Security Testing

Security Testing, Tools

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#QA-FrontEnd

Table of Contents

1. **Security Testing**: Overview and Concepts
   - Key Elements
   - Principles
   - Types
2. **Security Testing Tools**
3. **OWASP ZAP –** popular security testing tool
4. **Security attacks**

Security Testing

Overview

Security Testing: Overview

- Evaluates the ability of a software system to protect against:
  - Unauthorized **access, attacks,** other security **threats**
- Main goal
  - **Identify** vulnerabilities and weaknesses
  - **Checks** whether software is vulnerable to threats
  - **Provides evidence** that the system and information are safe and reliable
- Additional Points
  - **Regular testing** and **updates** to **security protocols** are crucial to maintaining security
  - Security testing encompasses various areas: **application security**, **network security**, and **system security**

Key elements

- Assets
  - Things that we need to **protect**
  - Data, software, hardware, intellectual property, people, and processes
- Risk
  - **Potential** for loss, damage, or destruction of an assets
- Threats
  - **Activities** that can cause damage to asset
  - Cyber attacks, malware, viruses, or physical theft or damage
- Vulnerabilities
  - **Weakness** in your hardware, software, or procedures

Real-World Examples

- **Assets**
  - Company data, customer information, proprietary software, intellectual property, employees' access credentials
- **Risk**
  - Financial loss from data breaches, reputation damage, operational disruptions
- **Threats**
  - Phishing attacks leading to data theft, malware causing system downtime, unauthorized physical access to hardware
- **Vulnerabilities**
  - Unpatched software, weak passwords, lack of encryption

Principles

- Confidentiality
  - **Sensitive information** is only accessible **by authorized users**
  - **Top priority** for most organizations
- Integrity
  - **Consistency**, **accuracy**, and **trustworthiness** of data
  - Particularly important for **financial**, **medical**, or other **critical data**, where inaccuracies or modifications could have **serious consequences**
- These principles often **overlap**; Ensuring data integrity also supports confidentiality and vice versa

Security Testing: Principles

- Availability
  - Information is **accessible** and **usable** when **needed**
- Authentication
  - Confirms the **identity** of the user
  - Critical component, important to **ensure** that the **user** is who they claim to be
  - Emphasis on **multi-factor authentication** (e.g., password + SMS code)
- Authorization
  - Specifies the **access rights** of users
  - Grants access only to **resources needed**, based on the **user's role**

Security Testing: Principles

- Non-repudiation
  - **Proof** that a message or transaction was **sent** or **received** and **cannot be denied** by either party
- Resilience
  - Ability of a system to **withstand internal** and **external attacks** and **quickly recover** from them
  - Importance of security **monitoring** and **logging**:
    - Continuous **monitoring** for unusual activity and immediate response
  - Regular **security drills** and **incident response** plans:
    - Conducting **simulations** of cyber attacks to prepare for real incidents

Security Testing Types

Security Testing Types

- Vulnerability Scanning
  - Scan the system for **known vulnerabilities**
  - Search for **outdated software**, **unpatched systems**, **misconfigured settings**, **weak passwords**, etc.
  - **Example:** Running a scan to find unpatched versions of operating systems
- Security Scanning
  - Identifying network and system **configuration weakness**
  - Analyzing network **protocols**, **services**, and **applications** for potential security gaps
  - **Example**: Reviewing firewall configurations to ensure they are properly set up

Security Testing Types

- Penetration testing
  - **Simulates an attack** by a malicious hacker
  - **Example**: Hiring a third-party firm to attempt to breach network defenses
- Risk Assessment
  - Involves the **analysis** of security risks observed in the organization
  - **Example**: Analyzing the risk of data breaches for a company handling sensitive customer information

Security Testing Types

- Security Auditing
  - Inspects **applications** and **operating systems** for **security flaws**
  - Ensures **compliance** with security **standards** and **policies**
  - **Example**: Conducting a thorough review of a company's security policies and practices to ensure they meet industry standards
- Ethical hacking
  - Penetration testing conducted by a security professional (white hat hacking)
  - Identifies security weaknesses from the perspective of an attacker
  - Example: An ethical hacker performing a simulated attack to test the organization's defenses

Security Testing Types

- **Posture Assessment**
  - Combines **Security Scanning**, **Ethical Hacking**, and Risk **Assessments** to show an overall security posture of an organization
  - Provides a **full view** of an organization's **security standing**, identifying strengths and areas for improvement
  - Evaluates the **effectiveness** of current **security controls** and **measures**
  - Helps in **prioritizing security investments** and initiatives based on the identified risks and vulnerabilities
  - **Example**: A comprehensive report detailing the overall security readiness of an organization

