

OWASP ZAP

Web Application Vulnerability Scanner



Kushal Goswami

PTID-CHE-SEP-25-219

Introduction

Why Vulnerability Management Matters

Vulnerability management is the proactive discipline of finding and fixing security flaws before attackers can exploit them. In today's threat landscape, regular scanning is essential to detect emerging vulnerabilities, reduce organizational risk, and maintain a robust security posture.

OWASP ZAP stands out as a powerful, open-source solution offering comprehensive scanning capabilities with detailed, actionable reporting that security teams can trust.



About OWASP ZAP



Open-Source Security

OWASP ZAP (Zed Attack Proxy) is a community-driven, open-source tool designed specifically for testing web application security vulnerabilities.



Comprehensive Detection

Identifies critical vulnerabilities including missing security headers, insecure cookie configurations, SQL injection, XSS, and more.



Cross-Platform

Completely free and supports Windows, Linux, and macOS environments, making it accessible for any security team.



Flexible Scanning

Offers both automated scanning for rapid assessment and manual exploration modes for deep security testing with detailed reports.

ZAP Testing Workflow

Follow this structured approach to conduct comprehensive web application security assessments using OWASP ZAP.

01

Select Scan Mode

Choose between Automated Scan for quick vulnerability detection or Manual Explore for hands-on testing and discovery

02

Password Testing

Utilize the Fuzzer tool to test password fields against wordlists and identify weak authentication mechanisms

03

Authentication Setup

Configure Form-Based Authentication to enable deeper scanning of authenticated areas and session management

04

Report Generation

Generate comprehensive reports detailing discovered vulnerabilities, severity levels, and recommended remediation steps

Automated Scan

Quick Vulnerability Assessment

The automated scan provides rapid security assessment of web applications with minimal configuration required.

→ Launch Quick Start

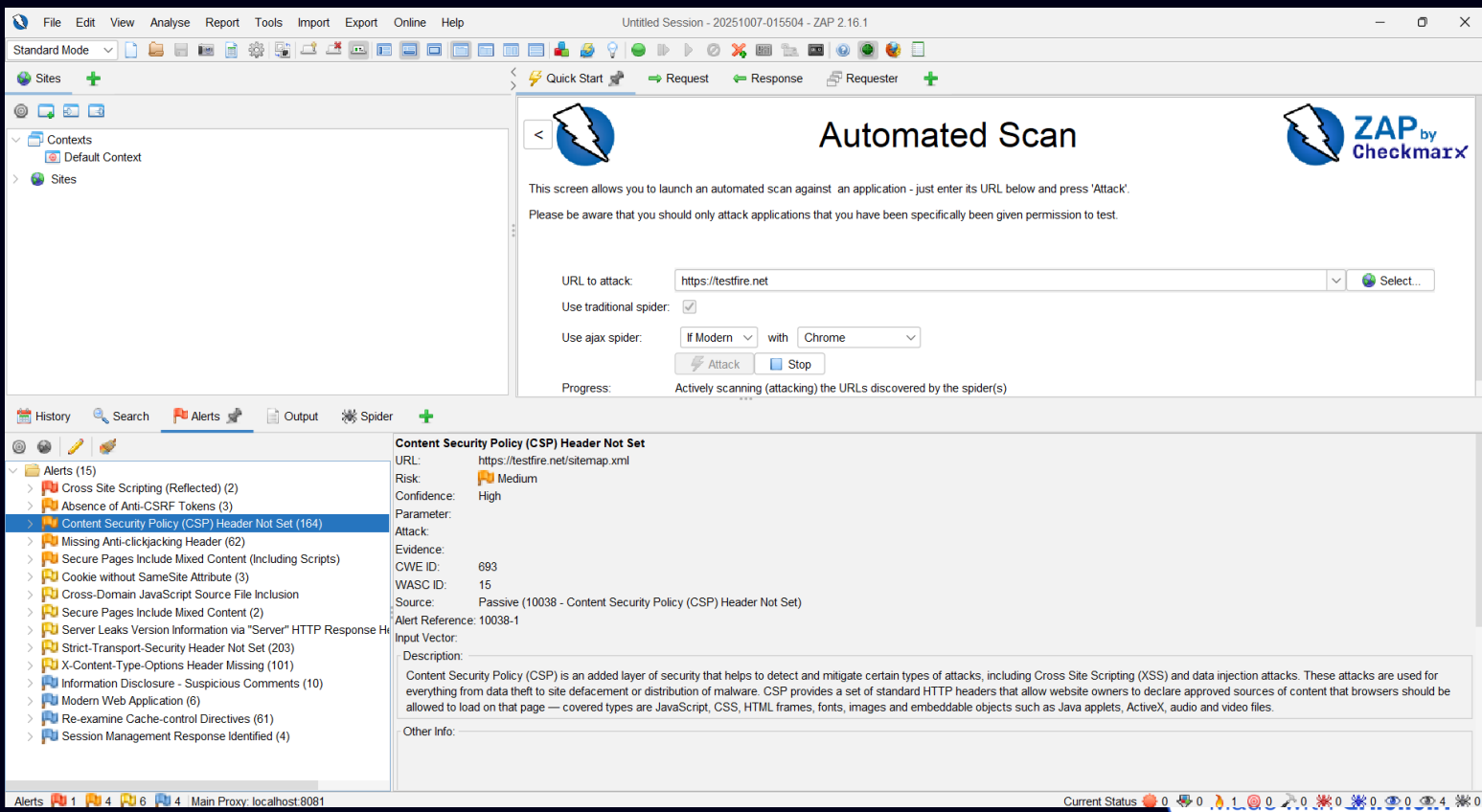
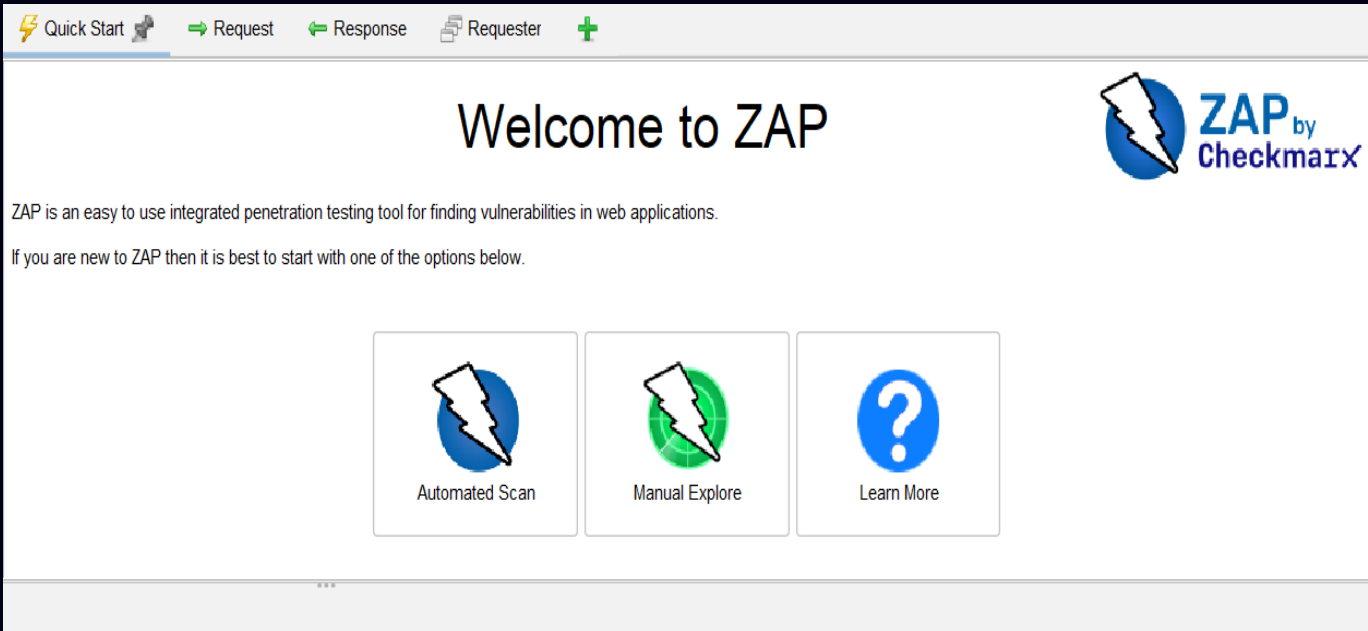
Open Quick Start tab and enter your target URL (e.g., <https://testfire.net>) into the designated field.

→ Initiate Attack

Click the 'Attack' button to begin automatic scanning and crawling of all accessible web pages and endpoints.

→ Review Findings

Navigate to the 'Alerts' tab to examine discovered vulnerabilities, each categorized by severity level (High, Medium, Low).



Manual Explore

Deep Dive into Application Behavior

Manual exploration allows security testers to interact naturally with the application while ZAP captures all HTTP traffic, revealing vulnerabilities that automated scans might miss.

Step 1: Launch Browser

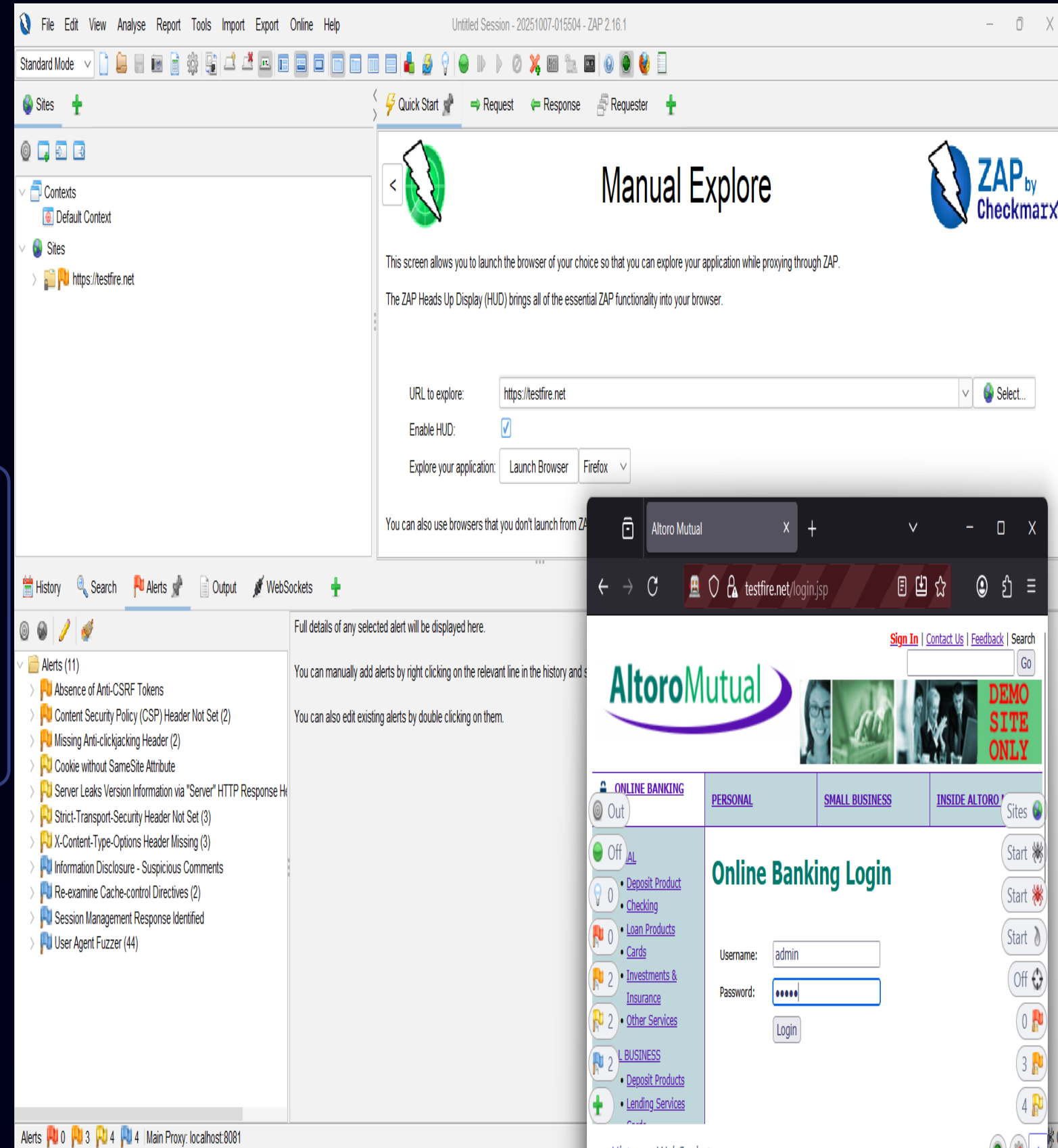
Open Quick Start → Manual Explore → Launch Browser. ZAP will configure a proxied browser instance automatically.

Step 2: Interactive Browsing

Manually navigate through the target site, clicking links, submitting forms, and exploring all functionality while ZAP captures requests and responses in real-time.

Step 3: Test Critical Areas

Focus on testing authentication pages, form submissions, session handling, and hidden sections that require user interaction to access.



Password Cracking (Fuzzer)

The Fuzzer tool in ZAP enables ethical password strength testing by systematically trying different password combinations against login forms to identify weak credentials.



Capture POST Request

Locate and capture the POST request from the login form in the Sites tree panel after attempting a login.



Configure Fuzzer

Right-click the request → Attack → Fuzz → Highlight the password field parameter for targeted fuzzing.



Load Wordlist

Load a password dictionary file (such as rockyou.txt or a custom wordlist) containing common passwords to test.



Execute Attack

Start the Fuzzer to systematically test each password. Monitor response codes and sizes to identify successful authentication attempts.

The screenshot displays the ZAP Fuzzer interface. The 'Attack' menu is open, showing the 'Fuzz...' option. The 'Fuzz Locations' panel shows the selected request and the 'Body' tab. The 'Add Payload' dialog is open, showing the 'Strings' type and the contents 'admin', 'password', and 'user123'. The 'Fuzzer' tab is active, showing the progress bar at 100% and the 'Current fuzzers: 0' status. The 'Messages Sent' table is visible, showing the results of the fuzzing attack.

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
1	Fuzzed	302	Found	803 ms	277 bytes	0 bytes			admin, admin
0	Original	302	Found	779 ms	126 bytes	0 bytes	Low		
2	Fuzzed	302	Found	792 ms	126 bytes	0 bytes			admin, user
3	Fuzzed	302	Found	772 ms	126 bytes	0 bytes			admin, password
4	Fuzzed	302	Found	792 ms	126 bytes	0 bytes			password, admin
5	Fuzzed	302	Found	802 ms	126 bytes	0 bytes			password, user
6	Fuzzed	302	Found	752 ms	126 bytes	0 bytes			password, password
7	Fuzzed	302	Found	802 ms	126 bytes	0 bytes			user123, admin
8	Fuzzed	302	Found	792 ms	126 bytes	0 bytes			user123, user
9	Fuzzed	302	Found	790 ms	126 bytes	0 bytes			user123, password

Authenticated Scan

Testing Behind the Login

Authenticated scanning enables ZAP to test vulnerabilities in protected areas of the application that require valid user credentials to access.

- **Create Context**

Define a new context including your target site, then configure Form-Based Authentication with login URL and form parameters.

- **Configure User**

Add a test user account with valid credentials and enable Forced User Mode to maintain automatic re-authentication.

- **Run Authenticated Scan**

Execute both Active Scan and Spider with the authenticated user to achieve comprehensive coverage of protected areas.

- **Compare Results**

Analyze the difference in vulnerability alerts discovered before and after authentication to identify access-control issues.

The image is a collage of five screenshots from the OWASP ZAP (Zed Attack Proxy) application, illustrating the steps to perform an authenticated scan.

- Top Left:** Shows the 'Sites' tree with 'https://testfire.net' selected. A context menu is open, showing options like 'Include in Context', 'Exclude from Context', and 'New Context'.
- Top Right:** Shows the 'Session Properties' dialog for '2: Authentication'. It is configured for 'Form-based Authentication' with the 'Login Form Target URL' set to 'https://testfire.net/doLogin' and 'URL to GET Login Page' set to 'https://testfire.net/doLogin'. The 'Login Request POST Data (if any)' is 'uid=admin&passw=1234&btnSubmit=Login'. The 'Username Parameter' is 'uid' and the 'Password Parameter' is 'passw'.
- Middle Left:** Shows the 'Add a New User' dialog. The 'User Name' is 'testuser', 'Enabled' is checked, 'Username' is 'admin', and 'Password' is masked with dots.
- Bottom Left:** Shows the 'Alerts (15)' list. It includes categories like 'Cross Site Scripting (DOM Based) (6)', 'SQL Injection (2)', 'SQL Injection - Authentication Bypass (2)', 'Absence of Anti-CSRF Tokens', 'Content Security Policy (CSP) Header Not Set (3)', 'Missing Anti-clickjacking Header (3)', 'Cookie No HttpOnly Flag', 'Cookie Without Secure Flag', 'Cookie without SameSite Attribute (3)', 'Server Leaks Version Information via "Server" HTTP Response Header', 'Strict-Transport-Security Header Not Set (4)', 'X-Content-Type-Options Header Missing (4)', 'Information Disclosure - Suspicious Comments', 'Re-examine Cache-control Directives (3)', and 'Session Management Response Identified (3)'.
- Bottom Right:** Shows the 'Active Scan' and 'Spider' configuration dialogs. Both are set to start at 'https://testfire.net', use 'Default Policy', 'Context: https://testfire.net', 'User: testuser', and 'Recurse' is checked.

Scan Results Analysis

Key Findings

Automated Scan

Detected multiple low and medium-risk vulnerabilities including missing security headers, outdated libraries, and potential XSS vectors.


Manual Exploration

Revealed hidden input fields, weak cookie configurations without HttpOnly/Secure flags, and exposed administrative endpoints.

Authenticated Testing

Discovered critical vulnerabilities behind authentication including SQL injection points, insecure direct object references, and privilege escalation paths.

ZAP by Checkmarx Scanning Report

Generated with  ZAP on Tue 7 Oct 2025, at 02:49:32

ZAP Version: 2.16.1

ZAP by [Checkmarx](#)

Contents

- [About This Report](#)
 - [Report Parameters](#)
- [Summaries](#)
 - [Alert Counts by Risk and Confidence](#)
 - [Alert Counts by Site and Risk](#)
 - [Alert Counts by Alert Type](#)
- [Alerts](#)
 - [Risk=High, Confidence=High \(1\)](#)
 - [Risk=High, Confidence=Medium \(2\)](#)
 - [Risk=Medium, Confidence=High \(1\)](#)
 - [Risk=Medium, Confidence=Medium \(1\)](#)
 - [Risk=Medium, Confidence=Low \(1\)](#)
 - [Risk=Low, Confidence=High \(2\)](#)
 - [Risk=Low, Confidence=Medium \(4\)](#)
 - [Risk=Informational, Confidence=Medium \(3\)](#)
 - [Risk=Informational, Confidence=Low \(1\)](#)
- [Appendix](#)

Comprehensive reports summarized all findings categorized by severity (Critical, High, Medium, Low, Informational) with detailed remediation recommendations and OWASP references.

Conclusion

Proven Capability

This project successfully demonstrated OWASP ZAP's comprehensive ability to detect, analyze, and report web application vulnerabilities across multiple scanning methodologies.

Security Best Practices

Emphasized the critical importance of continuous security scanning, regular vulnerability assessments, and timely remediation in maintaining application security.

Knowledge Growth

This hands-on project significantly strengthened practical understanding of web application security principles, vulnerability management workflows, and ethical penetration testing techniques.

THANK YOU