

VAPT on OWASP Juice Shop using Kali Linux

This project demonstrates a complete **Vulnerability Assessment and Penetration Testing (VAPT)** workflow using **Kali Linux** tools on the **OWASP Juice Shop**, a purposely vulnerable web application designed for security training. The goal is to simulate a real-world ethical hacking engagement, uncover security flaws, and document findings in a professional format.

What You'll Be Doing

You'll walk through each phase of a typical VAPT process:

1. **Lab Setup:** Deploy Juice Shop locally using Docker on Kali Linux.
2. **Reconnaissance:** Use tools like Nmap and WhatWeb to gather information about the target.
3. **Vulnerability Scanning:** Run Nikto and Burp Suite to identify weaknesses.
4. **Exploitation:** Perform manual attacks like SQL injection and explore automated options with Metasploit.
5. **Post-Exploitation:** Access sensitive data and understand the impact of successful attacks.
6. **Reporting:** Create a detailed report with screenshots, findings, and remediation suggestions.
7. **GitHub Deployment:** Push your project to GitHub to showcase your work to recruiters and peers.

Why This Matters

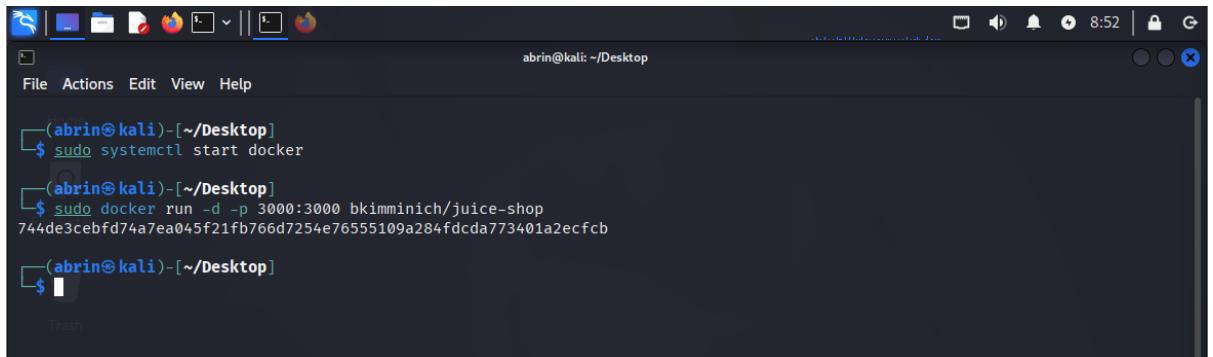
This hands-on project helps you:

- Understand how attackers think and operate
- Learn to use industry-standard tools
- Build a cybersecurity portfolio with real evidence
- Prepare for bug bounty programs or security certification

1. Step 1: Lab Setup

Install OWASP Juice Shop

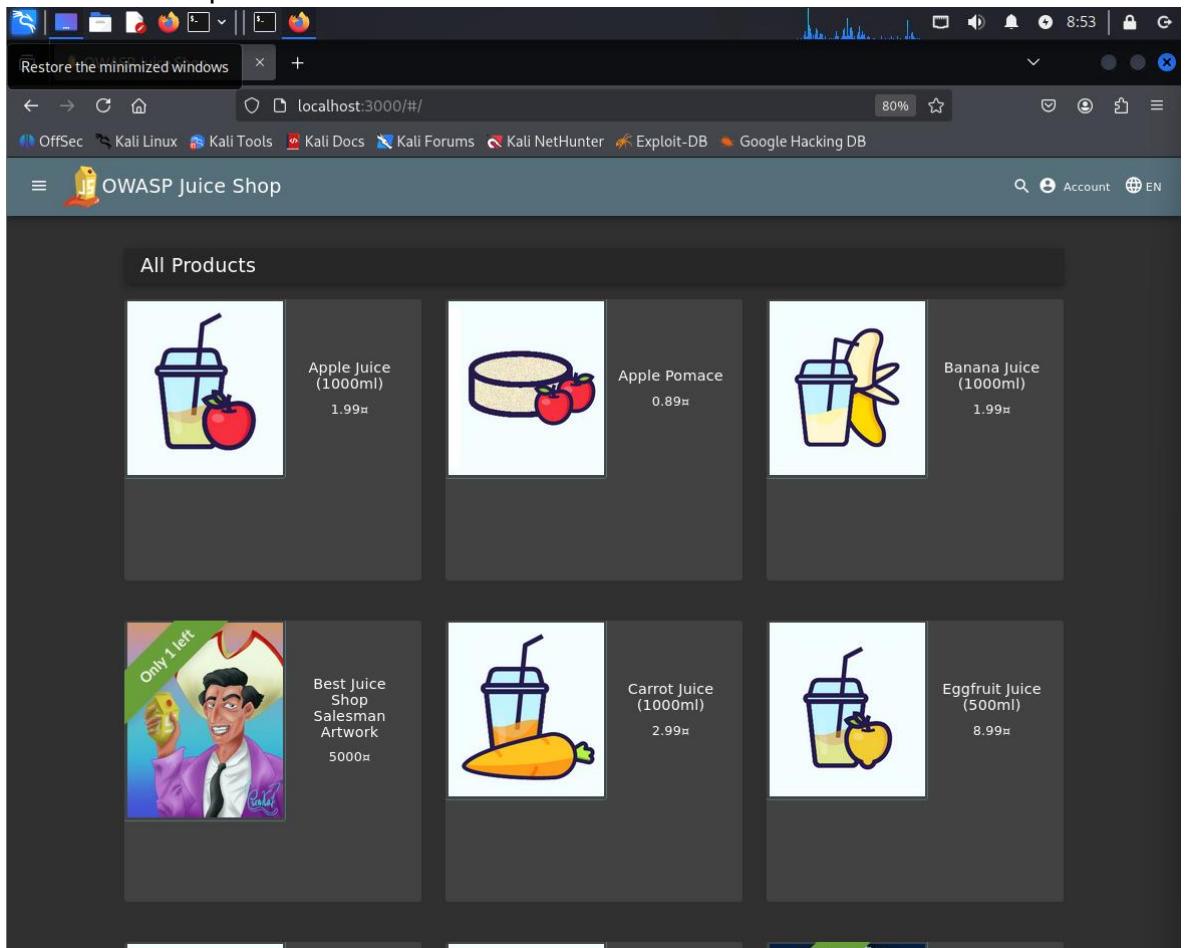
```
sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo docker pull bkimminich/juice-shop
sudo docker run -d -p 3000:3000 bkimminich/juice-shop
```



The terminal window shows the following command sequence:

```
(abrin@kali)-[~/Desktop]
$ sudo systemctl start docker
(abrin@kali)-[~/Desktop]
$ sudo docker run -d -p 3000:3000 bkimminich/juice-shop
744de3cebf74a7ea045f21fb766d7254e76555109a284fdcd773401a2ecfcba
```

Access it at <http://localhost:3000>



Step 2: Reconnaissance

Reconnaissance is the **first active phase** in penetration testing where you gather information about the target system. It helps you understand:

- What services are running
- Which ports are open
- What operating system and software versions are in use
- Potential entry points for exploitation

There are two types:

- **Passive Recon:** Observing without interacting (e.g., Google dorking, WHOIS lookup)
- **Active Recon:** Direct interaction with the target (e.g., Nmap scans)

In your Juice Shop project, you're doing **active recon** using tools like Nmap and WhatWeb.

Scan with Nmap

What is Nmap?

Nmap (Network Mapper) is a powerful open-source tool used to:

