# Experiment - 8

## Full Web Application Penetration Test

### Objective

To identify, exploit, and document web application vulnerabilities in the OWASP Juice Shop using OWASP ZAP.

### Prerequisites

- Environment Setup:
- Install OWASP Juice Shop: Follow the official installation guide.
- Install OWASP ZAP: Download from OWASP ZAP website.
- Install supporting tools:
- Browser with OWASP ZAP plugin or proxy settings configured.
- Burp Suite (optional for additional validation).

### 2. System Requirements:

o A system with Docker installed (for running Juice Shop).

o Sufficient permissions to configure network settings for proxy.

o At least 8GB RAM and 20GB storage recommended.

**Step 1: Cloning the Juice Shop Repository**

**Heading:**

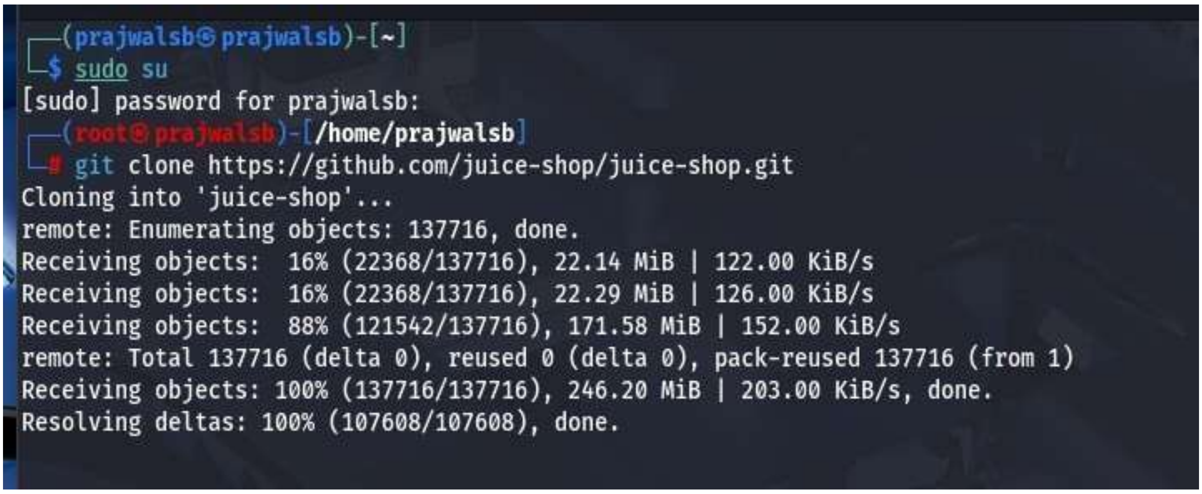
Setting Up the Vulnerable Web Application **Brief Explanation:**

The first screenshot shows the GitHub repository of OWASP Juice Shop, an intentionally vulnerable web application used for security training. The subsequent screenshots document the process of cloning this repository to a local machine using the git clone command. This step is crucial as it downloads the application's source code, allowing for local deployment and testing.

**Lab Observation:**

- The repository contains multiple branches and tags, indicating various versions and fixes.
- The cloning process completes successfully, downloading approximately 246 MB of data.

### Step 2: Installing Dependencies



```
┌──(prajwalsb㉿prajwalsb)-[~]
└─$ sudo su
[sudo] password for prajwalsb:
┌──(root㉿prajwalsb)-[/home/prajwalsb]
└─# git clone https://github.com/juice-shop/juice-shop.git
Cloning into 'juice-shop'...
remote: Enumerating objects: 137716, done.
Receiving objects:  16% (22368/137716), 22.14 MiB | 122.00 KiB/s
Receiving objects:  16% (22368/137716), 22.29 MiB | 126.00 KiB/s
Receiving objects:  88% (121542/137716), 171.58 MiB | 152.00 KiB/s
remote: Total 137716 (delta 0), reused 0 (delta 0), pack-reused 137716 (from 1)
Receiving objects: 100% (137716/137716), 246.20 MiB | 203.00 KiB/s, done.
Resolving deltas: 100% (107608/107608), done.
```

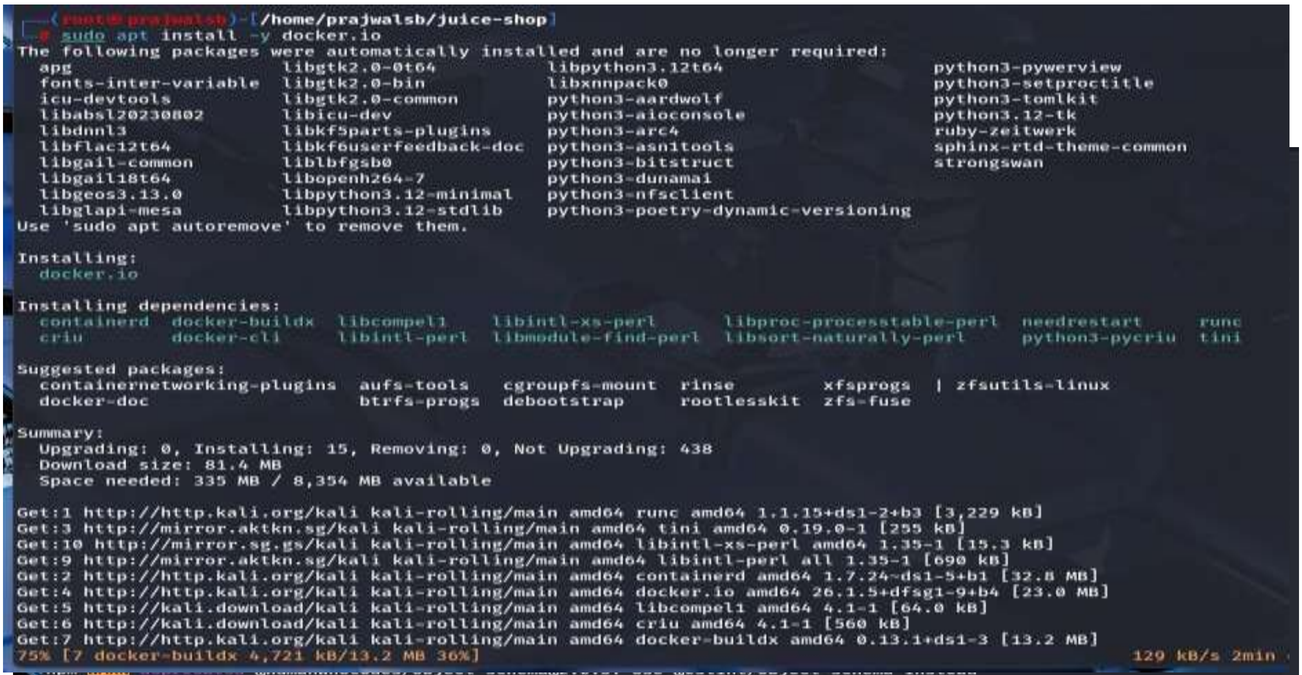
**Heading:**

Resolving Dependencies for Juice
Shop **Brief Explanation:**

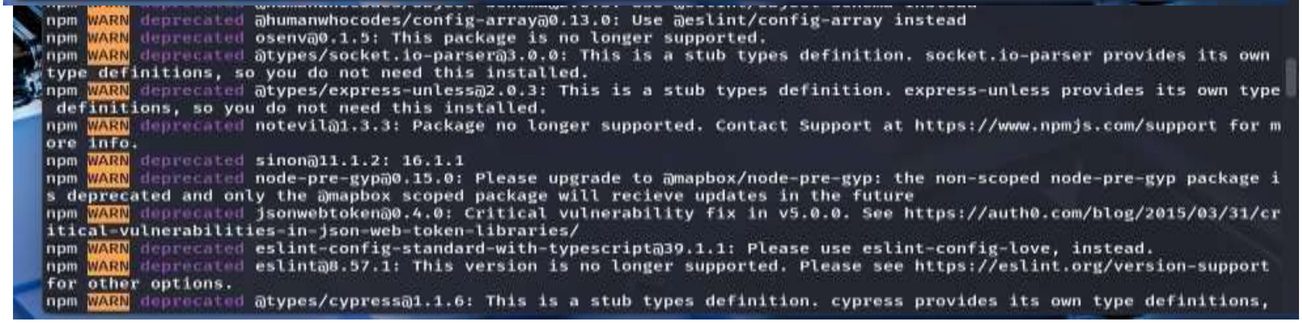
After cloning, the npm install command is executed to install the necessary dependencies for Juice Shop. The output shows numerous deprecated packages and security warnings, highlighting outdated or vulnerable components. This is intentional, as Juice Shop is designed to demonstrate security flaws.

**Lab Observation:**

- Multiple deprecated packages (e.g., inflight, fstream) are flagged, some with critical security issues.



