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

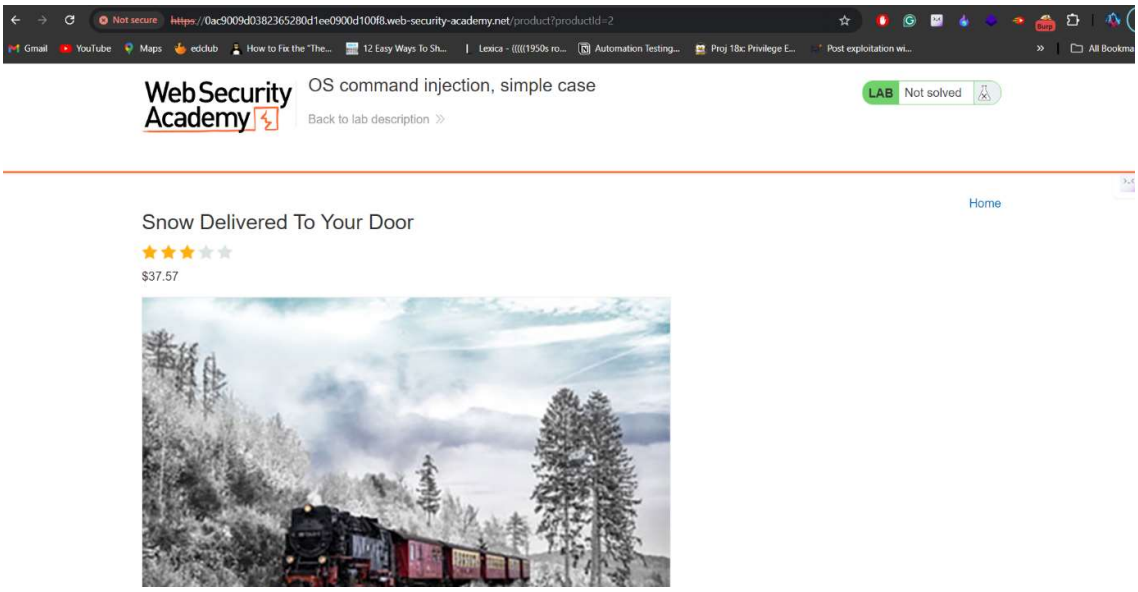
This lab focuses on an **OS command injection** vulnerability in a web application with functionality to check available stock. Using **Burp Suite**, I captured the requests made by the application and identified that it uses both **GET** and **POST** requests to retrieve data from the server.

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.

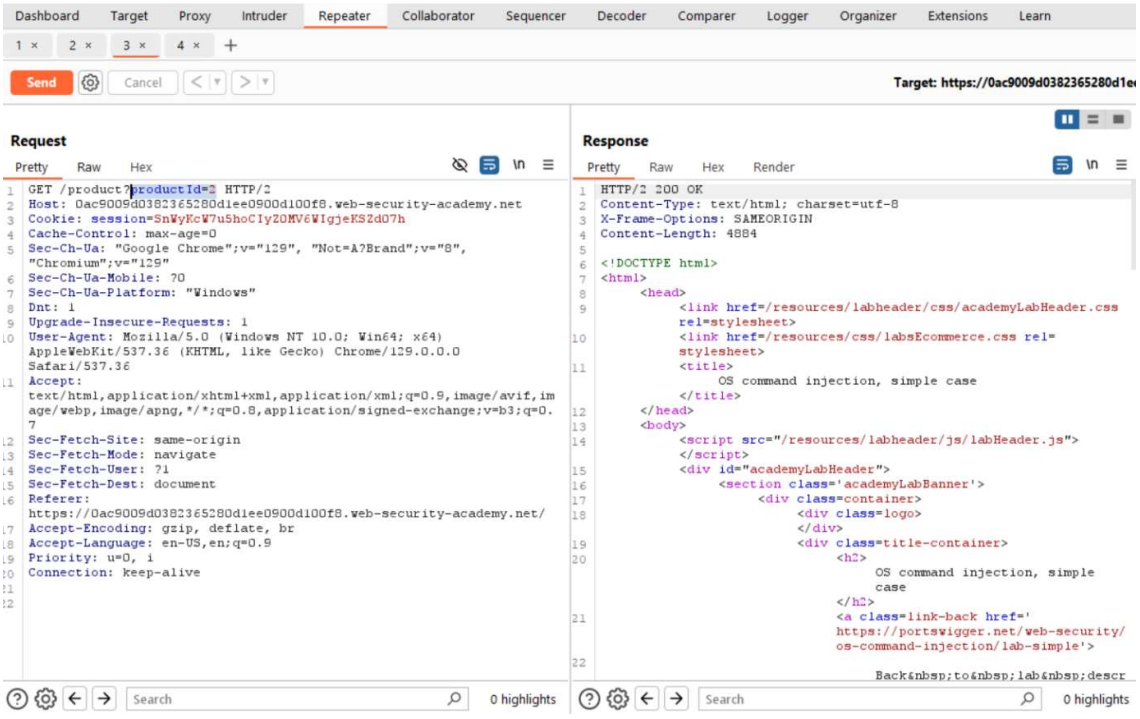
To test for an **OS command injection** vulnerability, I injected basic system commands into the parameters of the POST request. The server responded with the output of these commands, confirming the presence of the vulnerability in the application.