- Discover hosts and services on a network
- Identify open ports
- Detect service versions and operating systems
- Perform vulnerability scans

```

(abrin@kali:[~/Desktop]
$ nmap -sV -T4 -A localhost
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-21 08:54 IST | NetHunter | Exploit-DB | Google Hacking DB
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000025s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
3000/tcp   open  ppp?
| fingerprint-strings:
|   getRequest: 
|     HTTP/1.1 200 OK
|       Access-Control-Allow-Origin: *
|       X-Content-Type-Options: nosniff
|       X-Frame-Options: SAMEORIGIN
|       Feature-Policy: payment 'self'
|       X-Recruiting: /#/jobs
|       Accept-Ranges: bytes
|       Cache-Control: public, max-age=0
|       Last-Modified: Sun, 21 Sep 2025 03:21:11 GMT
|       ETag: W/"124fa-196a4a33d3"
|       Content-Type: text/html; charset=UTF-8
|       Content-Length: 75002
|       Vary: Accept-Encoding
|       Date: Sun, 21 Sep 2025 03:25:03 GMT
|       Connection: close
|       ←
|       Copyright (c) 2014-2025 Bjoern Kimminich & the OWASP Juice Shop contributors.
|       SPDX-License-Identifier: MIT
|       <!doctype html>
|       <html lang="en" data-beasties-container>
|       <head>
|       <meta charset="utf-8">
|       <title>OWASP Juice Shop</title>
|       <meta name="description" content="Probably the most modern and sophisticated insecure web application">
|       <meta name="viewport" content="width=device-width, initial-scale=1">
|       <link id="favicon" rel="icon" href="bestjuice"/>
|       HTTPOptions, RTSPRequest: 
|       HTTP/1.1 204 No Content
|       Access-Control-Allow-Origin: *
|       Artwork
|       Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
|       Vary: Access-Control-Request-Headers
|       Content-Length: 0
|       Date: Sun, 21 Sep 2025 03:25:03 GMT
|       Connection: close
|       Help, NCP:
|       HTTP/1.1 400 Bad Request
|       Connection: close

```

🔍 Command Breakdown:

`nmap -sV -T4 -A localhost`

<code>-sV</code>	Detects service versions (e.g., Apache 2.4.41)
<code>-T4</code>	Sets aggressive timing for faster scanning (T0 to T5; T4 is fast but still stable)
<code>-A</code>	Enables OS detection, version detection, script scanning, and traceroute
<code>localhost</code>	Target is your own machine (where Juice Shop is running)

💡 What You'll Learn from This Scan:

- Which ports are open (e.g., 3000 for Juice Shop)
- What services are running (e.g., Node.js, Express)
- OS fingerprinting results

- Potential vulnerabilities via Nmap scripts
-

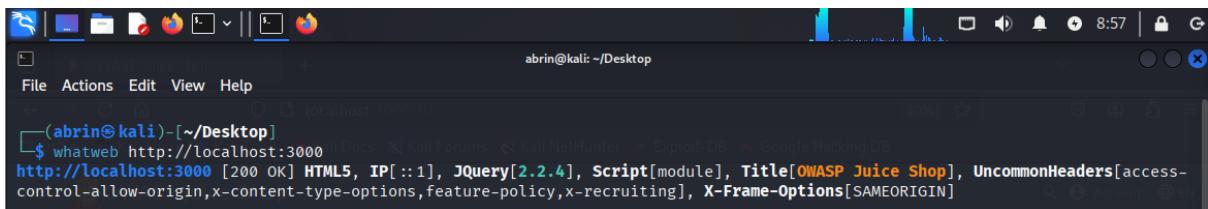
🔍 Fingerprint with WhatWeb

🔍 What is WhatWeb?

WhatWeb is a web scanner used for fingerprinting websites. It identifies technologies used by a web application, such as:

- Web servers (e.g., Express, Apache)
- Frameworks (e.g., Angular, React)
- CMS (e.g., WordPress, Joomla)
- Programming languages (e.g., PHP, Node.js)
- Analytics tools, cookies, and more

It's fast, lightweight, and perfect for reconnaissance.



The screenshot shows a terminal window on a Kali Linux desktop environment. The title bar says "abrin@kali: ~/Desktop". The terminal window displays the command \$ whatweb http://localhost:3000 followed by its output: http://localhost:3000 [200 OK] HTML5, IP[::1], JQuery[2.2.4], Script[module], Title[OWASP Juice Shop], UncommonHeaders[access-control-allow-origin,x-content-type-options,feature-policy,x-recruiting], X-Frame-Options[SAMEORIGIN].

💡 What Happens When You Run whatweb http://localhost:3000

You're scanning the Juice Shop running on your local machine. Here's what each part does:

whatweb	Launches the WhatWeb tool
http://localhost:3000	Targets the Juice Shop app running on port 3000

💡 What You'll Discover:

- **Server type:** Likely Node.js with Express
- **Cookies:** Session or tracking cookies
- **JavaScript libraries:** AngularJS, Bootstrap, etc.
- **Security headers:** CSP, X-Frame-Options, etc.
- **Other metadata:** HTML comments, analytics, etc.

Step 3: Vulnerability Assessment

🔍 Run Nikto

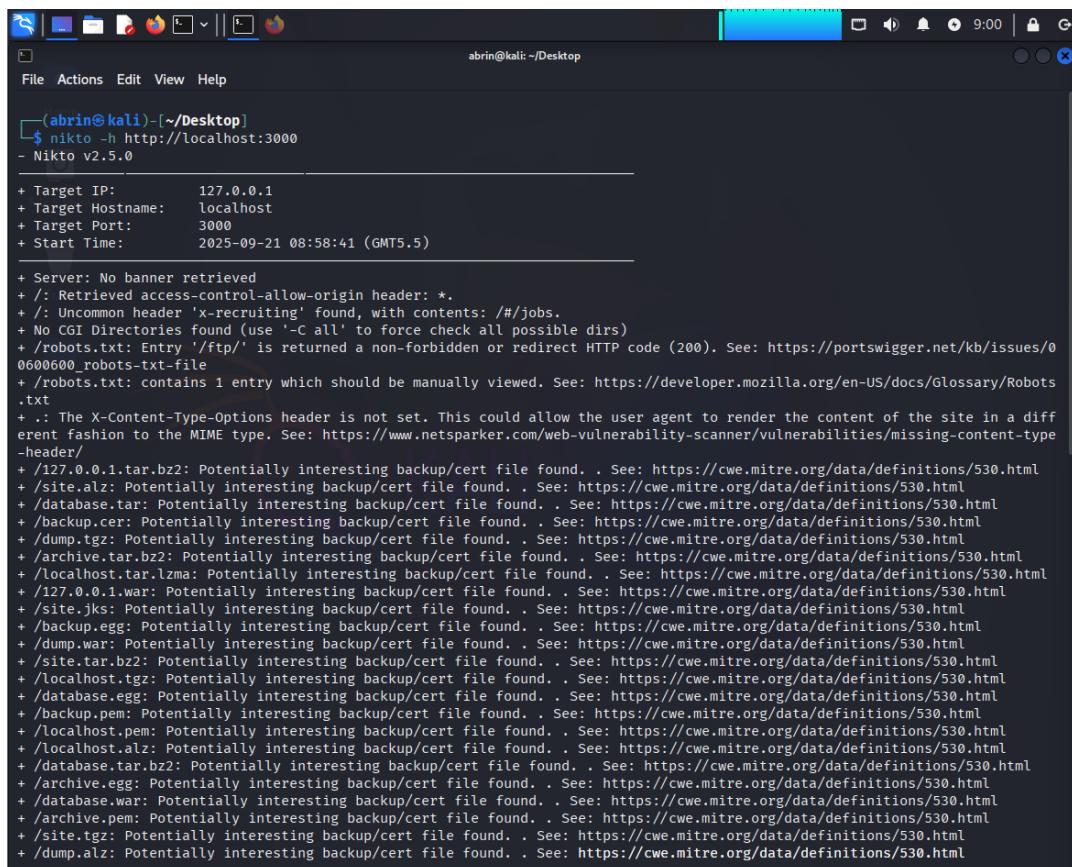
📌 What is Nikto?

Nikto is a web server vulnerability scanner that checks for:

- Dangerous files and directories
- Outdated server software
- Misconfigurations
- Security headers
- Known vulnerabilities

📝 What Happens in This Scan:

- Nikto sends multiple HTTP requests to Juice Shop
- It analyzes the server response for signs of weakness
- You'll get a report showing:
 - HTTP methods allowed (e.g., PUT, DELETE)
 - Missing security headers (e.g., X-Frame-Options)
 - Known vulnerabilities in the server software



The screenshot shows a terminal window on a Kali Linux desktop environment. The title bar says "abrin@kali: ~/Desktop". The terminal command is \$ nikto -h http://localhost:3000. The output shows the following details:

```
(abrin@kali)-[~/Desktop]
$ nikto -h http://localhost:3000
- Nikto v2.5.0

+ Target IP:      127.0.0.1
+ Target Hostname: localhost
+ Target Port:    3000
+ Start Time:    2025-09-21 08:58:41 (GMT5.5)

+ Server: No banner retrieved
+: Retrieved access-control-allow-origin header: *.
+/: Uncommon header 'x-recruiting' found, with contents: /#/jobs.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+/robots.txt: Entry '/ftp/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/0600600_robots-txt-file
+/robots.txt: contains 1 entry which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+.: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+/127.0.0.1.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/site.alz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/database.tar: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/backup.cer: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/dump.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/archive.tar.bzz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/localhost.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/127.0.0.1.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/site.jks: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/backup.egg: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/dump.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/site.tar.bzz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/localhost.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/database.egg: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/backup.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/localhost.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/localhost.alz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/database.tar.bzz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/archive.egg: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/database.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/archive.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/site.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+/dump.alz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
```

🔍 Use Burp Suite

🛠 Steps:

1. **Open Burp Suite** in Kali Linux
2. **Configure Browser Proxy:**
 - Set your browser's proxy to 127.0.0.1:8080
 - This routes traffic through Burp Suite
3. **Intercept Login Requests:**
 - Go to Juice Shop login page
 - Enter dummy credentials
 - Burp Suite will intercept the HTTP POST request
 - You can inspect:
 - Request headers
 - Parameters (e.g., username, password)
 - Cookies
 - Response from the server

🧠 Why This Matters:

- You can analyze how login data is sent
- Modify requests to test for SQL injection, XSS, etc.
- Understand session handling and authentication flow

The screenshot shows the Burp Suite interface with the following details:

- Top Bar:** Burp, Project, Intruder, Repeater, View, Help, Burp Suite Community Edition v2025.7.4 - Temporary Project.
- Menu Bar:** Dashboard, Target, **Proxy**, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, Learn.
- Sub-Menu Bar:** Intercept, HTTP history, WebSockets history, Match and replace, Proxy settings.
- Toolbar:** Intercept on, Forward, Drop, Open browser, Help.
- Request List:** Shows a list of captured requests with columns: Time, Type, Direction, Method, URL, Status code, Length. One request is highlighted: "Request to http://localhost:3000 [127.0.0.1]".
- Request Panel:** Displays the raw request details. The selected line is: "1 GET /api/Challenges/?name=Score%20Board HTTP/1.1".
- Inspector Panel:** Shows sections for Request attributes, Request query parameters, Request body parameters, Request cookies, and Request headers.
- Bottom Status Bar:** Event log, All issues, Memory: 117.8MB, Disabled, Right Ctrl.

Step 4: Exploitation

SQL Injection (Manual)

SQL Injection is a web security vulnerability that allows an attacker to interfere with the queries an application makes to its database. It happens when user input **is not properly sanitized**, allowing malicious SQL code to be executed.

The screenshot shows a browser window for the OWASP Juice Shop application at localhost:3000/#/login. The title bar says "OWASP Juice Shop". The main content is a "Login" form. In the "Email*" field, the value "' or 1=1 --" has been entered, which is a common SQL injection payload. The "Password*" field contains "password". Below the form, there's a link "Forgot your password?" and two buttons: "Log in" and "Remember me". At the bottom of the form, there's a link "Not yet a customer?". The entire screenshot is framed by a dark border.

🔍 Breakdown of the Payload:

'	Closes the original string in the SQL query
OR 1=1	A condition that is always true
--	SQL comment syntax — ignores the rest of the query (like password check)

What Happens Behind the Scenes:

Assume the original SQL query looks like this:

```
SELECT * FROM users WHERE username = '[input]' AND password = '[input]';
```

After injection, it becomes:

```
SELECT * FROM users WHERE username = '' OR 1=1 --' AND  
password = 'password';
```

- OR 1=1 makes the condition always true
- -- comments out the rest of the query
- The database returns **all users**, and the app logs you in as the first one

Why This Is Dangerous

- Bypasses authentication
- Grants unauthorized access
- Can lead to **data theft, account takeover, or full system compromise**

Step 5: Post Exploitation

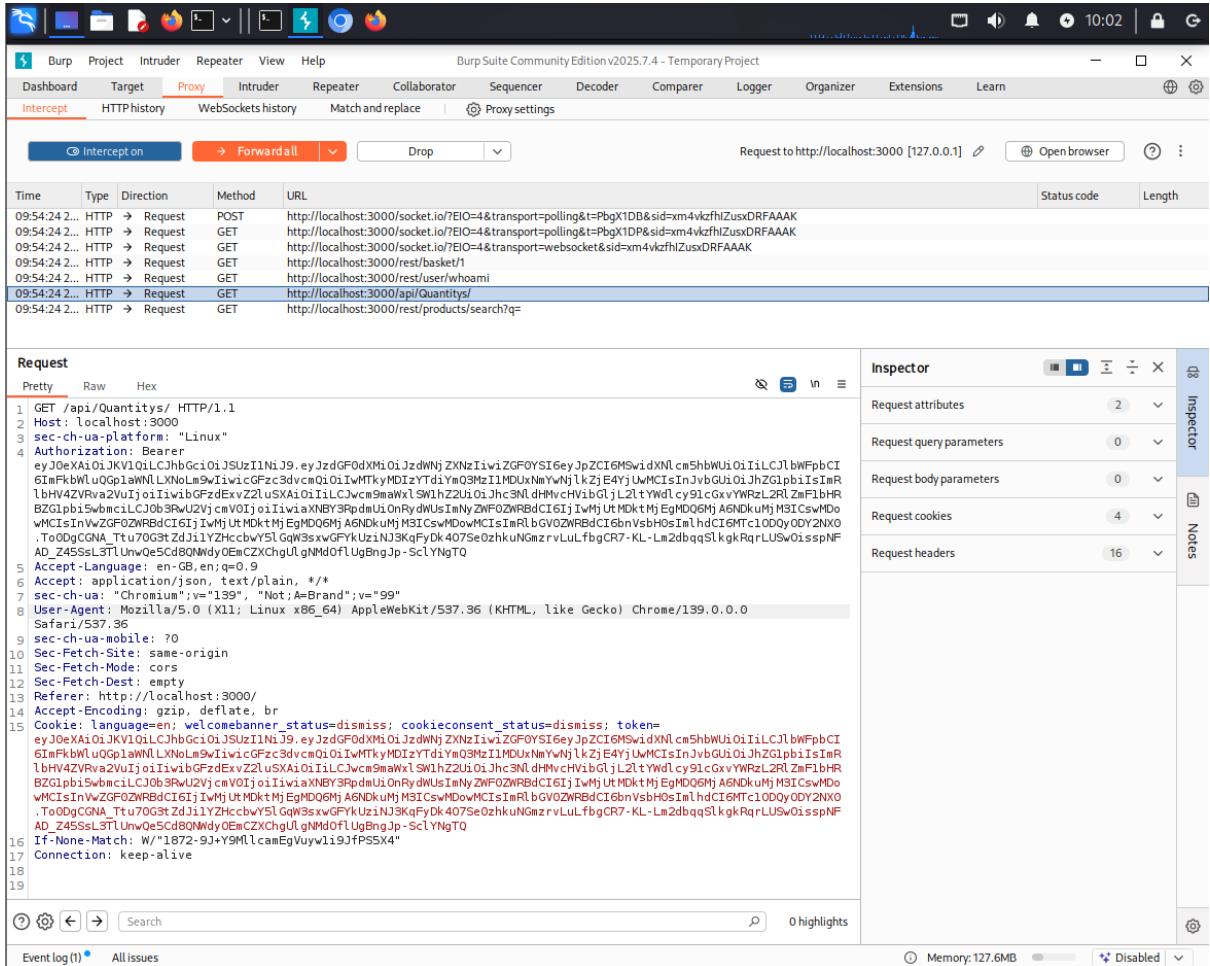
Access Sensitive Data

After bypassing login via SQL injection, you're technically logged in — but to interact with protected endpoints like **/rest/user/whoami**, you need to prove your identity to the server. That's where the session token comes in.

What Is a Session Token?

When you log in, the server creates a **session token** and sends it to your browser as a **cookie**. This token:

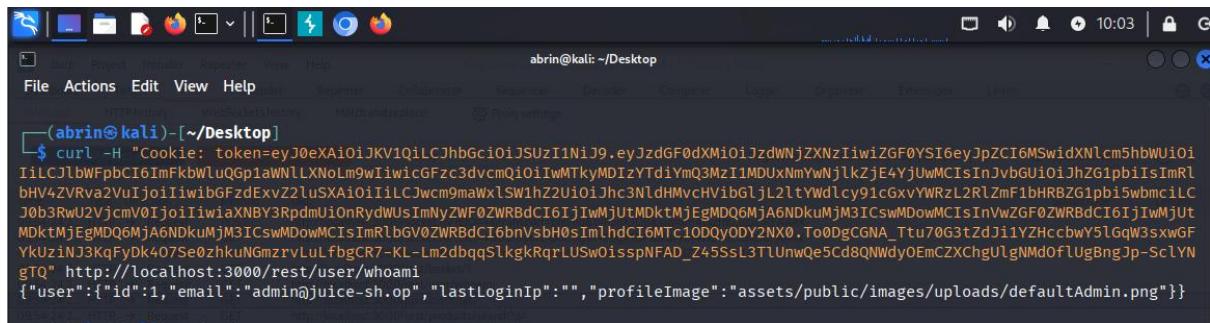
- Identifies you as an authenticated user
- Is stored in your browser
- Is sent with every request to protected resources



The screenshot shows the Burp Suite interface with the "Proxy" tab selected. The "Intercept" button is active. The "Request" pane displays a captured HTTP request to `http://localhost:3000/api/Quantities/`. The "Inspector" pane on the right shows the "Request headers" section, which includes the session token `token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9` in the `Authorization` header. The "Raw" tab of the request pane shows the full HTTP message, including the session token in the Authorization header.

```
1 GET /api/Quantities/ HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua-platform: "Linux"
4 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdF0dMjQ0IiJsdWNjZKNzIiwiZGF0YSI6eyJpZCfGMswidxNlcm5hbWUiOiIiLCJlbiWfpbCI6InFkbWluGp1wNLXN0Lm9wIiwiGFzC3dvcmQ1QiIwMTkyMDIxY7dsYnQ3MzIIMDUxNmVwNjlkZjE4YjUwMCIsInJvbGUiOiJzZ1pbIsImR1bHV4ZVRva2ViujiiwibGFzdExvZ2lusuSAiOiIiLCJwc9maWx1SW1hZ2UuOljhcnVwHvhGljL2ltYWdlcy9lcvxYRzL2RLZmF1bHRBZGJpb1SwbmcuCiCjObt9RwU2VjcmVOIjoiIiwiXaNBY3RpdmlUsOnRydwUsInNyZWF0ZWRBdCI6Ij1wMjUtMDktMjEgMD06MjA6NDkuMjM3ICsVMDoWHCisInVwZGF0ZWRBdCI6Ij1wMjUtMDktMjEgMD06MjA6NDkuMjM3ICsVMDoMCIsInRlBV0ZWRBdCI6bnVshHo5ImhdcI6Mtc10D0yODY2NKO .To0dgCGNA_Ttu70GStZdjiYZHccbwY5lGqW3sxwGFYkUz1Nj3KqFyDk407se0zhkuNGmzrvLuLfbgCR7-KL-Lm2dbqqSLkgkRqrLUSw0issppNF AD_Z45SsL3TUnw0e5Cd80NWyd0EmCZXCchgUlgnMd0fUgBngJp-SclYNgTQ
5 Accept-Language: en-GB;en;q=0.9
6 Accept: application/json, text/plain, */*
7 sec-ch-ua: "Chromium";v="139", "Not=A[Brand]";v="99"
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0
9 Safari/537.39
10 sec-ch-ua-mobile: ?0
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate, br
16 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dissmiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdF0dMjQ0IiJsdWNjZKNzIiwiZGF0YSI6eyJpZCfGMswidxNlcm5hbWUiOiIiLCJlbiWfpbCI6InFkbWluGp1wNLXN0Lm9wIiwiGFzC3dvcmQ1QiIwMTkyMDIxY7dsYnQ3MzIIMDUxNmVwNjlkZjE4YjUwMCIsInJvbGUiOiJzZ1pbIsImR1bHV4ZVRva2ViujiiwibGFzdExvZ2lusuSAiOiIiLCJwc9maWx1SW1hZ2UuOljhcnVwHvhGljL2ltYWdlcy9lcvxYRzL2RLZmF1bHRBZGJpb1SwbmcuCiCjObt9RwU2VjcmVOIjoiIiwiXaNBY3RpdmlUsOnRydwUsInNyZWF0ZWRBdCI6Ij1wMjUtMDktMjEgMD06MjA6NDkuMjM3ICsVMDoWHCisInVwZGF0ZWRBdCI6Ij1wMjUtMDktMjEgMD06MjA6NDkuMjM3ICsVMDoMCIsInRlBV0ZWRBdCI6bnVshHo5ImhdcI6Mtc10D0yODY2NKO .To0dgCGNA_Ttu70GStZdjiYZHccbwY5lGqW3sxwGFYkUz1Nj3KqFyDk407se0zhkuNGmzrvLuLfbgCR7-KL-Lm2dbqqSLkgkRqrLUSw0issppNF AD_Z45SsL3TUnw0e5Cd80NWyd0EmCZXCchgUlgnMd0fUgBngJp-SclYNgTQ
17 If-None-Match: W/"1872-90-j9MllcamEgVuyw119JfPS5X4"
18 Connection: keep-alive
19
```