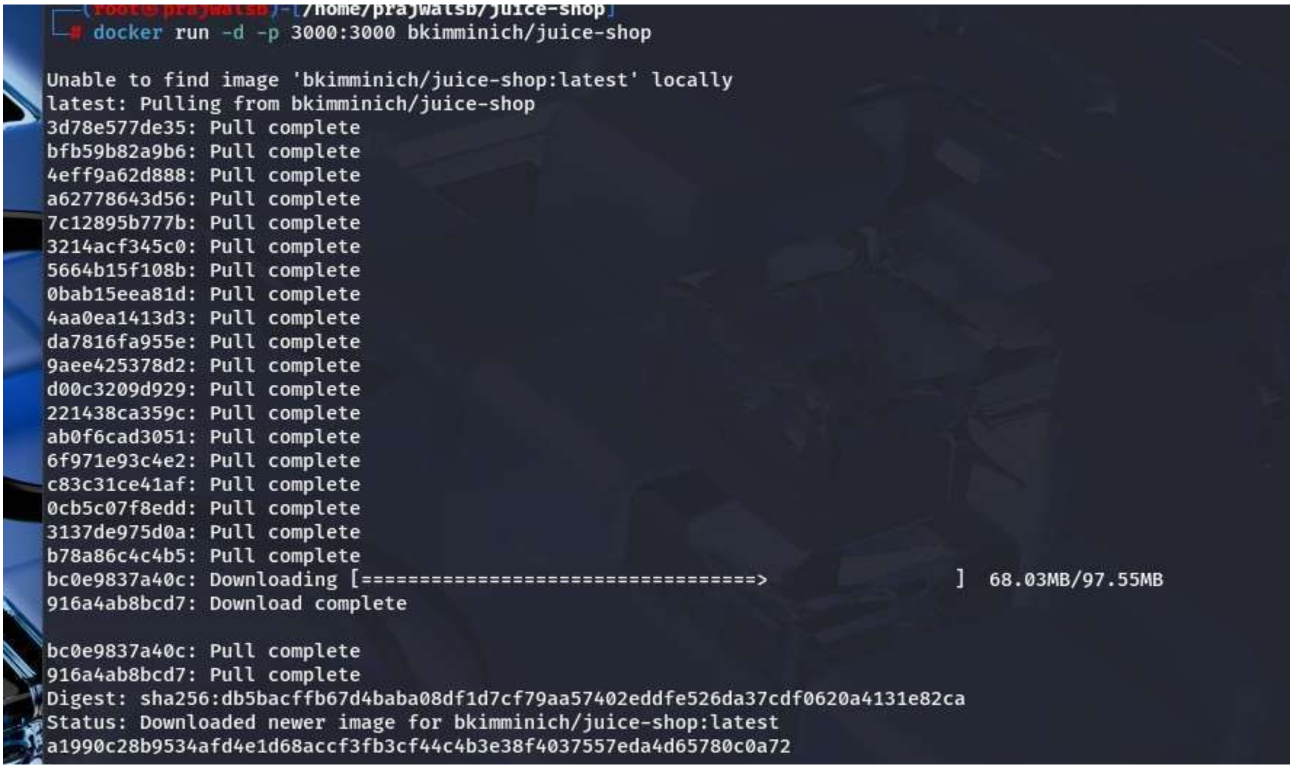
```
┌──(root@prajwalsb)-[/home/prajwalsb/juice-shop]
└─# sudo apt install -y docker.io
The following packages were automatically installed and are no longer required:
  apg                     libgtk2.0-0t64           libpython3.12t64                   python3-pywerview
  fonts-inter-variable    libgtk2.0-bin            libxnnpack0                        python3-setproctitle
  icu-devtools            libgtk2.0-common         python3-aardwolf                   python3-tomlkit
  libabsl20230802         libicu-dev               python3-aioconsole                 python3.12-tk
  libdnnl3                libkf5parts-plugins      python3-arc4                       ruby-zeitwerk
  libflac12t64            libkf6userfeedback-doc   python3-asn1tools                  sphinx-rtd-theme-common
  libgail-common          liblbfgsb0               python3-bitstruct                  strongswan
  libgail18t64            libopenh264-7            python3-dunamai
  libgeos3.13.0           libpython3.12-minimal    python3-nfsclient
  libglapi-mesa           libpython3.12-stdlib     python3-poetry-dynamic-versioning
Use 'sudo apt autoremove' to remove them.

Installing:
  docker.io

Installing dependencies:
  containerd  docker-buildx  libcompel1    libintl-xs-perl      libproc-processtable-perl  needrestart     runc
  criu        docker-cli     libintl-perl  libmodule-find-perl  libsort-naturally-perl     python3-pycriu  tini

Suggested packages:
  containernetworking-plugins  aufs-tools   cgroupfs-mount  rinse        xfsprogs  | zfsutils-linux
  docker-doc                   btrfs-progs  debootstrap     rootlesskit  zfs-fuse

Summary:
  Upgrading: 0, Installing: 15, Removing: 0, Not Upgrading: 438
  Download size: 81.4 MB
  Space needed: 335 MB / 8,354 MB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 runc amd64 1.1.15+ds1-2+b3 [3,229 kB]
Get:3 http://mirror.aktkn.sg/kali kali-rolling/main amd64 tini amd64 0.19.0-1 [255 kB]
Get:10 http://mirror.sg.gs/kali kali-rolling/main amd64 libintl-xs-perl amd64 1.35-1 [15.3 kB]
Get:9 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libintl-perl all 1.35-1 [690 kB]
Get:2 http://http.kali.org/kali kali-rolling/main amd64 containerd amd64 1.7.24~ds1-5+b1 [32.8 MB]
Get:4 http://http.kali.org/kali kali-rolling/main amd64 docker.io amd64 26.1.5+dfsg1-9+b4 [23.0 MB]
Get:5 http://kali.download/kali kali-rolling/main amd64 libcompel1 amd64 4.1-1 [64.0 kB]
Get:6 http://kali.download/kali kali-rolling/main amd64 criu amd64 4.1-1 [560 kB]
Get:7 http://http.kali.org/kali kali-rolling/main amd64 docker-buildx amd64 0.13.1+ds1-3 [13.2 MB]
75% [7 docker-buildx 4,721 kB/13.2 MB 36%]                                                   129 kB/s 2min
```



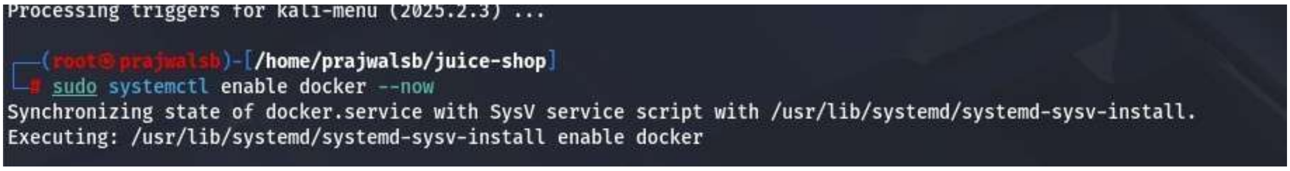
```
npm WARN deprecated @humanwhocodes/config-array@0.13.0: Use @eslint/config-array instead
npm WARN deprecated osenv@0.1.5: This package is no longer supported.
npm WARN deprecated @types/socket.io-parser@3.0.0: This is a stub types definition. socket.io-parser provides its own
type definitions, so you do not need this installed.
npm WARN deprecated @types/express-unless@2.0.3: This is a stub types definition. express-unless provides its own type
 definitions, so you do not need this installed.
npm WARN deprecated notevil@1.3.3: Package no longer supported. Contact Support at https://www.npmjs.com/support for m
ore info.
npm WARN deprecated sinon@11.1.2: 16.1.1
npm WARN deprecated node-pre-gyp@0.15.0: Please upgrade to @mapbox/node-pre-gyp: the non-scoped node-pre-gyp package i
s deprecated and only the @mapbox scoped package will recieve updates in the future
npm WARN deprecated jsonwebtoken@0.4.0: Critical vulnerability fix in v5.0.0. See https://auth0.com/blog/2015/03/31/cr
itical-vulnerabilities-in-json-web-token-libraries/
npm WARN deprecated eslint-config-standard-with-typescript@39.1.1: Please use eslint-config-love, instead.
npm WARN deprecated eslint@8.57.1: This version is no longer supported. Please see https://eslint.org/version-support
for other options.
npm WARN deprecated @types/cypress@1.1.6: This is a stub types definition. cypress provides its own type definitions,
```

- The warnings suggest the application includes outdated dependencies for educational purposes.