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

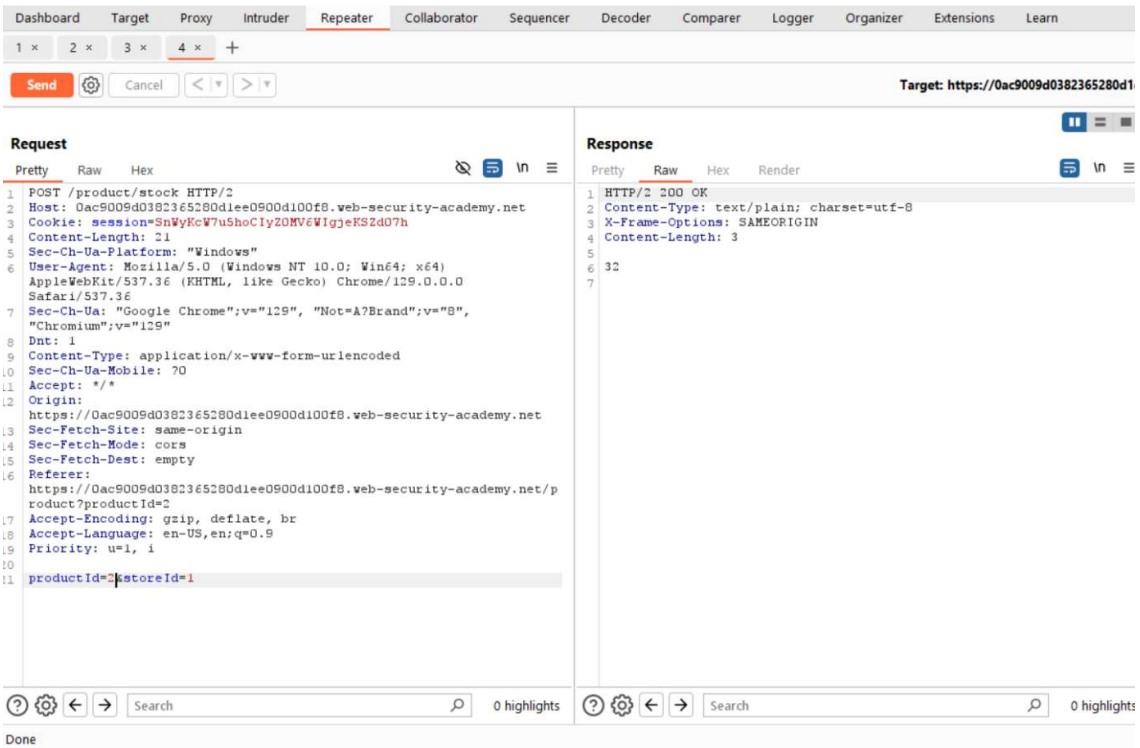
Application functionality:



Get Request:



Post Request:

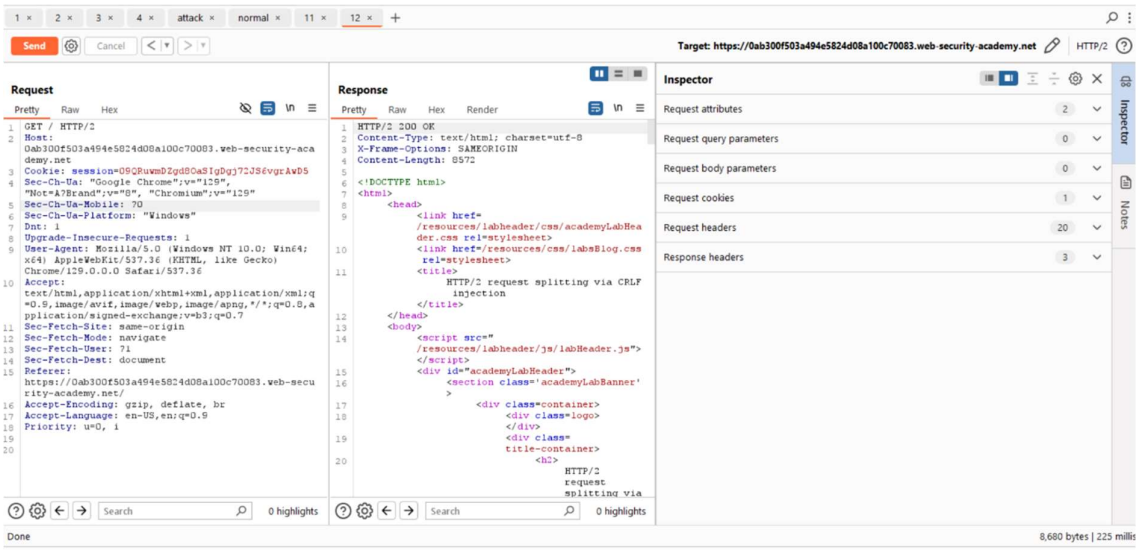


Lab 4: https://portswigger.net/web-security/request-smuggling/advanced/lab-request-smuggling-h2-request-splitting-via-crlf-injection

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:



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: