**PRACTICAL - 4**

**AIM:** Perform a comprehensive scan of a web application, detecting and documenting security vulnerabilities using OWASP ZAP.

**Solution:**

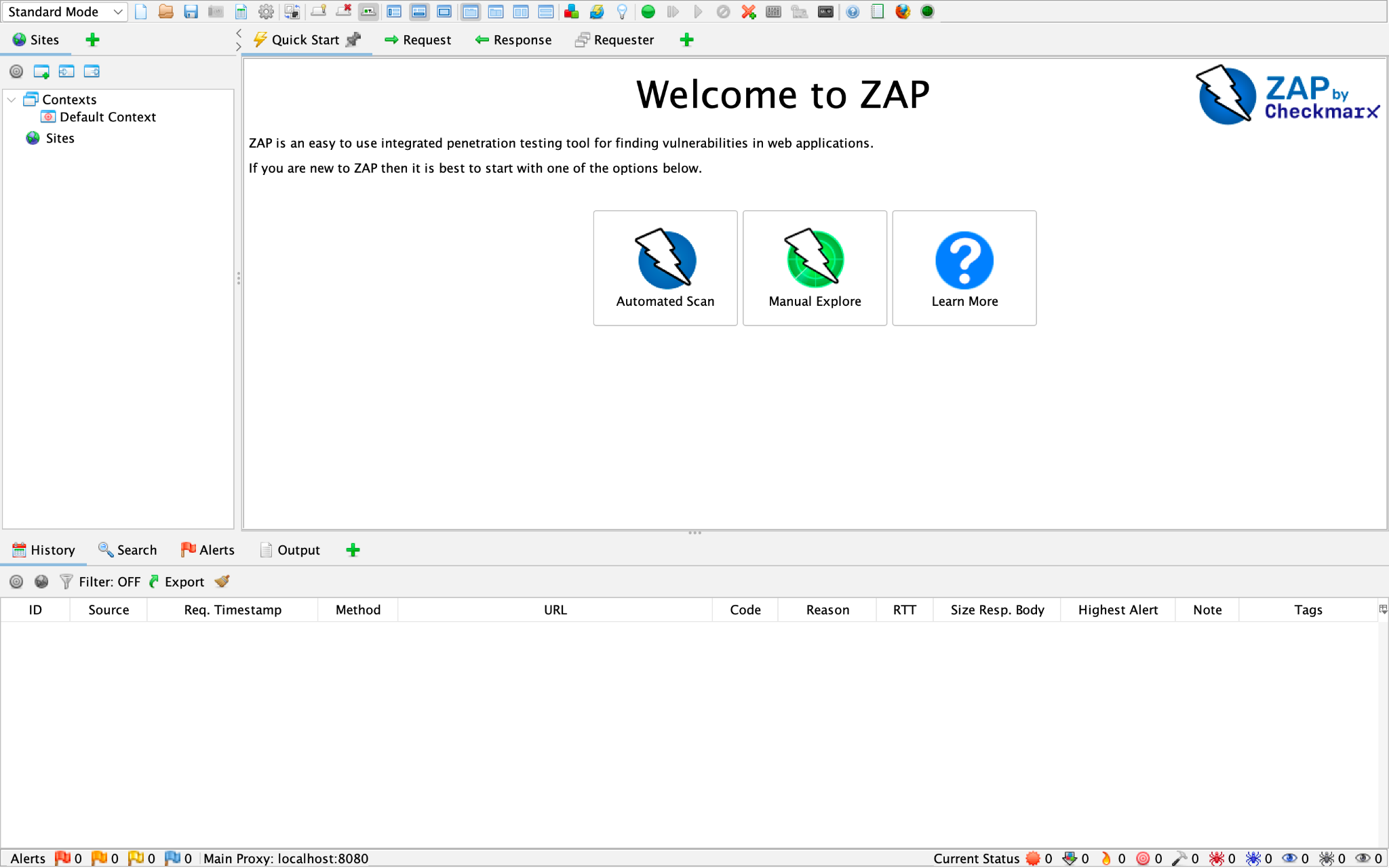
**Introduction**

**OWASP ZAP (Zed Attack Proxy)** is a widely used, open-source security tool developed by the **Open Worldwide Application Security Project (OWASP)**. It is specifically designed to help developers and security professionals identify vulnerabilities in web applications through both **automated scanning** and **manual testing**. As one of the most active open-source security projects, ZAP plays a crucial role in detecting a wide range of issues, including those listed in the **OWASP Top 10** such as **Cross-Site Scripting (XSS)**, **SQL Injection** and **Broken Authentication**.