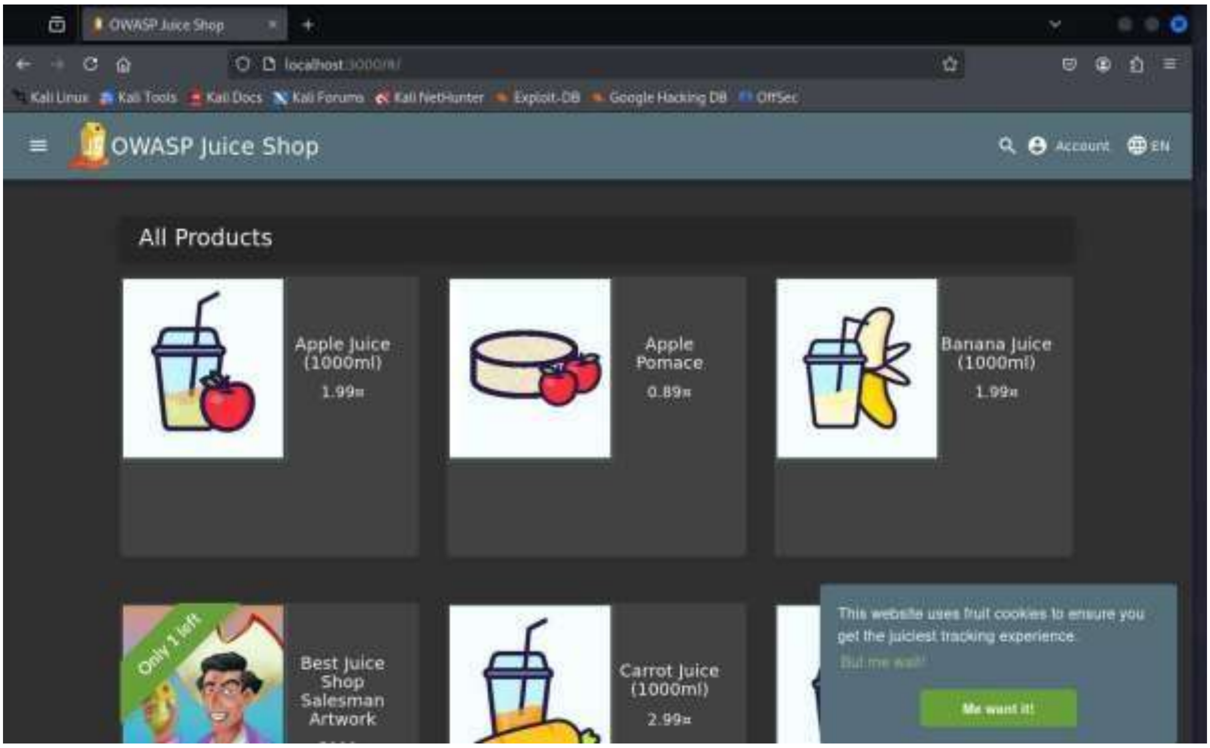
### Step 3: Deploying Juice Shop via Docker



```
┌──(root㉿prajwalsb)-[/home/prajwalsb/juice-shop]
└─# docker run -d -p 3000:3000 bkimminich/juice-shop

Unable to find image 'bkimminich/juice-shop:latest' locally
latest: Pulling from bkimminich/juice-shop
3d78e577de35: Pull complete
bfb59b82a9b6: Pull complete
4eff9a62d888: Pull complete
a62778643d56: Pull complete
7c12895b777b: Pull complete
3214acf345c0: Pull complete
5664b15f108b: Pull complete
0bab15eea81d: Pull complete
4aa0ea1413d3: Pull complete
da7816fa955e: Pull complete
9aee425378d2: Pull complete
d00c3209d929: Pull complete
221438ca359c: Pull complete
ab0f6cad3051: Pull complete
6f971e93c4e2: Pull complete
c83c31ce41af: Pull complete
0cb5c07f8edd: Pull complete
3137de975d0a: Pull complete
b78a86c4c4b5: Pull complete
bc0e9837a40c: Downloading [==================================>                ]  68.03MB/97.55MB
916a4ab8bcd7: Download complete

bc0e9837a40c: Pull complete
916a4ab8bcd7: Pull complete
Digest: sha256:db5bacffb67d4baba08df1d7cf79aa57402eddfe526da37cdf0620a4131e82ca
Status: Downloaded newer image for bkimminich/juice-shop:latest
a1990c28b9534afd4e1d68accf3fb3cf44c4b3e38f4037557eda4d65780c0a72
```



```
Processing triggers for kali-menu (2025.2.3) ...

┌──(root㉿prajwalsb)-[/home/prajwalsb/juice-shop]
└─# sudo systemctl enable docker --now
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```



This website uses fruit cookies to ensure you get the juiciest tracking experience. But me wait!

**Heading:** Containerized
Deployment for Easy Access

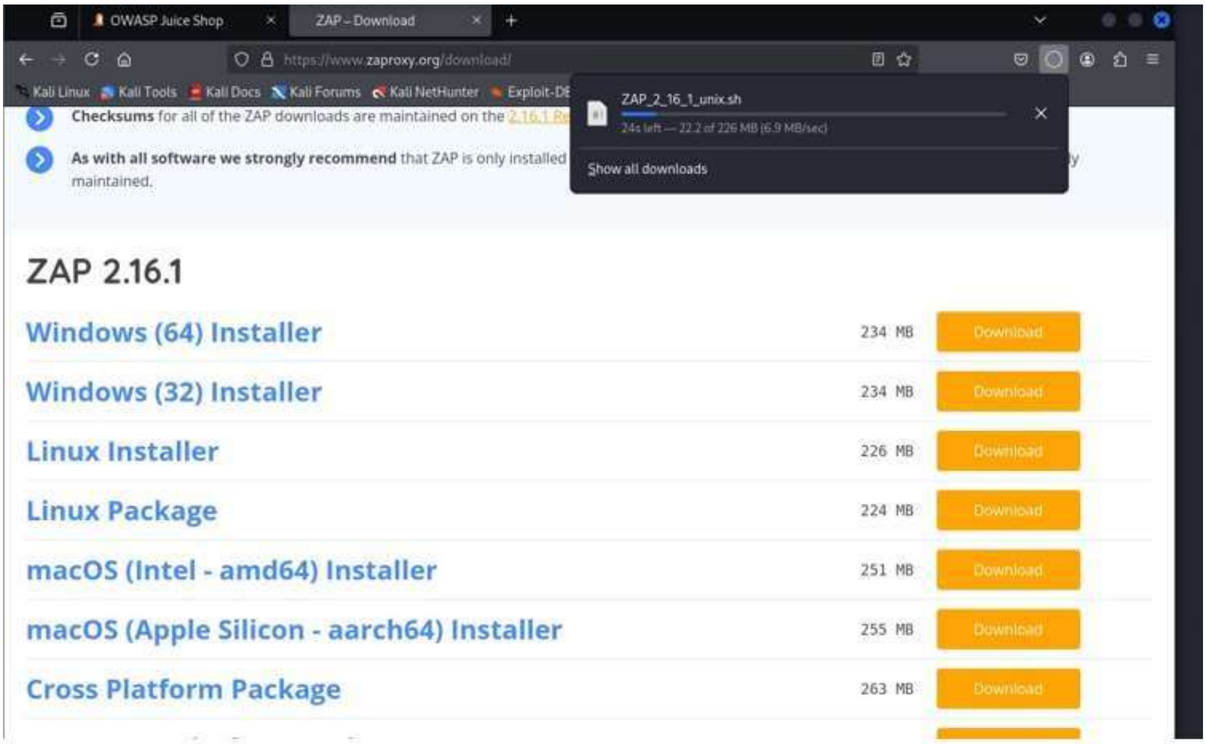
**Brief Explanation:**

Instead of running Juice Shop locally, Docker is used to deploy it as a container. The command docker run -d -p 3000:3000 bkimminich/juice-shop pulls the pre-built image and starts the application on port 3000. This method simplifies setup and ensures consistency.

**Lab Observation:**

- The Docker image is downloaded successfully, and the container starts without errors.
- Accessing http://localhost:3000 confirms the application is running, displaying a mock ecommerce interface.

**Step 4: Installing and Configuring ZAP Proxy**

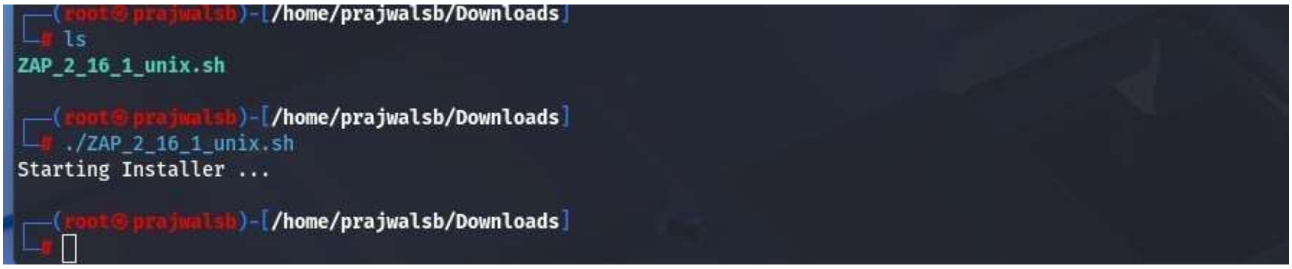


**Checksums** for all of the ZAP downloads are maintained on the 2.16.1 Re

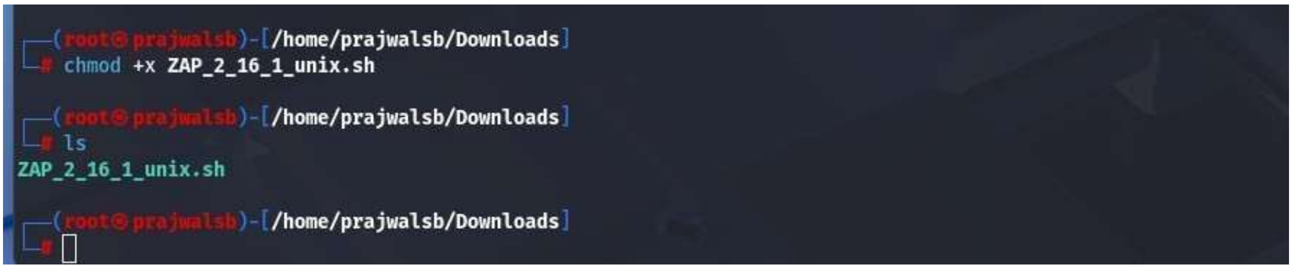
**As with all software we strongly recommend** that ZAP is only installed y maintained.

## ZAP 2.16.1

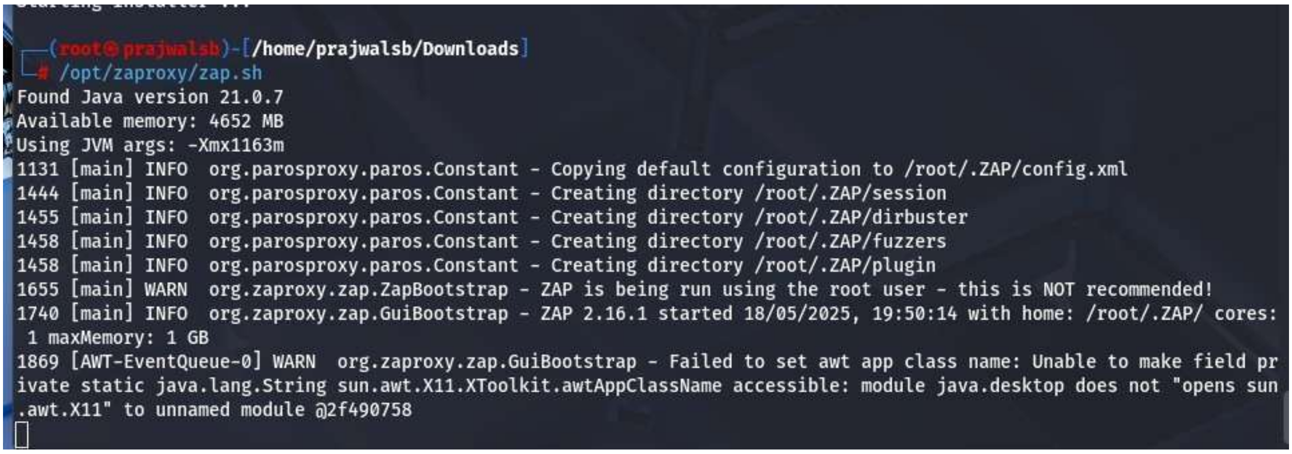
| | | |
|---|---|---|
| Windows (64) Installer | 234 MB | Download |
| Windows (32) Installer | 234 MB | Download |
| Linux Installer | 226 MB | Download |
| Linux Package | 224 MB | Download |
| macOS (Intel - amd64) Installer | 251 MB | Download |
| macOS (Apple Silicon - aarch64) Installer | 255 MB | Download |
| Cross Platform Package | 263 MB | Download |



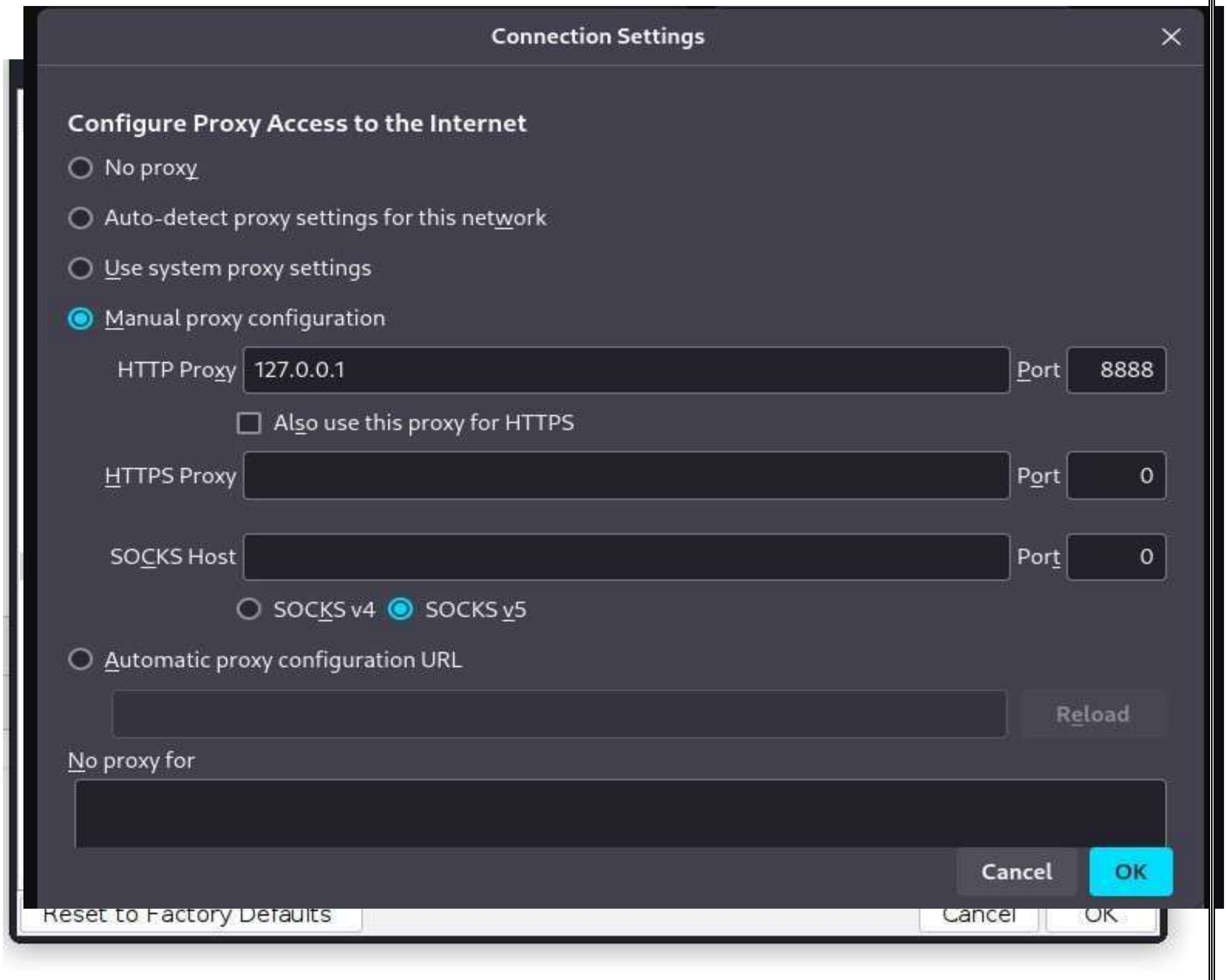
```
┌──(root㉿prajwalsb)-[/home/prajwalsb/Downloads]
└─# ls
ZAP_2_16_1_unix.sh

┌──(root㉿prajwalsb)-[/home/prajwalsb/Downloads]
└─# ./ZAP_2_16_1_unix.sh
Starting Installer ...