Security Risks and Resources

Security Resources

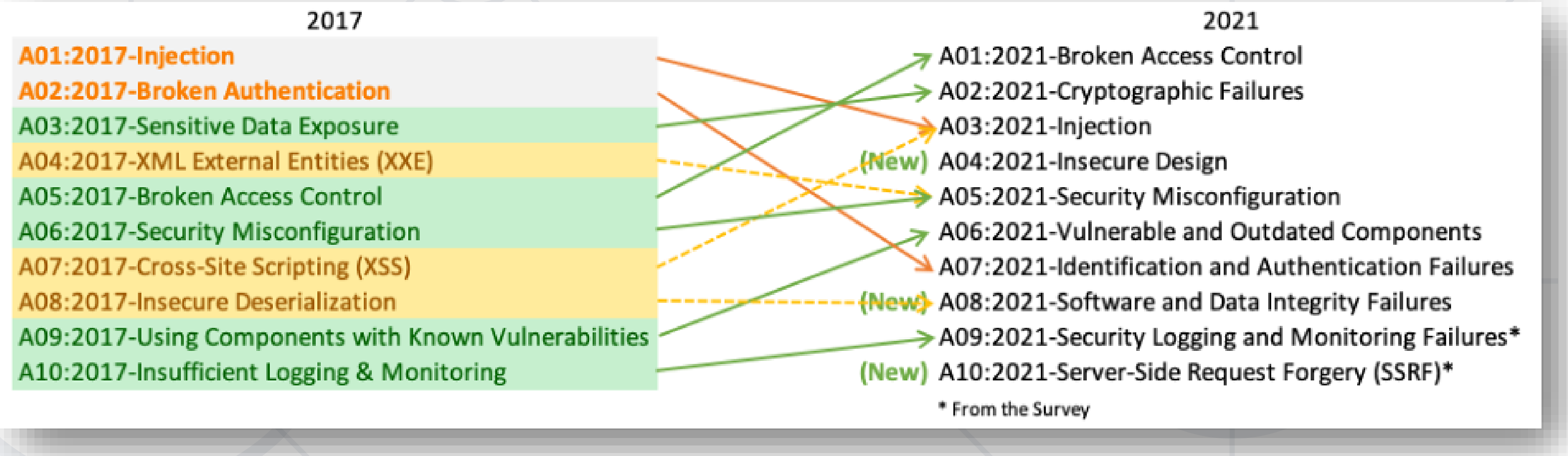
- **SANS Institute**
  - Launched in 1989 as a cooperative for information security thought leadership, it is SANS' ongoing mission to empower cyber security professionals with the practical skills and knowledge they need to make our world a safer place
  - SANS Institute also offers a wealth of resources such as courses, certifications, and whitepapers that can further aid in enhancing security skills and knowledge
- **SANS Top 25**
  - TOP 25 Most Dangerous Software Errors
  - The Top 25 provides much needed guidance for software developers focusing on eliminating software security defects in their products

Security Sources

- The Open Worldwide Application Security Project (**OWASP**)
  - Non-profit foundation that works to improve the security of software
- **OWASP Top 10**
  - https://owasp.org/www-project-top-ten/
  - The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications

OWASP Top 10 Web Application Security Risks

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery



2017

A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

2021

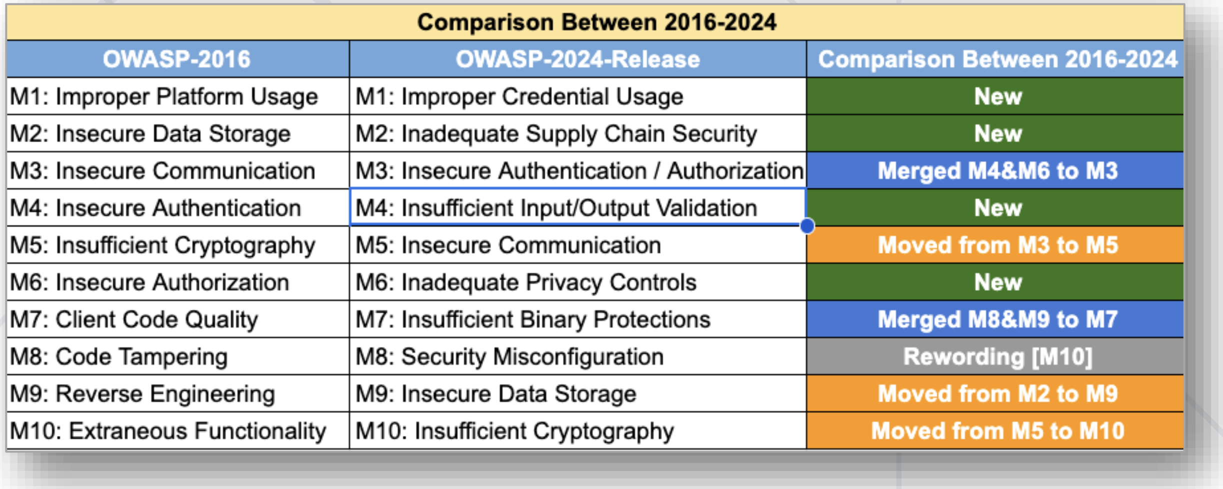
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

