OWASP ZAP (Zed Attack Proxy)
-A detailed case study

Team Details:

Abishek E	CB.EN.U4CSE22001
Aditya R	CB.EN.U4CSE22004
Guhanesh T	CB.EN.U4CSE22015
Prashanna R	CB.EN.U4CSE22036

Introduction :

OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner designed to identify vulnerabilities in web applications. Developed by the Open Web Application Security Project (OWASP), it is widely used by developers, security analysts, and penetration testers to perform security assessments. It acts as a man-in-the-middle proxy, intercepting and inspecting HTTP/S traffic between the browser and the target application.

Why ZAP?

Open Source: Free to use, modify, and distribute.

Versatility: Supports automated scans for beginners and manual tools for experts.

Community: Backed by OWASP's global security community.

Integration: Fits into DevOps pipelines for continuous security testing.

FEATURES :

Passive Scanning:

- Non-intrusive analysis of traffic.
- Detects issues like insecure cookies, mixed content, or misconfigured CORS headers.

Active Scanning:

- Simulates attacks to uncover vulnerabilities (directory traversal, server-side injection).
- Scan Policies: Customize rulesets to exclude false positives or focus on specific risks.

Spider Attack:

- It is used to automatically discover new resources/URLs on your website
- It visits those URLs, identifies the hyperlinks and adds them to the list.

Quick Attack:

- This helps to test the application using ZAP in the quickest way possible. (combines both spidering and active scanning)
- ZAP will use its spider to crawl through the application, which will automatically scan all of the pages discovered. It will then use the active scanner to attack all of the pages.
- This is a useful way to perform an initial assessment of an application.

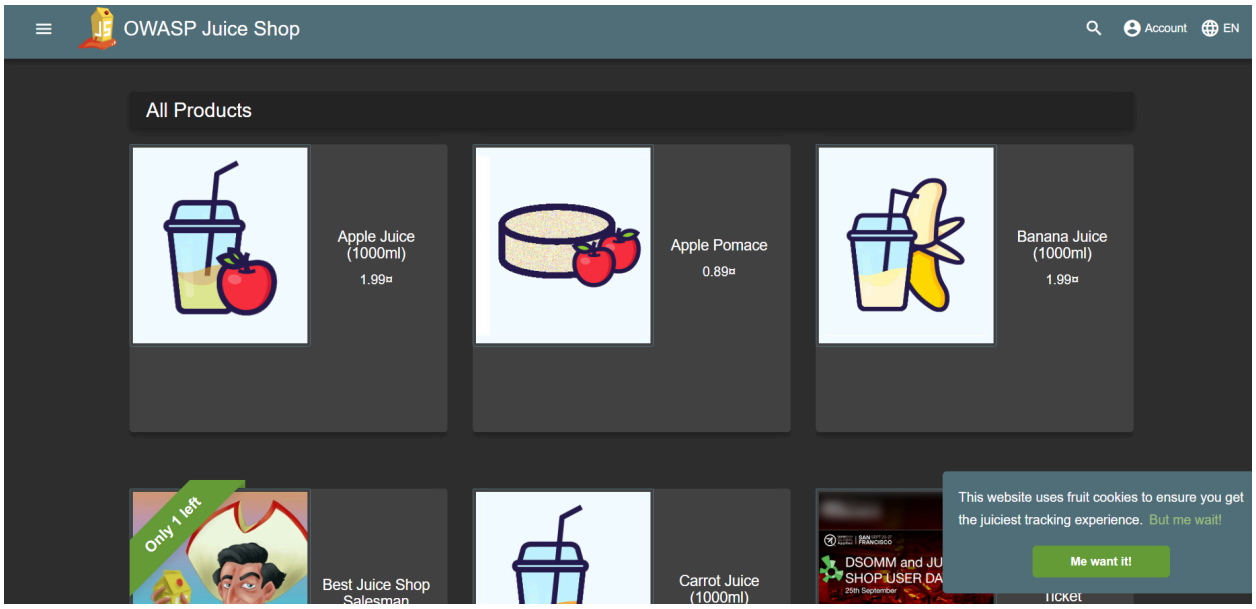
Alerts:

- Alerts are thrown as results of attacks performed by Spider/Active Scan (or any other attack).
- Alerts are the potential vulnerabilities which are flagged as High, Medium, or Low according to the risk level.

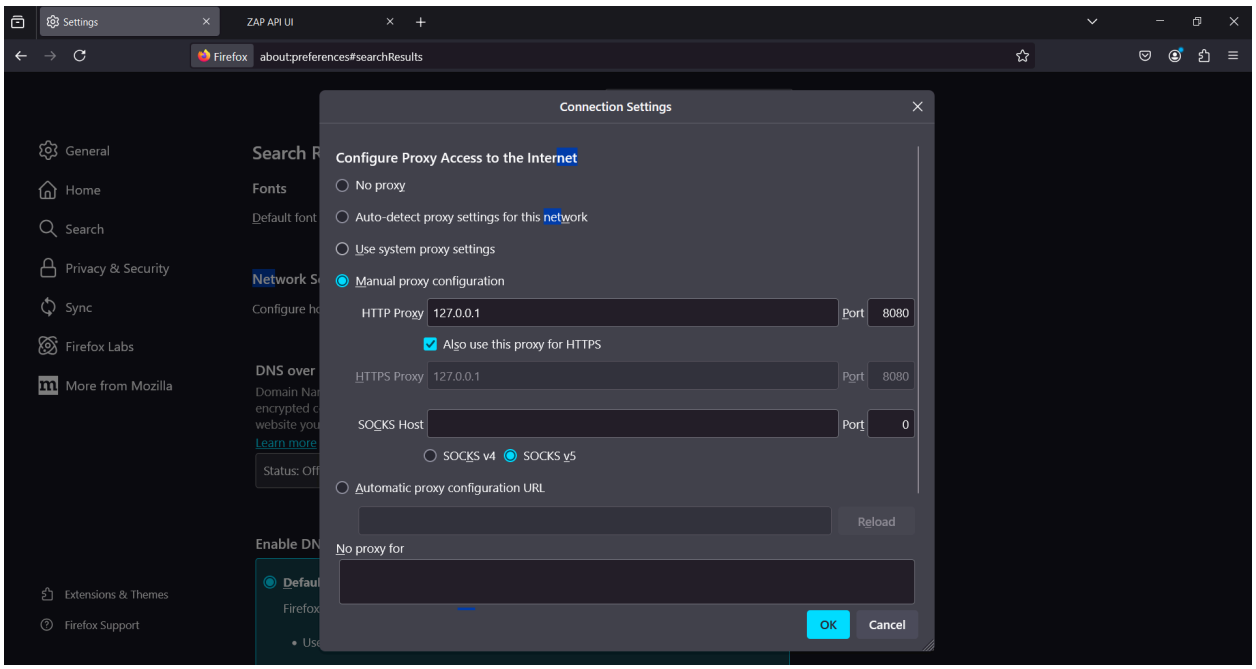
Setup:

Configure Browser to Use ZAP as a Proxy:

To explore the application, ZAP has provided a few web applications to attack and scan.We chose the juice app.

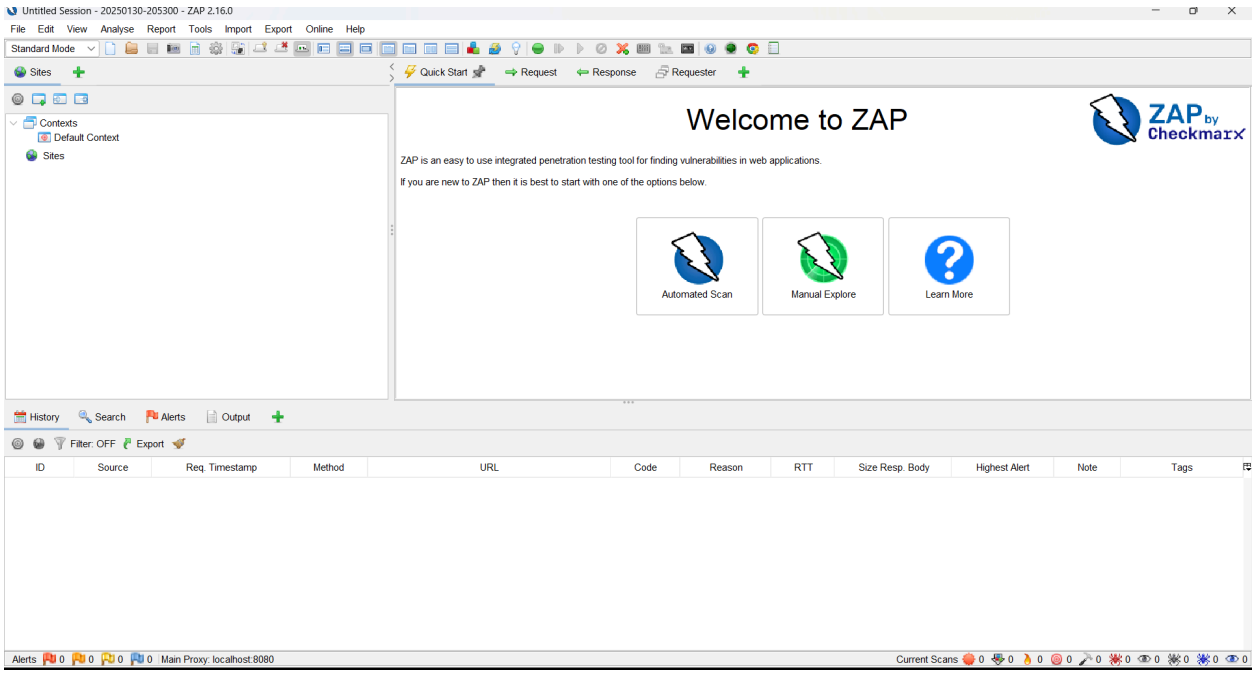


The ZAP server is running in localhost.To use it in firefox, it is set as a proxy site in network settings.



Dashboard:

Version:2.16.0



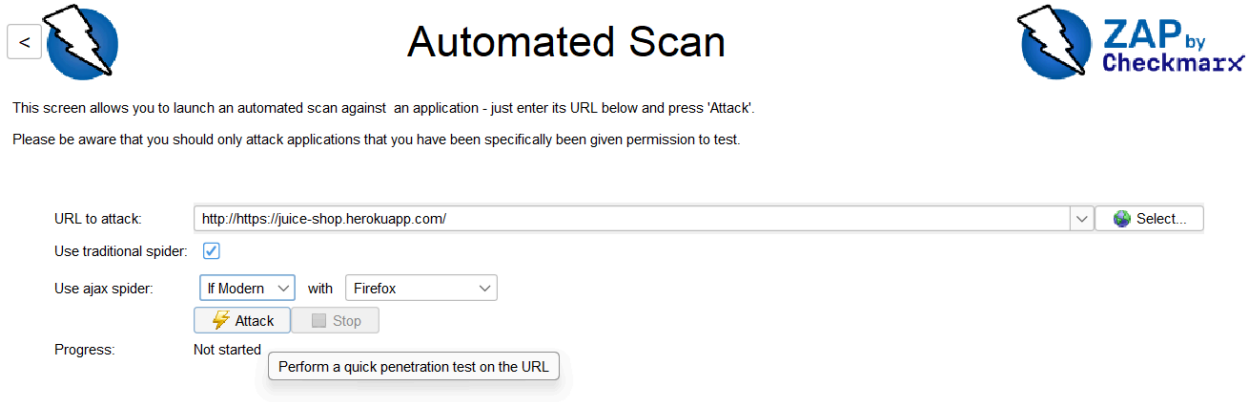
It has 4 main modes for exploration: Safe,Protected,Standard and Attack mode.
Reports can be generated as a pdf/HTML file covering all security aspects of the web application.
Further features are explored as below

Exploration Summary:

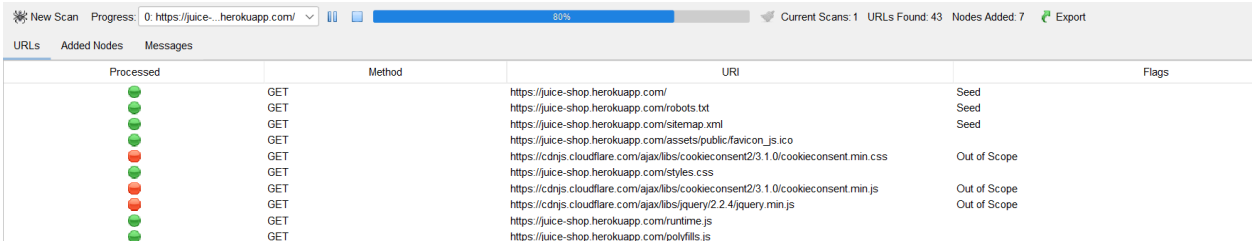
Spidering (Crawling a Website)

A **Spider** in OWASP ZAP is a tool that **crawls** a website to discover all available pages, links, and resources

Attack:



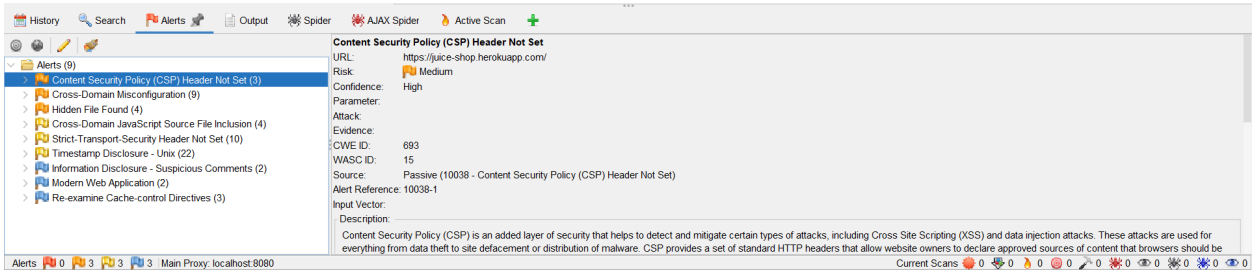
Result Log:



It shows the different URI in the page and their status (discoverable/not-discoverable).

Passive Scanning

When using the application as a proxy to scan a web app, the passive scanning checks for vulnerabilities in the application and displays them in the alerts page.



It gives a detailed explanation of the alert and flags it as high/low/medium risk.

Active Scanning (Automated Vulnerability Testing)

Report:

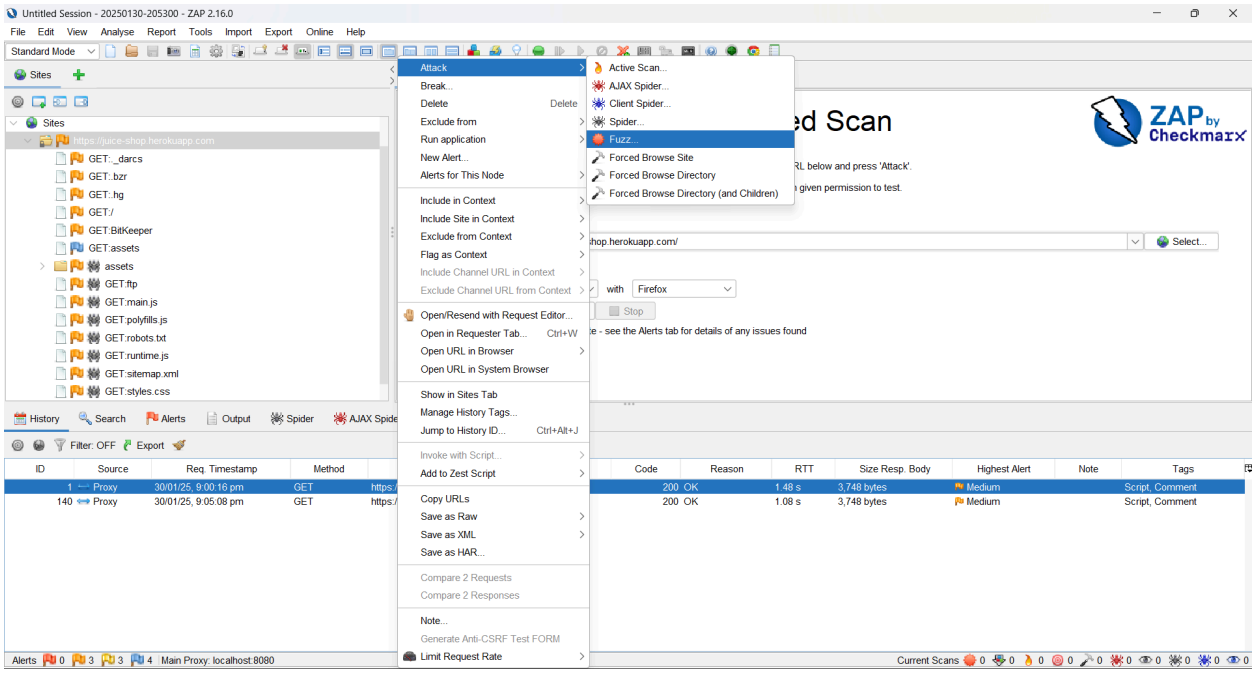
Tests all possible attacks like SQL Injection, XSS, Broken Authentication etc., to check the integrity of the application.