┌──(root㉿prajwalsb)-[/home/prajwalsb/Downloads]
└─# 
```



```
┌──(root㉿prajwalsb)-[/home/prajwalsb/Downloads]
└─# chmod +x ZAP_2_16_1_unix.sh

┌──(root㉿prajwalsb)-[/home/prajwalsb/Downloads]
└─# ls
ZAP_2_16_1_unix.sh

┌──(root㉿prajwalsb)-[/home/prajwalsb/Downloads]
└─# 
```



```
Starting Installer ...

┌──(root㉿prajwalsb)-[/home/prajwalsb/Downloads]
└─# /opt/zaproxy/zap.sh
Found Java version 21.0.7
Available memory: 4652 MB
Using JVM args: -Xmx1163m
1131 [main] INFO  org.parosproxy.paros.Constant - Copying default configuration to /root/.ZAP/config.xml
1444 [main] INFO  org.parosproxy.paros.Constant - Creating directory /root/.ZAP/session
1455 [main] INFO  org.parosproxy.paros.Constant - Creating directory /root/.ZAP/dirbuster
1458 [main] INFO  org.parosproxy.paros.Constant - Creating directory /root/.ZAP/fuzzers
1458 [main] INFO  org.parosproxy.paros.Constant - Creating directory /root/.ZAP/plugin
1655 [main] WARN  org.zaproxy.zap.ZapBootstrap - ZAP is being run using the root user - this is NOT recommended!
1740 [main] INFO  org.zaproxy.zap.GuiBootstrap - ZAP 2.16.1 started 18/05/2025, 19:50:14 with home: /root/.ZAP/ cores:
 1 maxMemory: 1 GB
1869 [AWT-EventQueue-0] WARN  org.zaproxy.zap.GuiBootstrap - Failed to set awt app class name: Unable to make field pr
ivate static java.lang.String sun.awt.X11.XToolkit.awtAppClassName accessible: module java.desktop does not "opens sun
.awt.X11" to unnamed module @2f490758
```



Connection Settings

Configure Proxy Access to the Internet

○ No proxy

○ Auto-detect proxy settings for this network

○ Use system proxy settings

● Manual proxy configuration

HTTP Proxy 127.0.0.1 Port 8888

□ Also use this proxy for HTTPS

HTTPS Proxy Port 0

SOCKS Host Port 0

○ SOCKS v4 ● SOCKS v5

○ Automatic proxy configuration URL

No proxy for

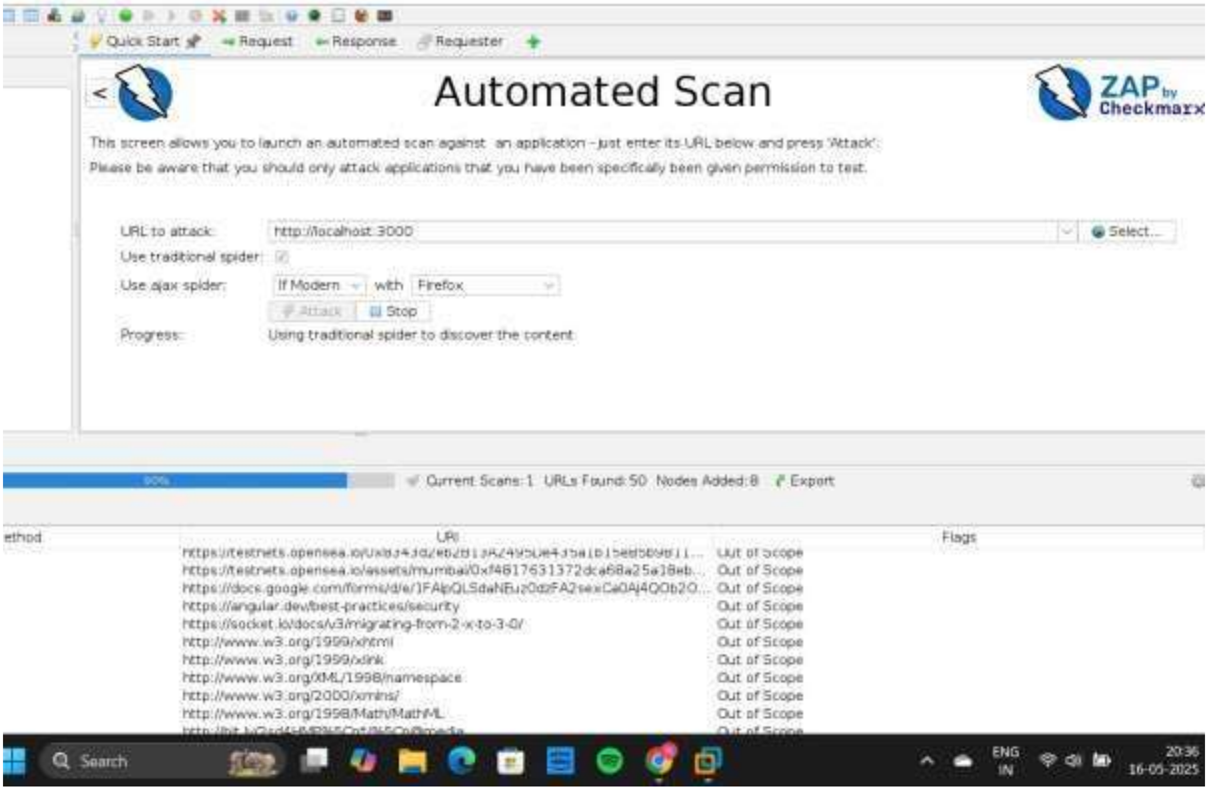
**Heading:** Preparing the Penetration Testing Tool **Brief Explanation:**

OWASP ZAP (Zed Attack Proxy) is downloaded and installed to perform security testing on Juice Shop. The installer is executed, and ZAP is configured to proxy traffic through 127.0.0.1:8888. This setup allows ZAP to intercept and analyze HTTP/HTTPS requests.

**Lab Observation:**

- ZAP 2.16.1 is installed on a Linux system.
- Proxy settings are configured manually to ensure traffic routing through ZAP.

## Step 5: Initiating an Automated Scan



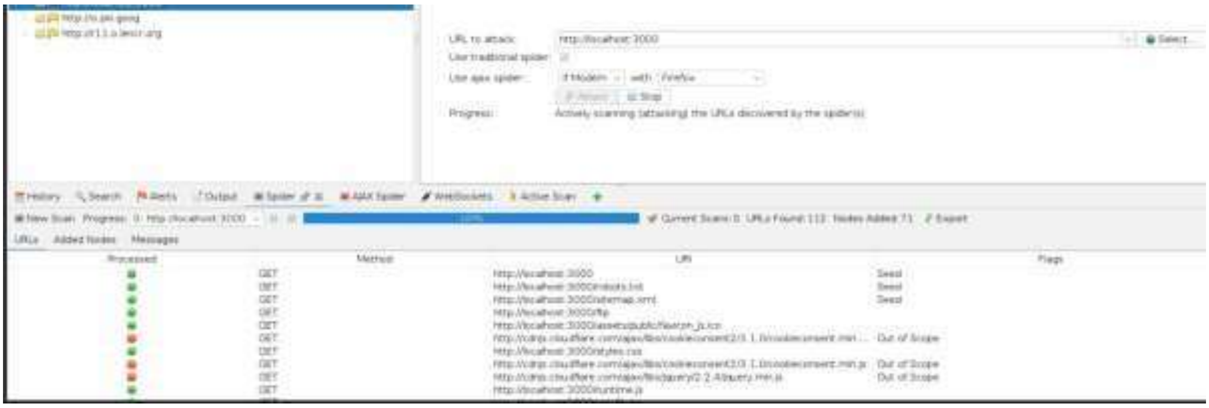
Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://localhost:3000
Use traditional spider: ☑
Use ajax spider: If Modern with Firefox
Attack | Stop
Progress: Using traditional spider to discover the content

60% Current Scans: 1 URLs Found: 50 Nodes Added: 8 Export

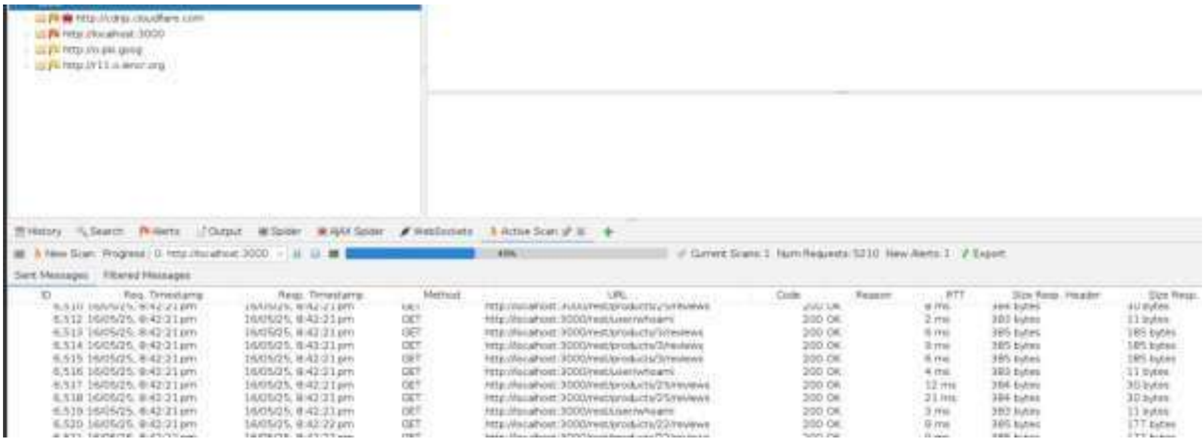
| ethod | URI | Flags |
|---|---|---|
| | https://testnets.opensea.io/assets/mumbai/0xf4817631372dca68a25a18eb… | Out of Scope |
| | https://docs.google.com/forms/d/e/1FAIpQLSdaNEuz0dzFA2sexCa0Aj4QOb2O… | Out of Scope |
| | https://angular.dev/best-practices/security | Out of Scope |
| | https://socket.io/docs/v3/migrating-from-2-x-to-3-0/ | Out of Scope |
| | http://www.w3.org/1999/xhtml | Out of Scope |
| | http://www.w3.org/1999/xlink | Out of Scope |
| | http://www.w3.org/XML/1998/namespace | Out of Scope |
| | http://www.w3.org/2000/xmlns/ | Out of Scope |
| | http://www.w3.org/1998/Math/MathML | Out of Scope |



URL to attack: http://localhost:3000
Use traditional spider: ☑
Use ajax spider: If Modern with Firefox
Attack | Stop
Progress: Actively scanning (attacking) the URLs discovered by the spider(s)

New Scan Progress: 0: http://localhost:3000 100% Current Scans: 0 URLs Found: 112 Nodes Added: 71 Export

| Processed | Method | URI | Flags |
|---|---|---|---|
| ● | GET | http://localhost:3000 | Seed |
| ● | GET | http://localhost:3000/robots.txt | Seed |
| ● | GET | http://localhost:3000/sitemap.xml | Seed |
| ● | GET | http://localhost:3000/ftp | |
| ● | GET | http://localhost:3000/assets/public/favicon_js.ico | |
| ● | GET | http://cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min… | Out of Scope |
| ● | GET | http://localhost:3000/styles.css | |
| ● | GET | http://cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js | Out of Scope |