The "Event log" and "All issues" panes at the bottom show no recent activity.



```
(abrin@kali)-[~/Desktop]
$ curl -H "Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwizGF0YST6eyJpZCI6MSwidXNlcm5hbWUiOiiLCJlbwFpbC16ImFkbWluQp1aWnLLXNoLm9wIwiCgFzc3dvcmQiOIMtkyMD1zYTDiyMq3MzI1MDUXNmYwNjlkZje4YjuwMCIsInjbGUoijhZG1pbisimRlbHv4ZVRva2VujoiliwibGfzdexv2luSXAiOiiilCJwcm9maWxlsW1hZ2UiOijhc3NldHMvchVibG1jL2ltYWdlcy9icGxvYWRzL2RlZmf1bHRBZG1pbis5wbmcilCJ0b3RwU2VjcmV0IjoiliwlaXNBY3RpdmUiOnRydWUsImNyZWFOZWRBdCI6IjIwMjUtMDktMjEgMDQ6MjA6NDkuMjM3ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUTMDktMjEgMDQ6MjA6NDkuMjM3ICswMDowMCIsImRlbGV0ZWRBdCI6bnYsbH0sImlhcdCI6MTc1ODQyODY2NX0.TooDgCGNA_Ttu70G3tZdj1iYZHccbWY5lgQW3sxwGFYKuZiN3KqFyDk407Se0zhkuNGmzrvLuLfbgCR7-KL-Lm2dbqqSlkgkRqrLUSw0isspNFAD_Z45SsL3TLUnwQe5Cd8QNwdy0EmCZxChgUlgNMdOfUgBngJp-SclYN gTQ" http://localhost:3000/rest/user/whoami
{"user":{"id":1,"email":"admin@juice-sh.op","lastLoginIp":"","profileImage":"assets/public/images/uploads/defaultAdmin.png"}}
```

This confirms you're authenticated and can now:

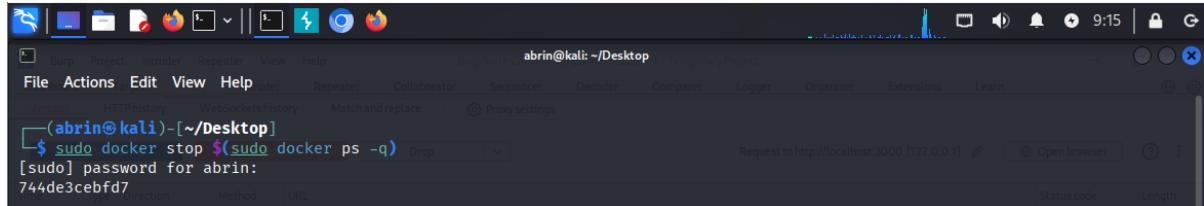
- Access user-specific data
- Explore other protected endpoints
- Simulate privilege escalation

Remediation

- Implement input validation and parameterized queries
- Secure API endpoints with authentication and authorization
- Add proper security headers to HTTP responses
- Regularly update and patch server software

Step 6: Cleanup

Stop Docker container:



Step 7: Reporting

report.md with the following sections:

📌 Scope

- Perform a VAPT on OWASP Juice Shop using Kali Linux
- Identify and exploit vulnerabilities in a controlled environment

🛠 Tools Used

- Kali Linux
- Docker
- Nmap
- WhatWeb
- Nikto
- Burp Suite
- Metasploit
- Curl

📸 Screenshots

- Juice Shop homepage
- Nmap scan results
- WhatWeb fingerprinting
- Nikto vulnerability scan
- Burp Suite intercepted login
- SQL injection success
- Sensitive data or order history via API