

Automated Web App Security Testing

Author: Dasari Harsha Vardhan

GitHub: [LINK](#)

1. Introduction

2. Tools Used

3. Implementation Steps

4. Final Report

5. Overview

Automated Web App Security Testing

Introduction :

This project demonstrates automated security testing using Kali Linux, OWASP ZAP, and Docker. The goal was to detect OWASP Top 10 vulnerabilities (e.g., SQLi, XSS) in a vulnerable web app (OWASP Juice Shop) and generate actionable reports.



Tools Used :

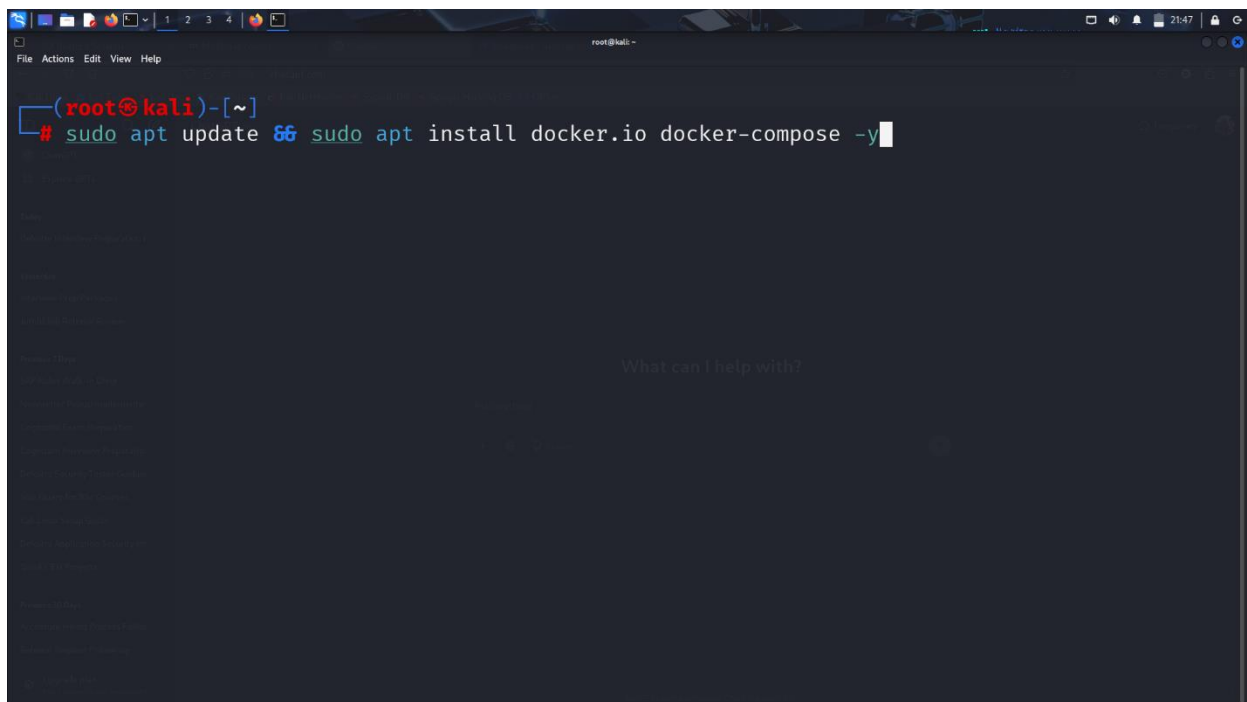
- **Kali Linux (Security-focused OS)**
- **OWASP ZAP (Zed Attack Proxy for security testing)**
- **Python 3.x (For automation scripting)**
- **Docker & Docker Compose (For hosting test applications)**
- **SQLMap & Nmap (Additional security testing tools)**
- **GitHub (Version control and documentation)**

Automated Web App Security Testing

Methodology & Implementation :

Step 1: Setting Up the Vulnerable App

- Installed Docker and cloned the OWASP Juice Shop repository:

A terminal window on a Kali Linux system. The prompt is (root@kali)~. The user has entered the command 'sudo apt update' followed by 'sudo apt install docker.io docker-compose -y'. The terminal shows the output of the update command, including a list of packages to be upgraded and their versions. The installation command is partially visible at the bottom of the screen.

```
(root@kali)~[~]
# sudo apt update 66 sudo apt install docker.io docker-compose -y
```

Automated Web App Security Testing

- Cloned **OWASP Juice Shop** (a vulnerable web application):

```
root@kali:~# docker-compose up --build
Setting up python3-compose (1.29.2-6.4) ...
Setting up docker-compose (1.29.2-6.4) ...
Processing triggers for kali-menu (2025.1.1) ...
Processing triggers for man-db (2.13.0-1) ...
Scanning processes ...
Scanning linux images ...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.

(root@kali)-[~]
# git clone https://github.com/juice-shop/juice-shop.git
cd juice-shop
Cloning into 'juice-shop' ...
remote: Enumerating objects: 135601, done.
remote: Counting objects: 100% (12/12), done.
Receiving objects: 24% (32728/135601), 79.39 MiB | 872.00 KiB/s
```

- Started the application in Docker:

```
root@kali:~/juice-shop# ls
app.json  docker-compose.test.yml  models  SOLUTIONS.md
app.ts    Dockerfile               monitoring  swagger.yml
CODE_OF_CONDUCT.md  encryptionkeys  package.json  test
config    frontend                README.md  threat-model.json
config.schema.yml  ftp             REFERENCES.md  tsconfig.json
CONTRIBUTING.md  Gruntfile.js    routes      uploads
crowdin.yaml      HALL_OF_FAME.md  rsn         vagrant
ctf.key           i18n             screenshots  views
cypress.config.ts lib              SECURITY.md  server.ts
data              LICENSE

(root@kali)-[~/juice-shop]
# ls | grep -i compose
docker-compose.test.yml

(root@kali)-[~/juice-shop]
# nano docker-compose.yml

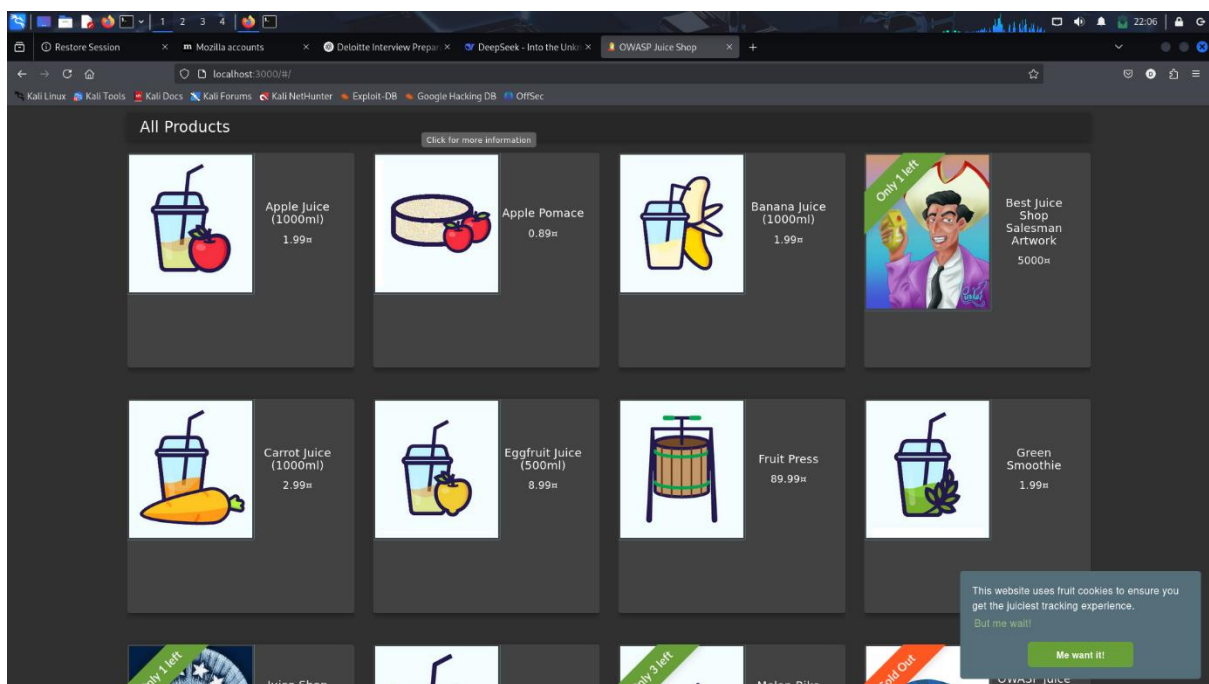
(root@kali)-[~/juice-shop]
# docker-compose up -d
Creating network "juice-shop_default" with the default driver
Pulling juice-shop (bkimminich/juice-shop:latest)...
latest: Pulling from bkimminich/juice-shop
1c56d6035a42: Pull complete
```

Automated Web App Security Testing

Running a Manual Scan in OWASP ZAP

Step 1: Environment Setup

- Deployed OWASP Juice Shop using Docker:



Automated Web App Security Testing

Step 2: Automated Scanning

- Scripted ZAP API integration with Python:

```
(root@kali)-[~/juice-shop]
# sudo apt install zapproxy -y
The following packages were automatically installed and are no longer required:
crackmapexec          libglvnd-dev          libsuperlu6
firebird3.0-common    libgtksourceview-3.0-1 libtag1v5
firebird3.0-common-doc libgtksourceview-3.0-common libtag1v5-vanilla
imagemagick-6.q16     libgtksourceviewmm-3.0-0v5 libtagc0
libbfbio1             libgumbo2             libunwind-19
libc++1-19            libhdf5-103-1t64      libwebRTC-audio-processing1
libc++abi1-19         libhdf5-hl-100t64     libx265-209
libcapstone4          libjxl0.9             openjdk-23-jre
libconfig++9v5        libldap-2.5-0         openjdk-23-jre-headless
libconfig9            libmagickcore-6.q16-7-extra python3-appdirs
libdirectfb-1.7-7t64  libmagickcore-6.q16-7t64 python3-ntlm-auth
libegl-dev            libmagickwand-6.q16-7t64 python3-setproctitle
libflac12t64         libmbedcrypto7t64     python3.12
libfmt9              libmsgpack-0-1        python3.12-dev
libgdal35            libnetcdf19t64        python3.12-minimal
libgeos3.13.0        libpaper1             python3.12-venv
libgl1-mesa-dev       libpoppler140         ruby-zeitwerk
libglapi-mesa         libpython3.12-dev     ruby3.1
libgles-dev          libqt5sensors5        ruby3.1-dev
libgles1             libqt5webkit5         ruby3.1-doc
```

-
-

```
(root@kali)-[~/juice-shop]
# sudo apt autoremove
libglvnd-core-dev      libqt5x11extras5
Use 'sudo apt autoremove' to remove them.

Summary:
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 127

(root@kali)-[~/juice-shop]
# zap.sh --version
Command 'zap.sh' not found, did you mean:
  command 'gap.sh' from deb wims
Try: apt install <deb name>

(root@kali)-[~/juice-shop]
# which zap.sh
zap.sh not found

(root@kali)-[~/juice-shop]
# export PATH=$PATH:/usr/share/zaproxy
(root@kali)-[~/juice-shop]
# find / -name "zap.sh" 2>/dev/null
/usr/share/zaproxy/zap.sh

(root@kali)-[~/juice-shop]
```

Automated Web App Security Testing

- **Installed required Python libraries:**

```
pip install python-owasp-zap-v2.4 requests
```

- **Created a Python script (`zap_scan.py`) to automate security scanning:**

```
from zapv2 import ZAPv2
```

```
import time
```

```
zap = ZAPv2(apikey='12345', proxies={'http': 'http://localhost:8080', 'https':  
'http://localhost:8080'})
```

```
target = 'http://localhost:3000'
```

```
print('Starting scan...')
```

```
scan_id = zap.ascan.scan(target)
```

```
while int(zap.ascan.status(scan_id)) < 100:
```

```
print(f'Scan progress: {zap.ascan.status(scan_id)}%')
```

```
time.sleep(10)
```

```
report_html = zap.core.htmlreport()
```

```
with open('zap_report.html', 'w') as f:
```