| ● | GET | http://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js | Out of Scope |
| ● | GET | http://localhost:3000/runtime.js | |



New Scan: Progress: 0: http://localhost:3000 49% Current Scans: 1 Num Requests: 5210 New Alerts: 1 Export

| ID | Req. Timestamp | Resp. Timestamp | Method | URL | Code | Reason | RTT | Size Resp. Header | Size Resp. |
|---|---|---|---|---|---|---|---|---|---|
| 6,512 | 16/05/25, 8:42:21 pm | 16/05/25, 8:42:21 pm | GET | http://localhost:3000/rest/user/whoami | 200 | OK | 2 ms | 383 bytes | 11 bytes |
| 6,513 | 16/05/25, 8:42:21 pm | 16/05/25, 8:42:21 pm | GET | http://localhost:3000/rest/products/3/reviews | 200 | OK | 6 ms | 385 bytes | 185 bytes |
| 6,514 | 16/05/25, 8:42:21 pm | 16/05/25, 8:43:21 pm | GET | http://localhost:3000/rest/products/3/reviews | 200 | OK | 9 ms | 385 bytes | 185 bytes |
| 6,515 | 16/05/25, 8:42:21 pm | 16/05/25, 8:42:21 pm | GET | http://localhost:3000/rest/products/3/reviews | 200 | OK | 8 ms | 385 bytes | 185 bytes |
| 6,516 | 16/05/25, 8:42:21 pm | 16/05/25, 8:42:21 pm | GET | http://localhost:3000/rest/user/whoami | 200 | OK | 4 ms | 383 bytes | 11 bytes |
| 6,517 | 16/05/25, 8:42:21 pm | 16/05/25, 8:42:21 pm | GET | http://localhost:3000/rest/products/25/reviews | 200 | OK | 12 ms | 384 bytes | 30 bytes |
| 6,518 | 16/05/25, 8:42:21 pm | 16/05/25, 8:42:21 pm | GET | http://localhost:3000/rest/products/25/reviews | 200 | OK | 21 ms | 384 bytes | 30 bytes |
| 6,519 | 16/05/25, 8:42:21 pm | 16/05/25, 8:42:21 pm | GET | http://localhost:3000/rest/user/whoami | 200 | OK | 3 ms | 383 bytes | 11 bytes |
| 6,520 | 16/05/25, 8:42:21 pm | 16/05/25, 8:42:22 pm | GET | http://localhost:3000/rest/products/22/reviews | 200 | OK | 9 ms | 385 bytes | 177 bytes |

**Heading:**

Launching a Vulnerability Scan

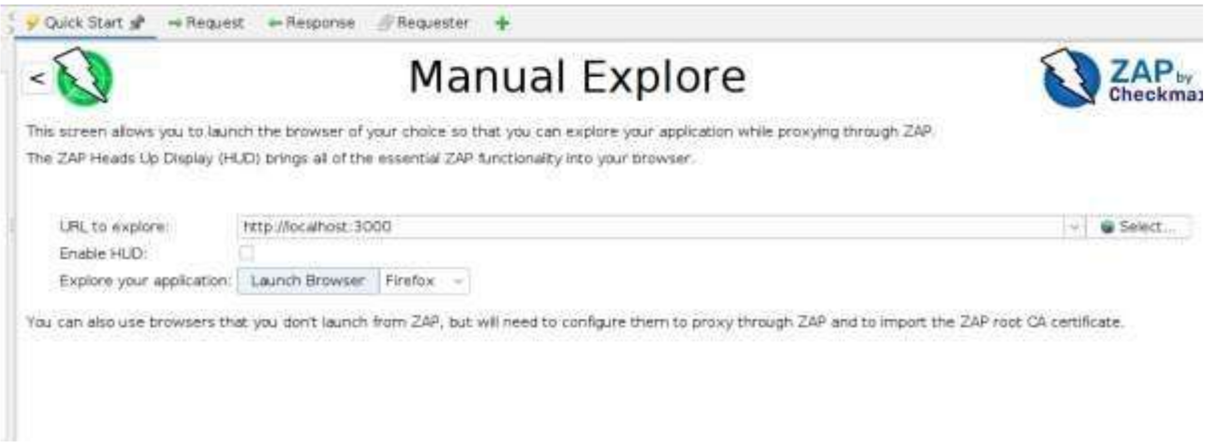
**Brief Explanation:**

ZAP's "Automated Scan" feature is used to test Juice Shop. The target URL (http://localhost:3000) is entered, and the scan begins, spidering the application to identify vulnerabilities. The progress is monitored in real-time.

**Lab Observation:**

- The scan discovers multiple URLs and nodes, though some are flagged as "Out of Scope" (e.g., external links).
- Initial results show low-severity alerts, but further analysis may reveal more critical issues.

## Step 6: Manual Exploration Setup



This screen allows you to launch the browser of your choice so that you can explore your application while proxying through ZAP.

The ZAP Heads Up Display (HUD) brings all of the essential ZAP functionality into your browser.

URL to explore: http://localhost:3000

Enable HUD: □

Explore your application: Launch Browser Firefox

You can also use browsers that you don't launch from ZAP, but will need to configure them to proxy through ZAP and to import the ZAP root CA certificate.

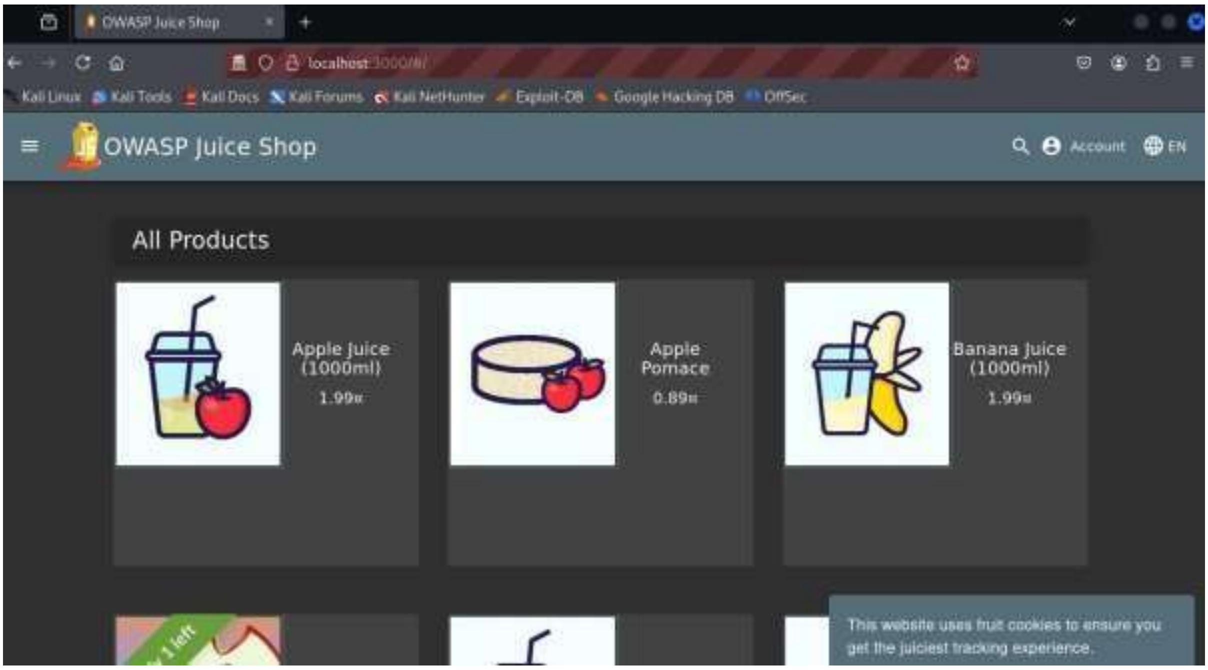
**Observation**:

- OWASP ZAP is configured to intercept traffic from http://localhost:3000.
- The HUD (Heads Up Display) is enabled for real-time feedback.
- Firefox is used as the browser configured with ZAP as a proxy.

**Purpose**:

- Initial setup to enable interception and monitoring of all interactions with the vulnerable Juice Shop application.

## Step 7: Product Listing Page



This website uses fruit cookies to ensure you get the juiciest tracking experience.

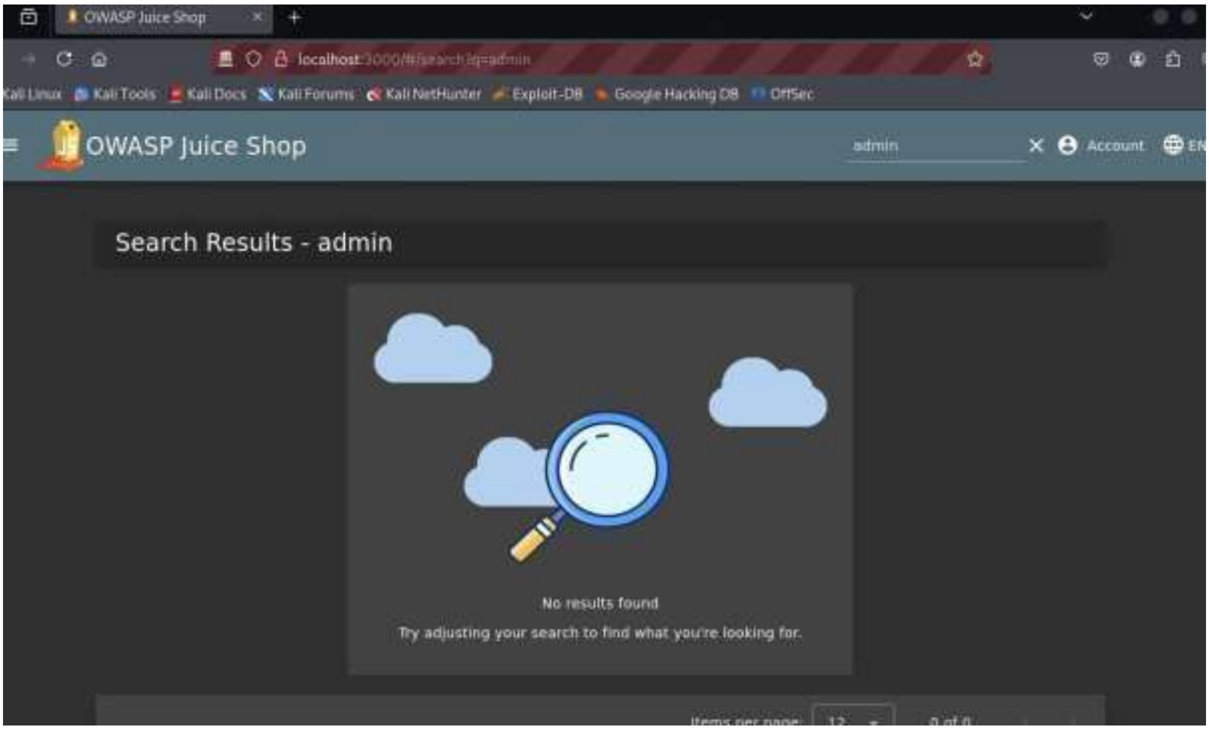
**Observation**:

- Displays product cards like Apple Juice, Banana Juice, and more.
- Includes a cookie tracking consent message.
- Shows the application’s default UI without any attacks.

**Purpose**:

- To establish a baseline of normal application behavior before testing vulnerabilities.

## Step 8: Search Functionality Test



Search Results - admin

No results found

Try adjusting your search to find what you're looking for.

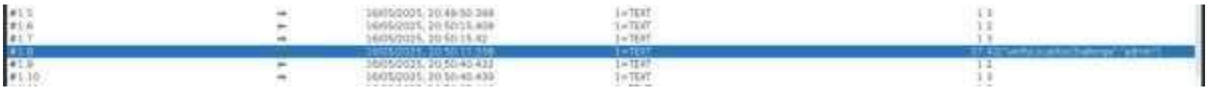
**Observation**:

- Searching for the keyword "admin" returns no visible results.
- Suggests possible hidden functionality or protection of admin features.

**Purpose**:

- Basic functionality test, possibly leading to enumeration attempts or privilege escalation testing.

## Step 9: WebSocket Traffic Analysis