**Procedure**

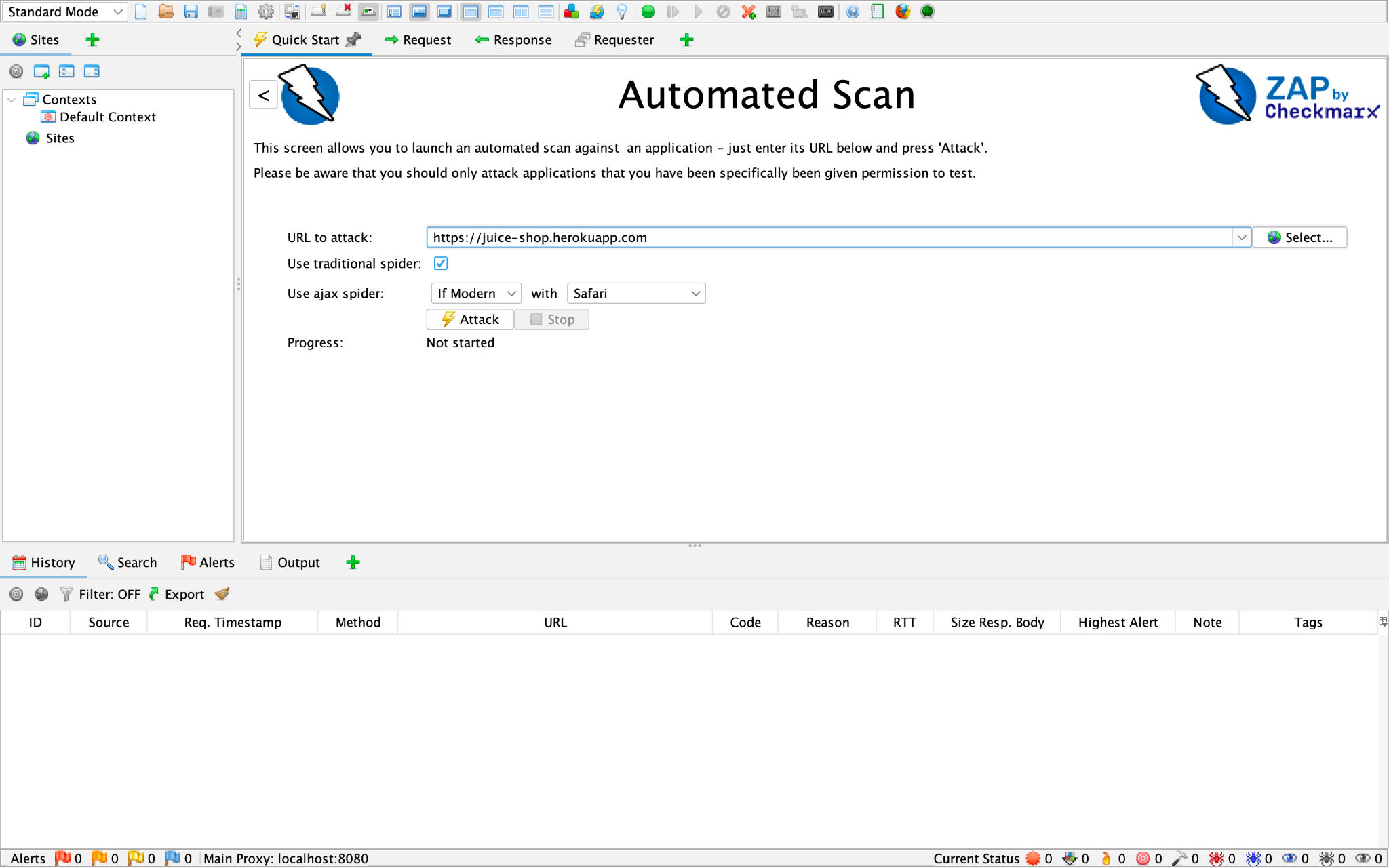
**Step 1: Install and Launch ZAP**

OWASP ZAP was downloaded and installed from the official website. The application was launched in **Standard Mode** for full functionality.



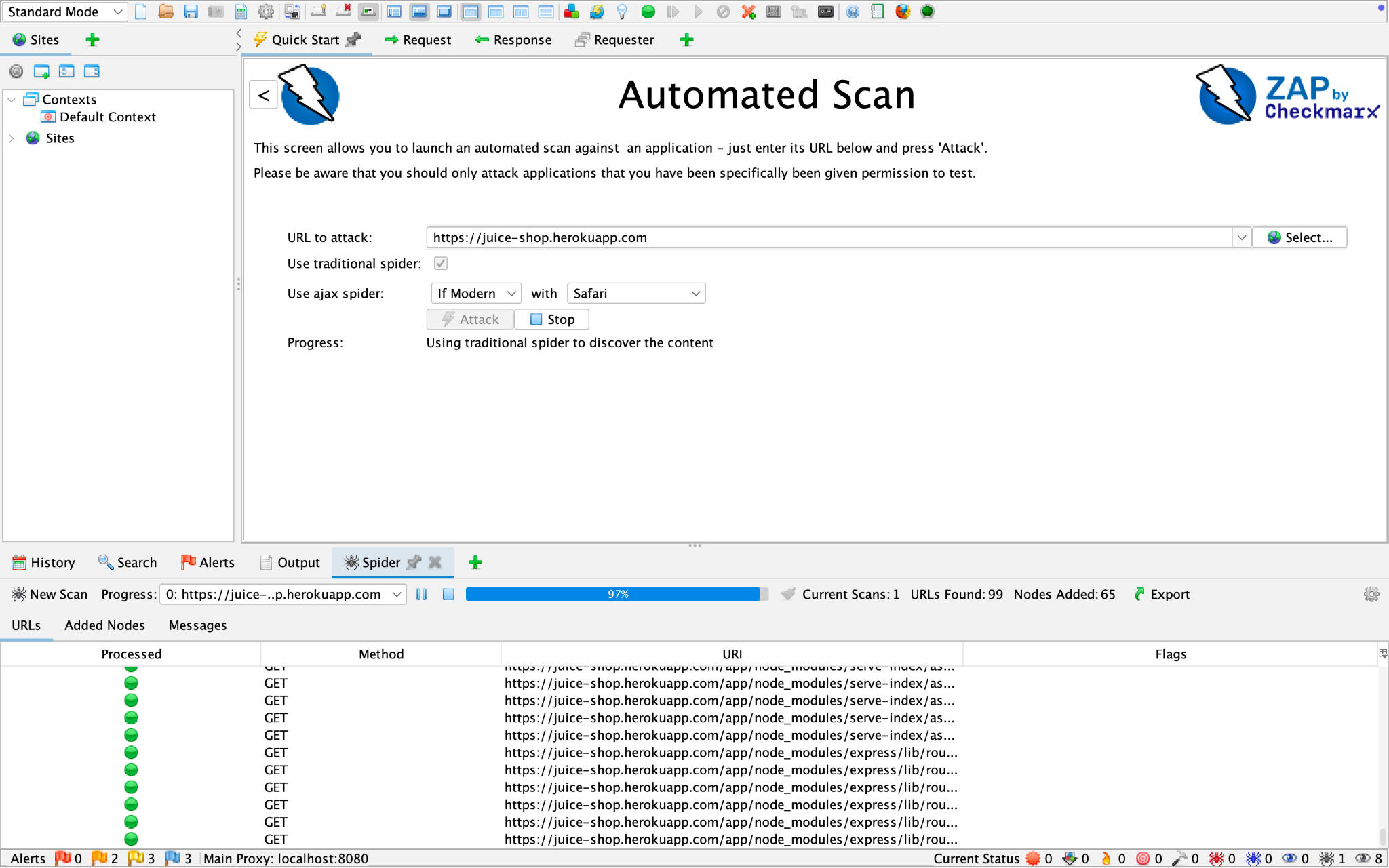
**Step 2: Select Target Web Application**

The chosen target for testing was **OWASP Juice Shop**, a deliberately insecure application hosted at: “ <https://juice-shop.herokuapp.com> ”



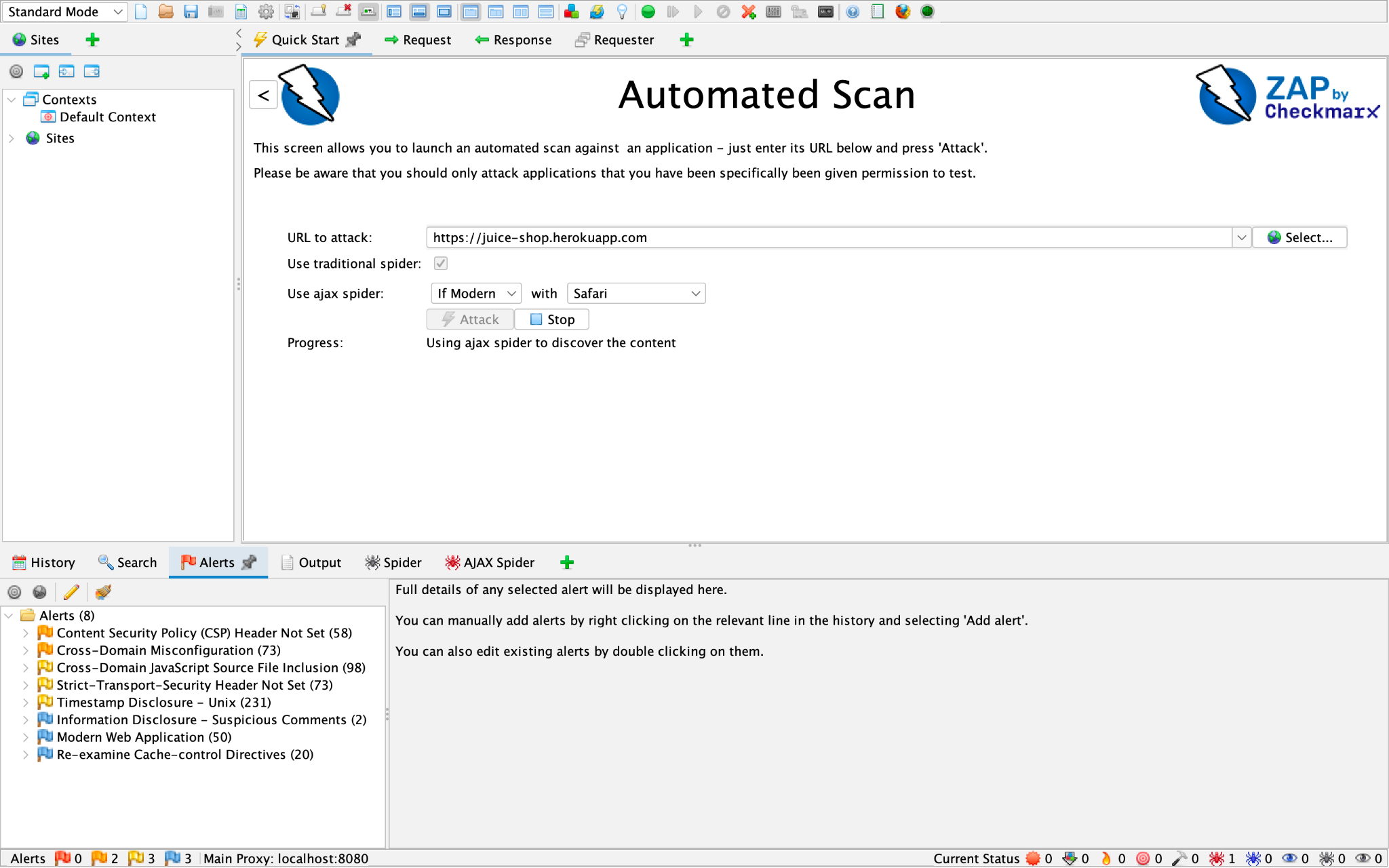
**Step 3: Configure Automated Scan**

The **Automated Scan** option was selected from the **Quick Start** menu. The target URL (<https://juice-shop.herokuapp.com>) was entered and both **Traditional Spider** and **AJAX Spider** options were enabled. The scan was initiated by clicking the **Attack** button and its progress was monitored through the ZAP interface.



**Step 4: Analyze Alerts**

After scanning, OWASP ZAP provided a **list** **of** **vulnerabilities**. Each alert included the name, risk level, affected element, evidence and suggested mitigation.



**Vulnerabilities Identified**

| **Vulnerability** | **Risk** | **Description** | **Affected Element** | **Suggested Mitigation** |
| --- | --- | --- | --- | --- |
| **Cross-Domain JavaScript Source File Inclusion** | Medium | Application loads JavaScript from third-party domains | External JS script | Load scripts from trusted, secure domains only |
| **Strict-Transport-Security Header Not Set** | Medium | Lack of HSTS header leaves the site open to downgrade and MITM attacks | HTTP response headers | Add Strict-Transport-Security header |
| **Content Security Policy Header Not Set** | Medium | CSP header missing, allowing untrusted script execution | HTTP response headers | Implement Content Security Policy to restrict resources |
| **Cross-Domain Misconfiguration** | Medium | Access-Control-Allow-Origin is set to wildcard | Response header | Restrict CORS access to trusted origins |
| **Timestamp Disclosure – Unix** | Low | Server reveals last-modified timestamps in response | Multiple endpoints | Avoid exposing detailed timestamps in responses |

## 

## 

## **Mitigation Recommendations**

To improve security posture and fix the above vulnerabilities, the following general mitigations are recommended:

* Implement secure HTTP headers: HSTS, CSP, X-Content-Type-Options, etc.
* Avoid loading untrusted third-party scripts
* Configure CORS policy properly
* Use input validation and sanitization
* Regularly audit dependencies and JavaScript libraries
* Avoid exposing unnecessary server information or metadata