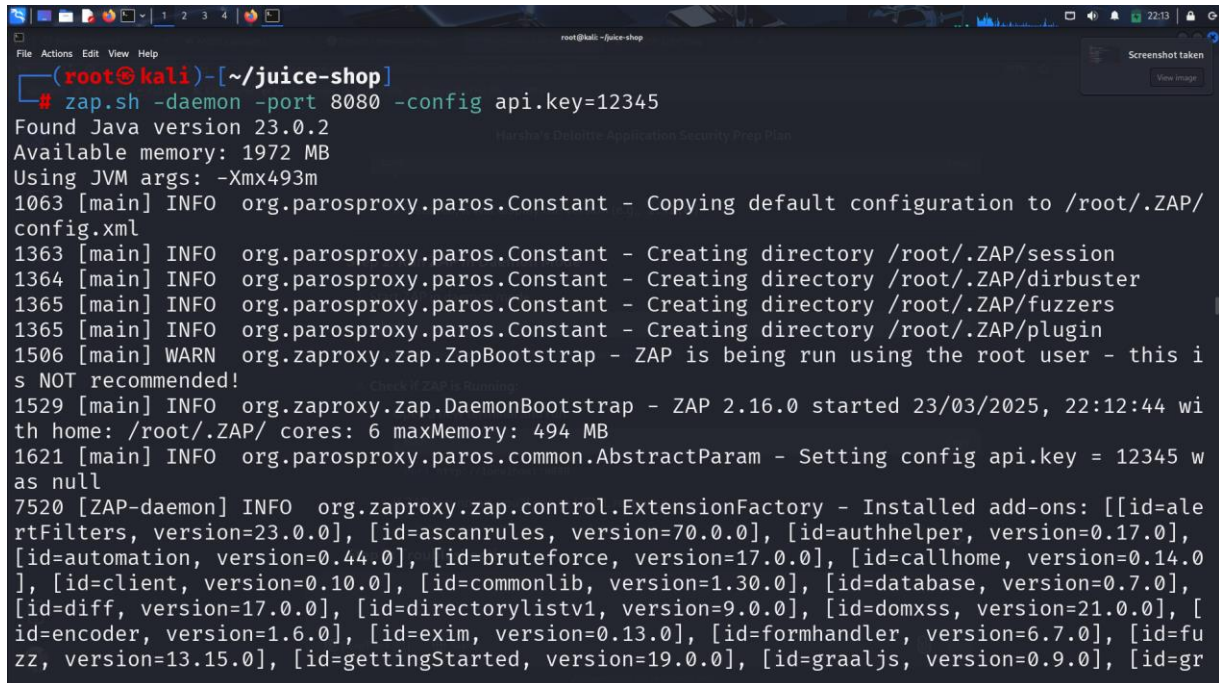
```
f.write(report_html)
```

```
print('Report saved as zap_report.html!')
```