```
#1.5    →    16/05/2025, 20:49:50.388    1=TEXT    1 3
#1.6    ←    16/05/2025, 20:50:15.408    1=TEXT    1 2
#1.7    →    16/05/2025, 20:50:15.42     1=TEXT    1 3
#1.8         16/05/2025, 20:50:11.558    1=TEXT    [illegible]["[illegible]","admin"]
#1.9    ←    16/05/2025, 20:50:40.432    1=TEXT    1 2
#1.10   →    16/05/2025, 20:50:40.430    1=TEXT    1 3
```

**Observation**:

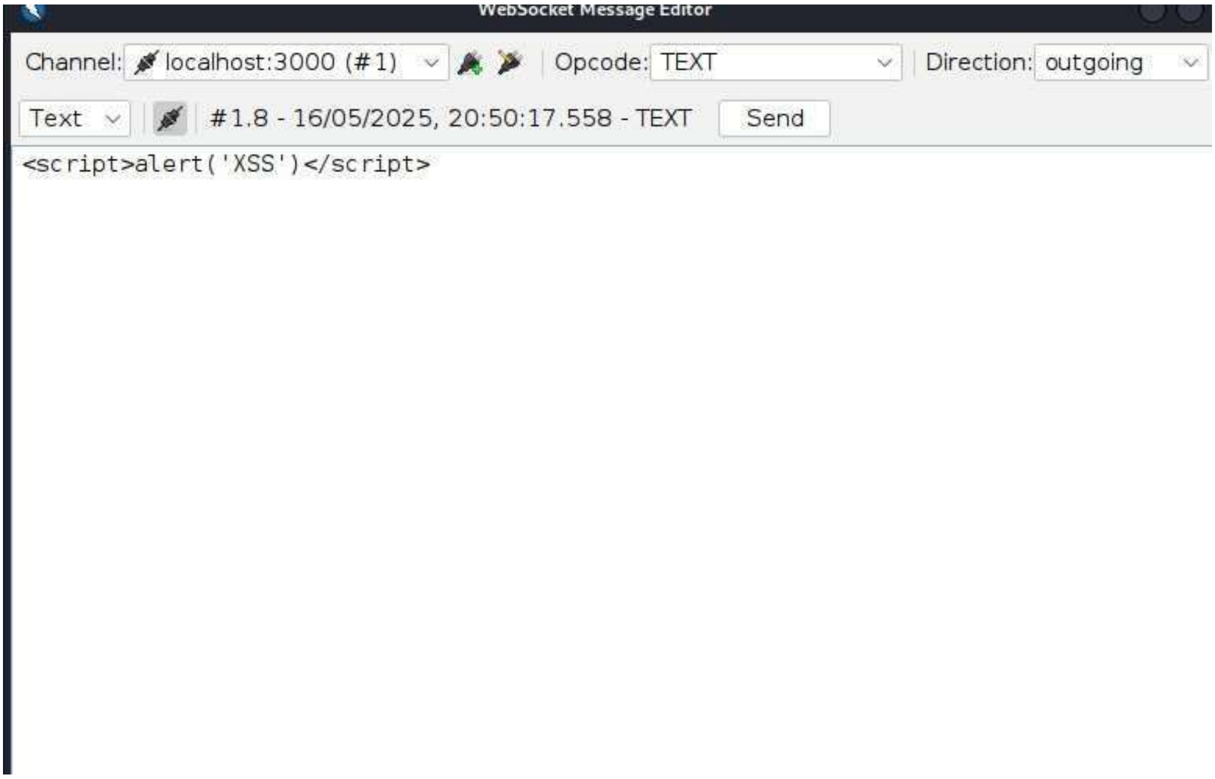
- ZAP logs WebSocket messages with TEXT opcodes and timestamps.
- One message references a path like /images/admin.

**Purpose**:

- Inspect real-time communication for hidden endpoints or sensitive data being exchanged via WebSockets.

### Step 10: XSS Payload Delivery

**Observation**:


WebSocket Message Editor
Channel: localhost:3000 (#1)
Opcode: TEXT
Direction: outgoing
Text
#1.8 - 16/05/2025, 20:50:17.558 - TEXT
Send

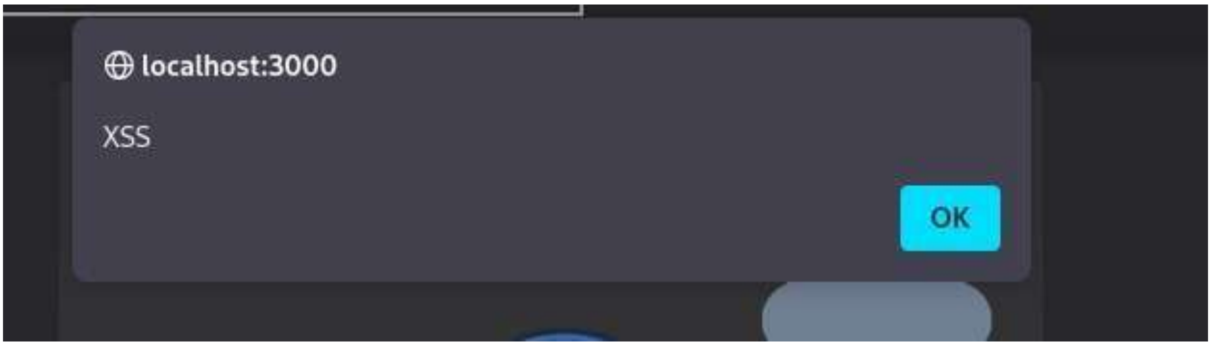
```
<script>alert('XSS')</script>
```

- JavaScript payload <script>alert('XSS')</script> is sent via WebSocket message.
- The message uses a TEXT opcode, indicating user-supplied input is being transmitted.

**Purpose**:

- Attempt to execute a stored or reflected XSS attack by injecting scripts into vulnerable input fields.

## Step 11: Successful XSS Execution



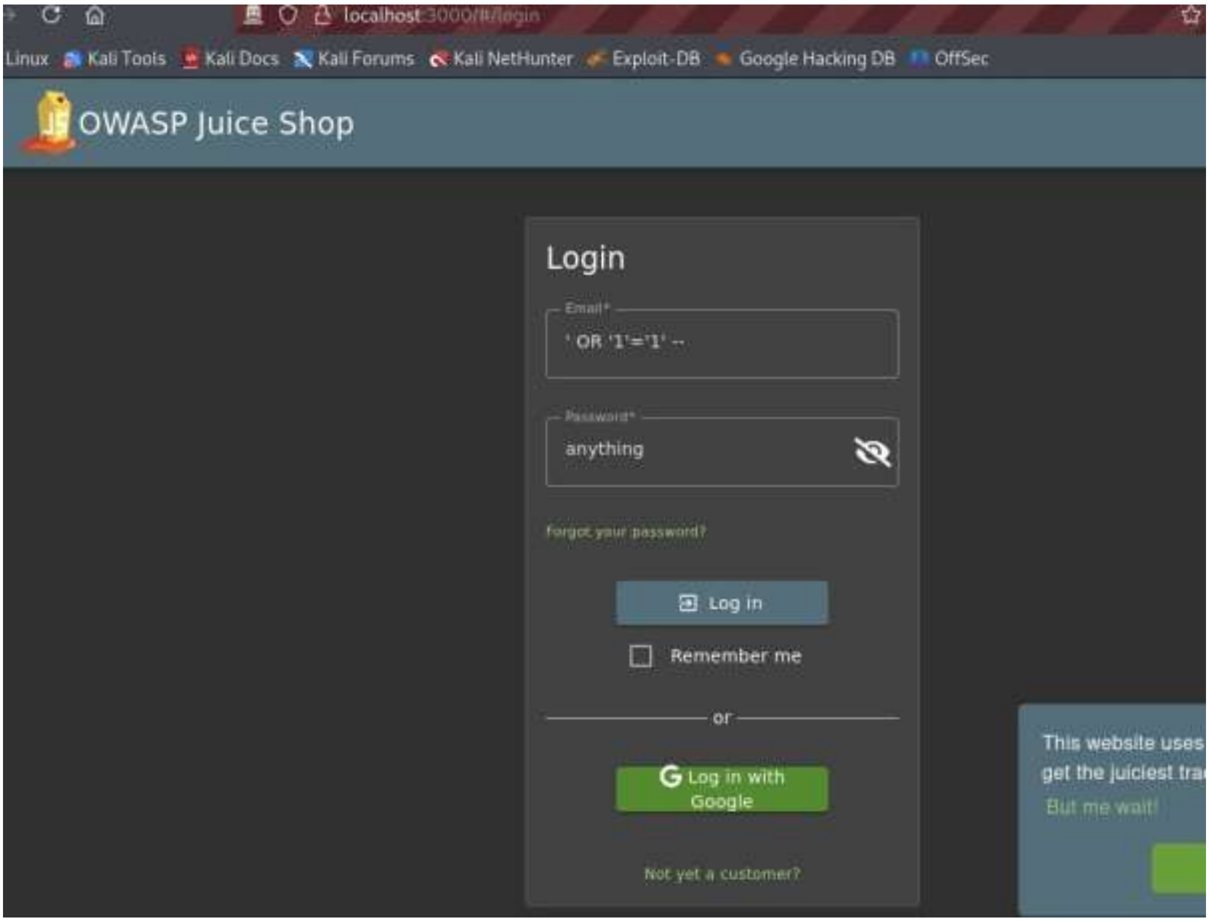
**Observation**:

- A popup alert box displays “XSS”, confirming the JavaScript payload was executed.

**Purpose**:

- Verify the XSS vulnerability and prove that script injection is possible, completing the related security challenge.

## Step 12: SQL Injection Attack



Email*: ' OR '1'='1' --

Password*: anything

□ Remember me

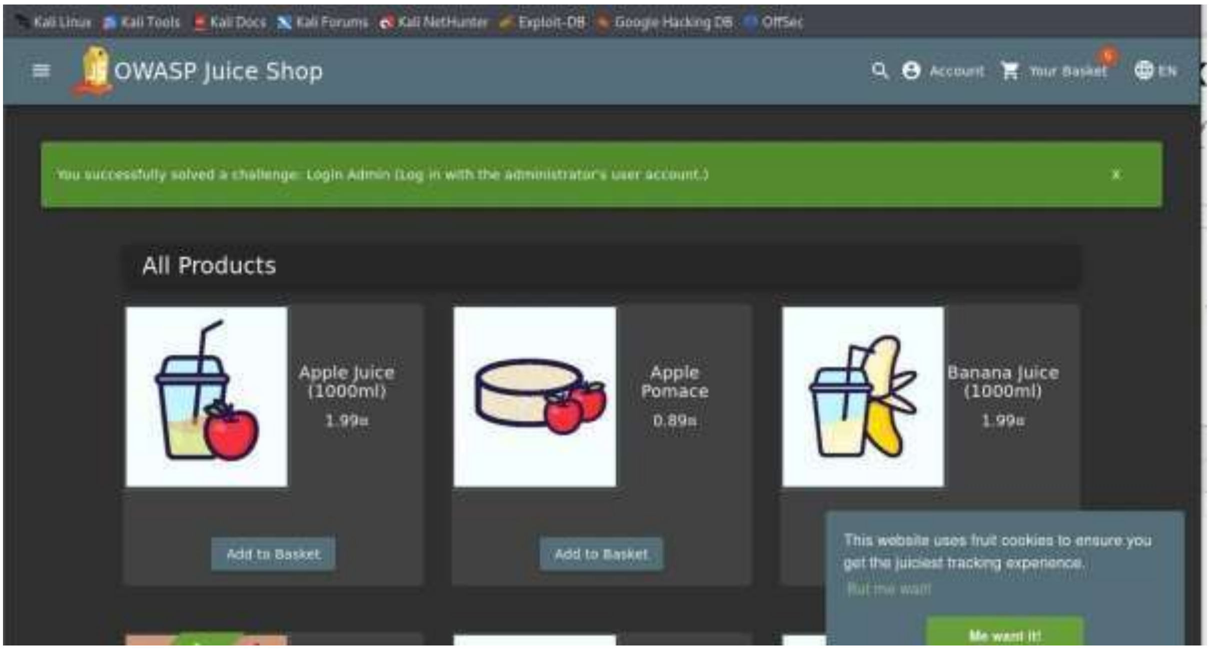
**Observation**:

- SQL Injection payload ' OR '1'='1' -- is injected into the login form’s email field.
- The password field is filled with irrelevant data to bypass checks.

**Purpose**:

- Exploit authentication logic flaw to bypass login restrictions and gain unauthorized access.

## Step 13: Successful Admin Login



You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

This website uses fruit cookies to ensure you get the juiciest tracking experience.

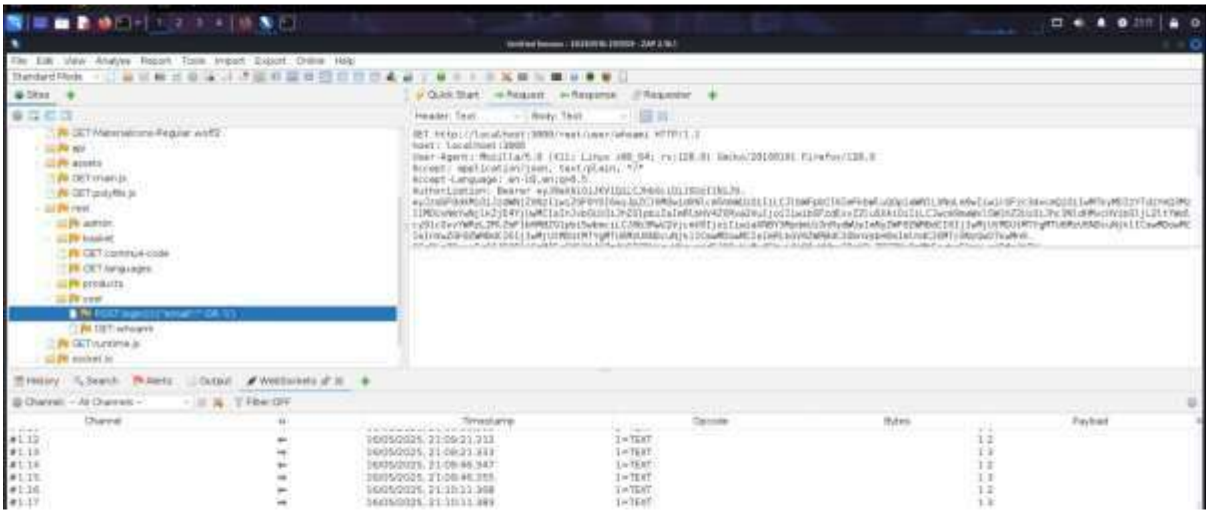
**Observation**:

- Application displays a message indicating that the “Login Admin” challenge was completed.
- Product listing page is visible post-login.

**Purpose**:

- Confirm that the SQL injection was successful and admin-level access was achieved.

## Step 14: Initial Login Request



