以分别统计两份文本的字母频率，相同出现频率的字母即为对应字母。

这类一一对应的加密方式互联网上也有很多相关在线网站，可以辅助解决（



使用的相关网址：https://quipqiup.com/

MISC

Base全家福

看题目，简单的 base64 + base32 + base16 依次解密后即得到 flag。

使用了在线网站解密：http://ctf.ssleye.com/

不起眼压缩包的养成的方法

压缩包相关知识点，直接以压缩包方式打开图片，在注释中得到提示：

`“Password is picture ID (Up to 8 digits)”`

看完图~~（三次元，直奔P站）~~，使用百度搜图得到图片 ID，打开 plain.zip ，内有 “NO PASSWORD.txt” 和 “flag.zip”。

结合提示 `“NO PASSWORD”` 和 `“I only use storage”` 两个关键提示，使用winrar storage 方式压缩 “NO PASSWORD.txt” 后用 ARCHPR 进行明文攻击，拿到密钥得到 flag.zip。

伪加密，我使用了winhex 16进制编辑器修改加密位，打开后得到 flag。也可以丢 linux 直接打开。

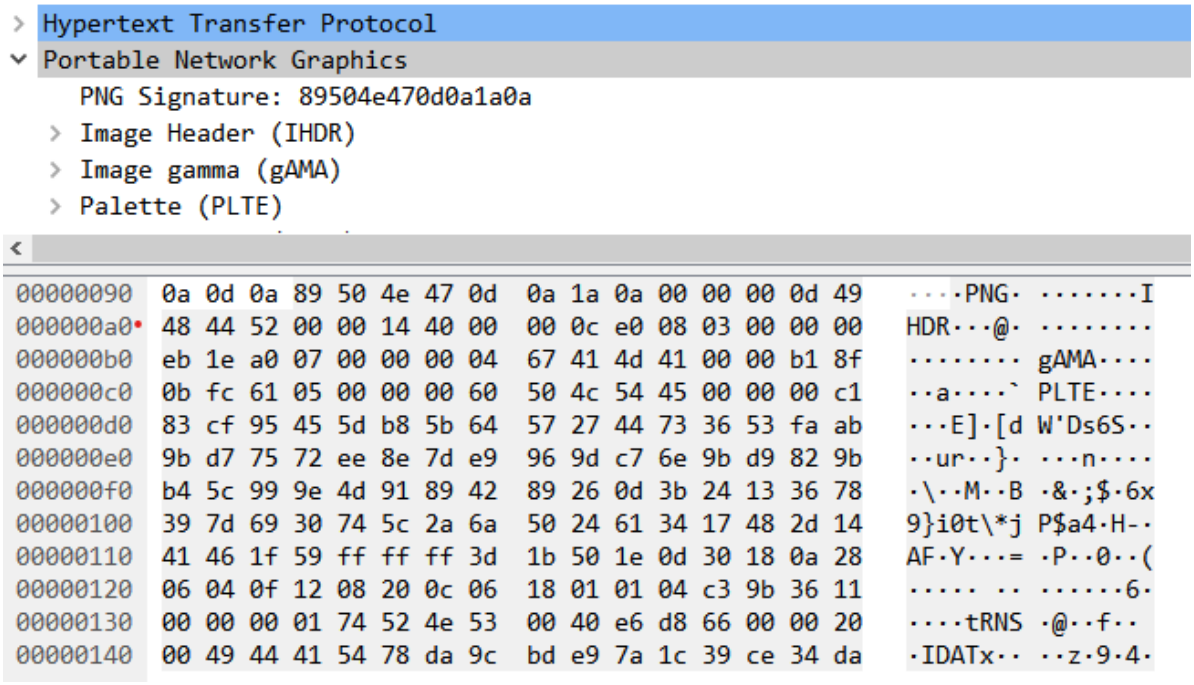
Galaxy

下载题目，先看到后缀为 pacpng，尝试改为 png 无法打开。

求助度娘后得知要使用 Wireshark 流量分析。

打开文件，按 info 排序，发现疑似目标：

```
192.168.43.199   192.168.43.146   HTTP    460 GET /galaxy.png HTTP/1.1
192.168.43.199   14.215.177.185   HTTP   1105 GET /i?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8&word=galaxy+wallpaper HTTP/1.1
192.168.43.199   14.215.177.185   HTTP   1167 GET /search/index?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8&word=galaxy+wallpaper HTTP/1.1
192.168.43.146   192.168.43.199   HTTP  30233 HTTP/1.1 200 OK  (PNG)
14.215.177.185   192.168.43.199   HTTP    469 HTTP/1.1 301 Moved Permanently  (text/html)
14.215.177.185   192.168.43.199   HTTP    491 HTTP/1.1 301 Moved Permanently  (text/html)
```



右键导出分组字节流，保存时添加 png 后缀。

得到一张星空图片，尝试修改长度后成功得到flag。

附件

附件1

```c
#include<stdio.h>
#include <string.h>
#include <windows.h>

double fun(double up, double down, double c1, double c2);
double fun_buding(double c, double c1, double c2);
void copy(char* str);

int main(){
    double up, down, c1, c2, result;
    char str[50];
    while(1){
        scanf("%lf%lf%lf%lf", &up, &down, &c1, &c2 );
        result = fun(up,down,c1,c2);
        sprintf(str,"%.2lf",result);
        printf("%s\n", str);
        copy(str);
    }
    return 0;
}

double fun(double up, double down, double c1, double c2){
    return fun_buding(up,c1,c2) - fun_buding(down,c1,c2);
}

double fun_buding(double c, double c1, double c2){
    return 0.5*c1*c*c + c2*c;
}

void copy(char* str){
```

```
    const char* output = str;
    const size_t len = strlen(output) + 1;
    HGLOBAL hMem =  GlobalAlloc(GMEM_MOVEABLE, len);
    memcpy(GlobalLock(hMem), output, len);
    GlobalUnlock(hMem);
    OpenClipboard(0);
    EmptyClipboard();
    SetClipboardData(CF_TEXT, hMem);
    CloseClipboard();
}
```