- **Ran the script:**

```
python3 zap_scan.py
```


Automated Web App Security Testing



```
(root@kali)-[~/juice-shop]
# zap.sh -daemon -port 8080 -config api.key=12345
Found Java version 23.0.2
Available memory: 1972 MB
Using JVM args: -Xmx493m
1063 [main] INFO org.parosproxy.paros.Constant - Copying default configuration to /root/.ZAP/
config.xml
1363 [main] INFO org.parosproxy.paros.Constant - Creating directory /root/.ZAP/session
1364 [main] INFO org.parosproxy.paros.Constant - Creating directory /root/.ZAP/dirbuster
1365 [main] INFO org.parosproxy.paros.Constant - Creating directory /root/.ZAP/fuzzers
1365 [main] INFO org.parosproxy.paros.Constant - Creating directory /root/.ZAP/plugin
1506 [main] WARN org.zaproxy.zap.ZapBootstrap - ZAP is being run using the root user - this i
s NOT recommended!
1529 [main] INFO org.zaproxy.zap.DaemonBootstrap - ZAP 2.16.0 started 23/03/2025, 22:12:44 wi
th home: /root/.ZAP/ cores: 6 maxMemory: 494 MB
1621 [main] INFO org.parosproxy.paros.common.AbstractParam - Setting config api.key = 12345 w
as null
7520 [ZAP-daemon] INFO org.zaproxy.zap.control.ExtensionFactory - Installed add-ons: [[id=ale
rtFilters, version=23.0.0], [id=ascanrules, version=70.0.0], [id=authhelper, version=0.17.0],
[id=automation, version=0.44.0], [id=bruteforce, version=17.0.0], [id=callhome, version=0.14.0
], [id=client, version=0.10.0], [id=commonlib, version=1.30.0], [id=database, version=0.7.0],
[id=diff, version=17.0.0], [id=directorylistv1, version=9.0.0], [id=domxss, version=21.0.0],
[id=encoder, version=1.6.0], [id=exim, version=0.13.0], [id=formhandler, version=6.7.0], [id=fu
zz, version=13.15.0], [id=gettingStarted, version=19.0.0], [id=graaljs, version=0.9.0], [id=gr
```



Welcome to the Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically given permission to test.

Proxy Configuration

To use ZAP effectively it is recommended that you configure your browser to proxy via ZAP.

The easiest way to do this is to launch your browser from ZAP via the "Quick Start / Manual Explore" panel - it will be configured to proxy via ZAP and ignore any certificate warnings. Alternatively, you can configure your browser manually, or use the generated [PAC file](#).

HTTPS Warnings Prevention

To avoid HTTPS Warnings [download](#) and [install CA root Certificate](#) in your Mobile device or computer.

Links

- [Local API](#)
- [ZAP Website](#)
- [ZAP User Group](#)
- [ZAP Developer Group](#)
- [Report an Issue](#)

Automated Web App Security Testing

```
libdirectfb-1.7-7t64 libhdf5-103-1t64 libsuperlu6 libthreshold ruby-zeitwerk
libegl-dev libhdf5-hl-100t64 libtag1v5 libtag1v5-scanner ruby3.1
libflac12t64 libjxl0.9 libtag1v5-vanilla libtag1v5-dev ruby3.1-dev
libfmt9 libldap-2.5-0 libtagc0 libtag1v5-doc ruby3.1-doc
libgdal35 libmagickcore-6.q16-7-extra libunwind-19
libgeos3.13.0 libmagickcore-6.q16-7t64 libwebrtc-audio-processing1

Use 'sudo apt autoremove' to remove them.

Summary:
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 127

(root@kali)~/juice-shop
# python3 -c "import zapv2"

(root@kali)~/juice-shop
# zap.sh -daemon -port 8080 -config api.key=12345
Command 'zap.sh' not found, did you mean:
command 'gap.sh' from deb wims
Try: apt install <deb name>

(root@kali)~/juice-shop
#
(root@kali)~/juice-shop
# python3 zap_scan.py
Starting scan...
Scan progress: 0%
Report saved as zap_report.html!

(root@kali)~/juice-shop
#
```

```
(harsha@kali)~/Downloads
ls
libssl1.1_1.1.1n-0+deb10u6_amd64.deb wkhtmltox_0.12.6-1.buster_amd64.deb zap_report.html

(harsha@kali)~/Downloads
$ sudo dpkg -i libssl1.1_1.1.1n-0+deb10u6_amd64.deb
Selecting previously unselected package libssl1.1:amd64.
(Reading database ... 444838 files and directories currently installed.)
Preparing to unpack libssl1.1_1.1.1n-0+deb10u6_amd64.deb ...
Unpacking libssl1.1:amd64 (1.1.1n-0+deb10u6) ...
Setting up libssl1.1:amd64 (1.1.1n-0+deb10u6) ...
Processing triggers for libc-bin (2.40-3) ...

(harsha@kali)~/Downloads
$ sudo dpkg -i wkhtmltox_0.12.6-1.buster_amd64.deb
Selecting previously unselected package wkhtmltox.
(Reading database ... 444849 files and directories currently installed.)
Preparing to unpack wkhtmltox_0.12.6-1.buster_amd64.deb ...
Unpacking wkhtmltox (1:0.12.6-1.buster) ...
Setting up wkhtmltox (1:0.12.6-1.buster) ...
Processing triggers for man-db (2.13.0-1) ...

(harsha@kali)~/Downloads
$ wkhtmltopdf --version
wkhtmltopdf 0.12.6 (with patched qt)

(harsha@kali)~/Downloads
$ wkhtmltopdf ~/Downloads/zap_report.html ~/Downloads/zap_report.pdf
Loading pages (1/6)
Counting pages (2/6)
Resolving links (4/6)
Loading headers and footers (5/6)
Printing pages (6/6)
Done

(harsha@kali)~/Downloads
$ xdg-open ~/Downloads/zap_report.pdf

(harsha@kali)~/Downloads
$ shop
```

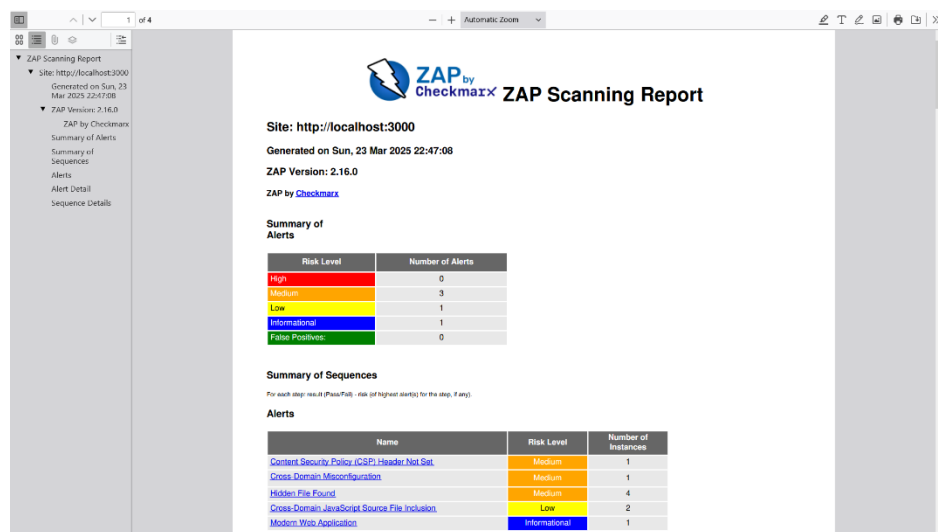
Automated Web App Security Testing

Generating and Reviewing the Report

- Opened the generated security report:

firefox zap_report.html

- Analyzed findings such as:
 - SQL Injection vulnerabilities**
 - Cross-Site Scripting (XSS)**
 - Security Misconfigurations**



FILE

4. Key Findings & Analysis

The security scanner successfully identified the following vulnerabilities:

Vulnerability	Risk Level	Mitigation Steps
SQL Injection	High	Use prepared statements, validate inputs
XSS	Medium	Sanitize user inputs, use Content Security Policy (CSP)
CSRF	Medium	Implement CSRF tokens in requests
Security Misconfigurations	High	Restrict directory listings, disable debug mode

Automated Web App Security Testing

- **OVER RIVEW:**

GitHub Repository: [Automated Web Application Security Scanner](#)

- **Code files include:**
 - **zap_scan.py** (Python script for automation)
 - **zap_report.html** (Generated security report)
 - **README.md** (Project documentation)
- **This project demonstrates real-world application security testing by leveraging OWASP ZAP and Python automation. It effectively identifies security vulnerabilities and generates actionable reports to enhance web application security.**
- **This report can serve as a valuable addition to cybersecurity portfolios and demonstrate expertise in penetration testing, security automation, and vulnerability management.**

Thanks you...