Host: https://juice-shop.herokuapp.com

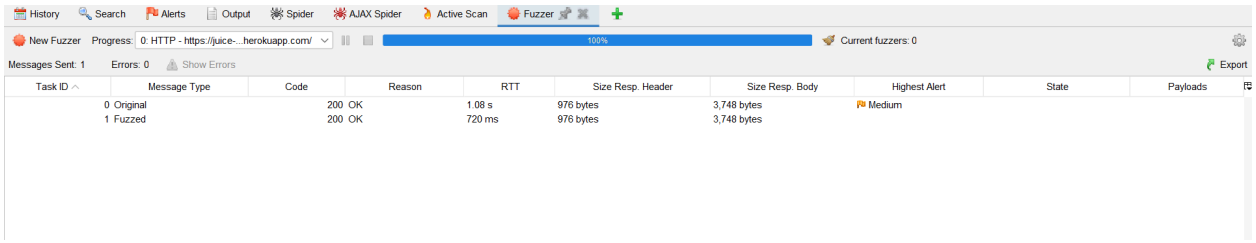
	Strength	Progress	Elapsed	Reqs	Alerts	Status
SQL Injection - MySQL	Medium		00:00.860	0	0	
SQL Injection - Hypersonic SQL	Medium		00:00.842	0	0	
SQL Injection - Oracle	Medium		00:01.390	0	0	
SQL Injection - PostgreSQL	Medium		00:00.865	0	0	
SQL Injection - SQLite	Medium		00:00.886	0	0	
Cross Site Scripting (DOM Based)	Medium		00:00.477	0	0	
SQL Injection - MsSQL	Medium		00:00.897	0	0	
Log4Shell	Medium		00:00.295	0	0	
Spring4Shell	Medium		00:17.548	27	0	
Server Side Code Injection	Medium		00:00.846	0	0	
Remote OS Command Injection	Medium		00:01.584	0	0	
XPath Injection	Medium		00:00.692	0	0	
XML External Entity Attack	Medium		00:01.078	0	0	
Generic Padding Oracle	Medium		00:00.811	0	0	
Cloud Metadata Potentially Exposed	Medium		00:02.669	9	0	
Server Side Template Injection	Medium		00:02.397	0	0	
Server Side Template Injection (Blind)	Medium		00:00.601	0	0	
Directory Browsing	Medium		00:02.719	13	0	
Buffer Overflow	Medium		00:00.857	0	0	
Format String Error	Medium		00:00.847	0	0	
CRLF Injection	Medium		00:01.120	0	0	
Parameter Tampering	Medium		00:01.106	0	0	
ELMAH Information Leak	Medium		00:00.550	1	0	
Trace.axd Information Leak	Medium		00:02.160	3	0	
.htaccess Information Leak	Medium		00:01.854	3	0	
.env Information Leak	Medium		00:01.717	3	0	
Spring Actuator Information Leak	Medium		00:02.452	2	0	
Hidden File Finder	Medium		00:18.219	50	4	
XSLT Injection	Medium		00:17.011	0	0	
GET for POST	Medium		00:02.190	0	0	
User Agent Fuzzer	Medium		02:53.630	147	24	
Script Active Scan Rules	Medium			0	0	
SOAP Action Spoofing	Medium			0	0	
SOAP XML Injection	Medium			0	0	

Fuzzing (Testing Input Fields for Security Weaknesses)

Tests an input field or a URL parameter in the application.



On a specific URL, the fuzzer tests the application by sending **unexpected, malformed, or random data** into input fields, URLs, or API endpoints.



Man-in-the-Middle (Intercept & Modify Requests/Responses)

A specific endpoint can be selected and its request header can be changed to simulate man-in-middle attacks.
Responses can be seen.

Final Report generation:

A final detailed report was generated showing all security concerns in the web app.

REPORT LINK:

https://drive.google.com/file/d/1lSdkN9OuR2b6boLNFMIC5rWv8PbREURh/view?usp=sharing