```
GET http://localhost:3000/rest/user/whoami HTTP/1.1
host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.
[illegible]
```

**Request**:

POST /rest/user/login HTTP/1.1
Host: localhost:3000

Content-Type: application/json

**Payload**:

["email":"tgt",”]  // malformed JSON

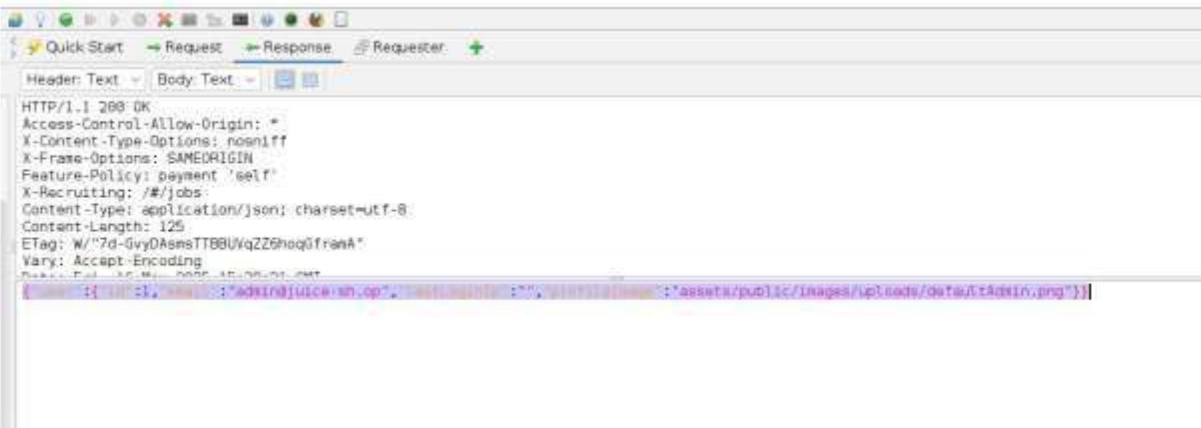
**Observation**:

- This malformed request is likely a test input to observe how the login endpoint handles broken JSON syntax.
- The application may log an error or return a 400 Bad Request, depending on its backend validation logic.

**Purpose**:

- To test the robustness of the backend login handler against invalid inputs, and potentially trigger verbose errors or identify parsing vulnerabilities.

## Step 15: Successful Admin Login Response



```
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
Content-Type: application/json; charset=utf-8
Content-Length: 125
ETag: W/"7d-GvyDAsmsTTBBUVqZZ6hoqGframA"
Vary: Accept-Encoding
{"user":{"id":1,"email":"admin@juice-sh.op","lastLoginIp":"","profileImage":"assets/public/images/uploads/defaultAdmin.png"}}
```

**Response**:

HTTP/1.1 200 OK

{"user":{"id":1,"email":"admin@juice-sh.op","profileImage":"defaultAdmin.png"}}

**Observation**:

- This response confirms a successful login for the admin user.
- Response headers include security-related configurations:

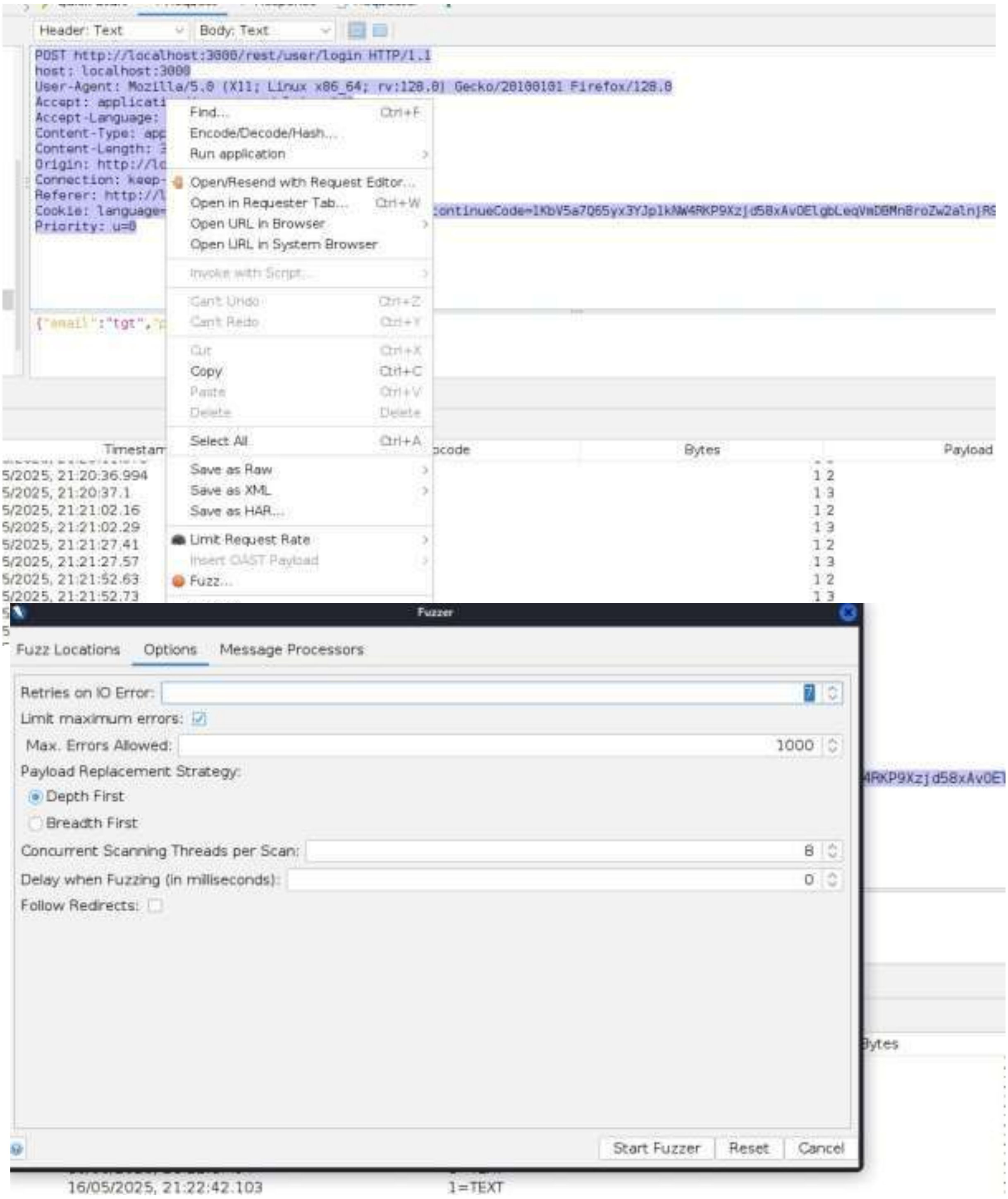
o X-Frame-Options: SAMEORIGIN o X-Content-Type-Options:

nosniff o Content- Security-Policy (possibly present)

**Purpose**:

- This step confirms that admin credentials were accepted and a session was successfully established.

## Step 16: Fuzzing Setup



```
POST http://localhost:3000/rest/user/login HTTP/1.1
host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: applicati
Accept-Language:
Content-Type: app
Content-Length: 3
Origin: http://lo
Connection: keep-
Referer: http://l
Cookie: language=                                continueCode=IKbV5a7Q65yx3YJplkNW4RKP9Xzjd58xAvOElgbLeqVmD8Mn8roZw2alnjRG
Priority: u=0
```

```
{"email":"tgt","p
```

```
5/2025, 21:20:36.994    1 2
5/2025, 21:20:37.1      1 3
5/2025, 21:21:02.16     1 2
5/2025, 21:21:02.29     1 3
5/2025, 21:21:27.41     1 2
5/2025, 21:21:27.57     1 3
5/2025, 21:21:52.63     1 2
5/2025, 21:21:52.73     1 3
```

Retries on IO Error: 7

Limit maximum errors: ☑

Max. Errors Allowed: 1000

Payload Replacement Strategy:

● Depth First

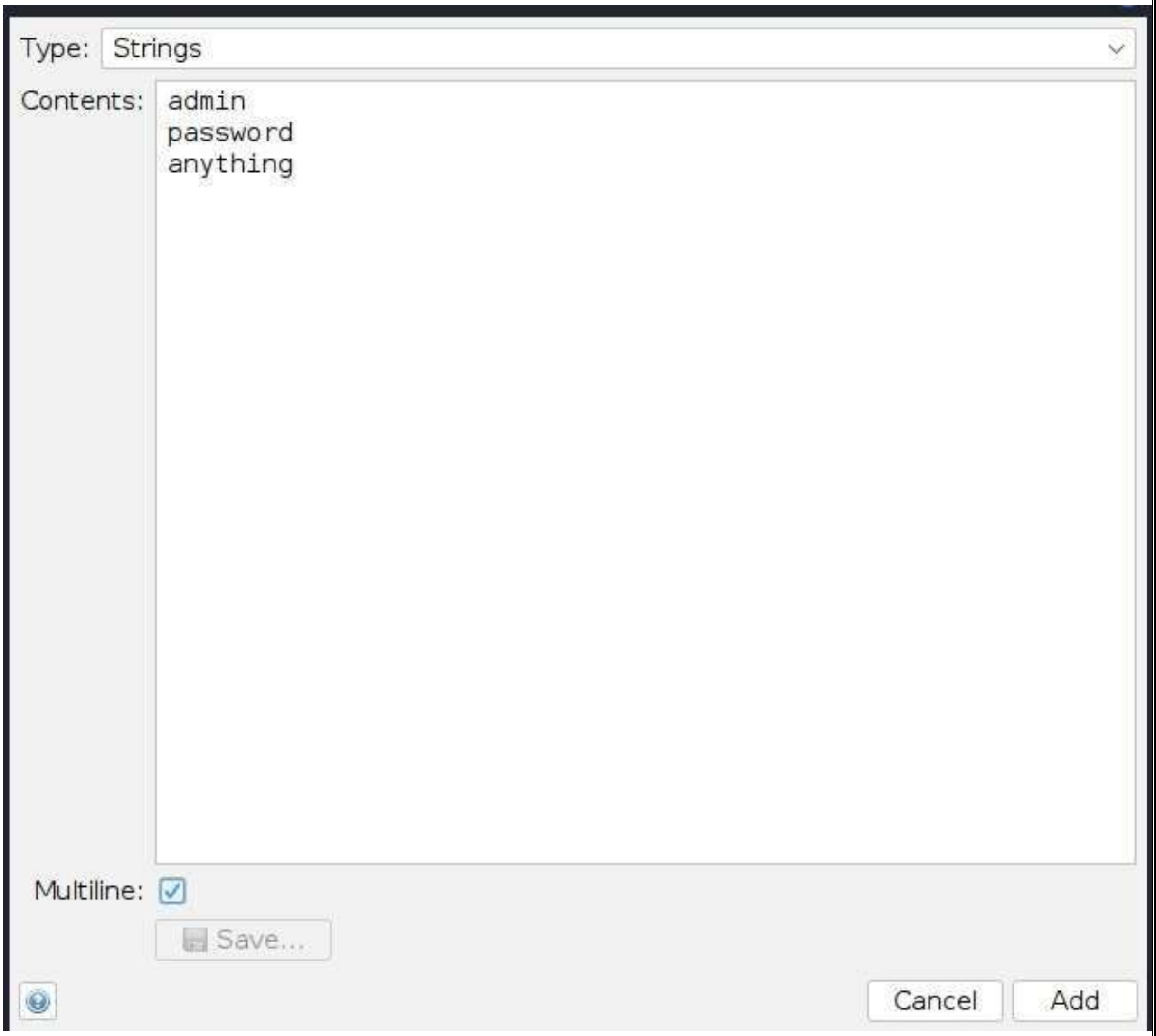
○ Breadth First

Concurrent Scanning Threads per Scan: 8

Delay when Fuzzing (in milliseconds): 0

Follow Redirects: □

```
4RKP9Xzjd58xAvOE
16/05/2025, 21:22:42.103          1=TEXT
```



Type: Strings

Contents:

```
admin
password
anything
```

Multiline: ☑

**Configuration**:

- Wordlist: admin, password, anything
- Threads: 8
- Delay: 0ms
- Strategy: **Depth First**
- Target: POST /rest/user/login

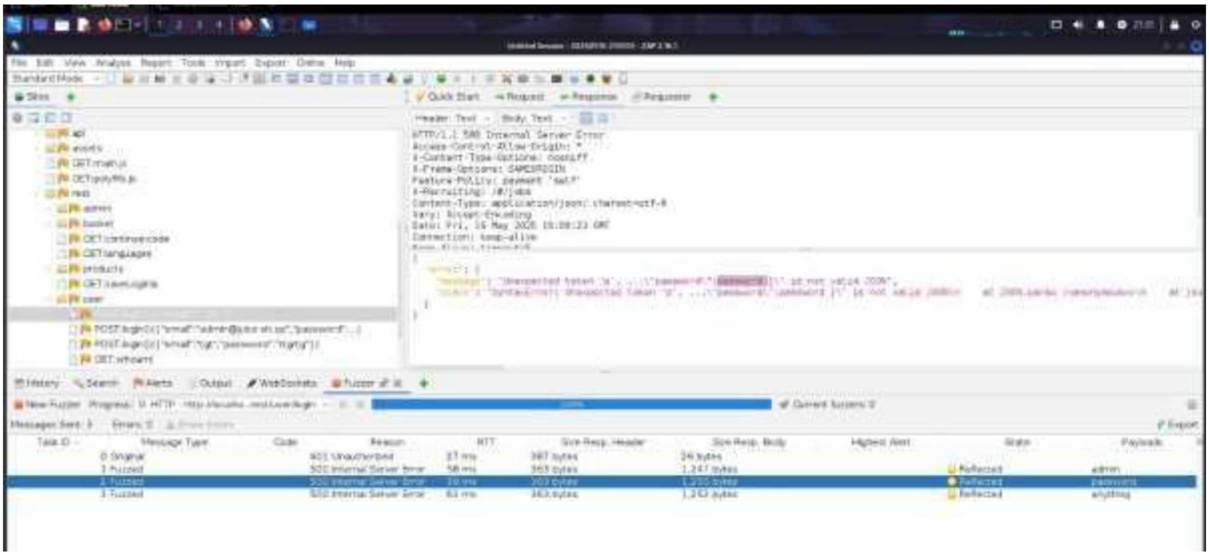
**Observation**:

- ZAP or another fuzzing tool is configured to test multiple login payloads using a basic wordlist.
- Fast threading and depth-first logic prioritize exploring deep inputs quickly.

**Purpose**:

- To brute-force or test login credentials and analyze how the server responds to different user/password combinations.

### Step 17: Fuzzing Execution and Output



```
HTTP/1.1 500 Internal Server Error
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Fri, 16 May 2025 15:08:23 GMT
Connection: keep-alive
```

| Task ID | Message Type | Code | Reason | RTT | Size Resp. Header | Size Resp. Body | Highest Alert | State | Payloads |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Original | 401 | Unauthorized | 27 ms | 387 bytes | 26 bytes | | | |
| 1 | Fuzzed | 500 | Internal Server Error | 58 ms | 363 bytes | 1,247 bytes | | Reflected | admin |
| 2 | Fuzzed | 500 | Internal Server Error | 39 ms | 363 bytes | 1,255 bytes | | Reflected | password |
| 3 | Fuzzed | 500 | Internal Server Error | 63 ms | 363 bytes | 1,253 bytes | | Reflected | anything |

**Results**:

- Base Request: 401 Unauthorized (expected for invalid credentials).
- Fuzzed Results:
  - Some requests return 500 Internal Server Error, possibly due to malformed inputs or unhandled exceptions.
  - No response with 200 OK, indicating that no login succeeded during fuzzing.

**Purpose**:

- To check if any known or common credentials can bypass authentication.
- Errors (500) might hint at poor input sanitization or exploitable bugs.

### Vulnerability Findings Table

| Vulnerability | Description | Affected Endpoint | Severity | Proof of Exploit | Recommendation |
|---|---|---|---|---|---|
| Cross-Site Scripting (XSS) | Reflected XSS in search input field | /search?q=<payload> | High | Images in Steps 10 and 11 | Implement strict input validation and output encoding |
| SQL Injection | Login form is susceptible to SQL injection | /login | Critical | Images in Steps 12,13,14 and 15 | Use parameterized queries to prevent SQL injection |
| Broken Authentication | Weak password policy allows brute-force attacks | /login | High | Images in Steps 16 and 17 | Enforce strong password policies and implement rate limiting |

## Impact Analysis and Mitigation

**1. Cross-Site Scripting (XSS)**

**Impact:**

- Allows attackers to inject malicious JavaScript into input fields (e.g., comment box, search bar).
- Enables session hijacking, phishing, redirection to malicious websites, and UI defacement.
- Compromises confidentiality and integrity of user sessions.

**Mitigation:**

- Perform strict input validation and sanitization on all user-supplied data.
- Use context-aware output encoding (e.g., HTML, JavaScript, URL).
- Implement browser security headers like **Content Security Policy (CSP)** to limit script execution.

---

**2. SQL Injection (SQLi)**

**Impact:**

- Lets attackers manipulate SQL queries to bypass authentication or retrieve unauthorized data.
- Can result in full compromise of the backend database.
- High risk when affecting login pages, search inputs, or forms interacting with the DB.

**Mitigation:**

- Use **parameterized queries** (prepared statements).
- Apply rigorous input validation and sanitization mechanisms.
- Ensure the database user has the **least-privilege** required for the application's function.

---

### 3. Broken Authentication

**Impact:**

- Weak password policies and lack of brute-force protection enable unauthorized login attempts.
- Can lead to full account takeover (including admin access).
- Increases risk of sensitive data leakage and system compromise.

**Mitigation:**

- Enforce strong password rules (minimum length, complexity, and expiration).
- Implement **CAPTCHA**, **account lockout**, and **rate limiting**.
- Enable **Multi-Factor Authentication (MFA)**.
- Monitor login attempts and generate alerts for suspicious behavior.

---

### 4. Insecure Direct Object References (IDOR)

**Impact:**

- Users may access other users' data or perform unauthorized actions by tampering with URL parameters.
- Results in exposure of private or business-sensitive information.

**Mitigation:**

- Always validate user access permissions on the **server side**.
- Avoid exposing sequential or predictable object identifiers.

- Use indirect object references like **UUIDs** or access tokens with proper validation.

## Result:

OWASP ZAP detected multiple vulnerabilities including XSS, SQLi, and insecure direct object references. Findings were documented with reco