* Top 10 Web Application Security Risks for 2024 is yet to come

OWASP Top 10 Mobile Security Risks

- M1: Improper Credential Usage
- M2: Inadequate Supply Chain Security
- M3: Insecure Authentication/Authorization
- M4: Insufficient Input/Output Validation
- M5: Insecure Communication
- M6: Inadequate Privacy Controls
- M7: Insufficient Binary Protections
- M8: Security Misconfiguration
- M9: Insecure Data Storage
- M10: Insufficient Cryptography



Comparison Between 2016-2024

OWASP-2016	OWASP-2024-Release	Comparison Between 2016-2024
M1: Improper Platform Usage	M1: Improper Credential Usage	New
M2: Insecure Data Storage	M2: Inadequate Supply Chain Security	New
M3: Insecure Communication	M3: Insecure Authentication / Authorization	Merged M4&M6 to M3
M4: Insecure Authentication	M4: Insufficient Input/Output Validation	New
M5: Insufficient Cryptography	M5: Insecure Communication	Moved from M3 to M5
M6: Insecure Authorization	M6: Inadequate Privacy Controls	New
M7: Client Code Quality	M7: Insufficient Binary Protections	Merged M8&M9 to M7
M8: Code Tampering	M8: Security Misconfiguration	Rewording [M10]
M9: Reverse Engineering	M9: Insecure Data Storage	Moved from M2 to M9
M10: Extraneous Functionality	M10: Insufficient Cryptography	Moved from M5 to M10

OWASP Top 10 API Security Risks

- API1:2023 - Broken Object Level Authorization
- API2:2023 - Broken Authentication
- API3:2023 - Broken Object Property Level Authorization
- API4:2023 - Unrestricted Resource Consumption
- API5:2023 - Broken Function Level Authorization
- API6:2023 - Unrestricted Access to Sensitive Business Flows
- API7:2023 - Server Side Request Forgery
- API8:2023 - Security Misconfiguration
- API9:2023 - Improper Inventory Management
- API10:2023 - Unsafe Consumption of APIs

Common Security Attacks

Understanding Key Threats in Cybersecurity

Common Security Attacks

- **SQL injection**
  - SQL code is injected into an application's database query, allowing access, modification, data deletion, or control of the database
  - **Example:** An attacker inputs malicious SQL statements into a login form to bypass authentication
- **OS command injection**
  - Malicious system-level commands are injected into input fields or URLs, allowing execution of arbitrary code, sensitive data access, and taking control of the entire system
  - **Example:** An attacker exploits a web application to execute system commands on the server

Common Security Attacks

- Cross-Site Scripting (XSS)
  - Executable scripts are injected into the code of a trusted application or website
  - **Example:** An attacker injects a script that runs in other users' browsers to steal their session cookies
- Cross-Site Request Forgery (CSRF)
  - Tricking a user into unknowingly performing an action on a web application by leveraging the user's existing session or login credentials
  - **Example:** A malicious email containing a link that performs an unwanted action when clicked by an authenticated user

Common Security Attacks

- **Unrestricted upload of dangerous file**
  - Allowing users to upload files without proper validation, which can lead to execution of malicious code
  - **Example:** An attacker uploads a script disguised as an image file, which gets executed on the server
- **URL redirection to untrusted site** (Open Redirect)
  - Redirecting users to untrusted websites, potentially leading to phishing attacks
  - **Example:** A user clicks a link that redirects them to a malicious website designed to steal their credentials

Common Security Attacks

- **Buffer overflow**
  - Program attempts to store more data in a buffer than it is designed to hold, resulting in overflow of data into adjacent memory locations, causing the program to crash or behave unpredictably
  - **Example:** An attacker sends oversized input to a program to overwrite memory and execute arbitrary code
- **Improper limitation of a pathname**
  - Failure to properly restrict access to files and directories based on their pathname
  - **Example:** An attacker accesses sensitive files by exploiting path traversal vulnerabilities

Common Security Attacks

- **Download of a code without integrity check**
  - Downloading code without verifying its integrity, potentially allowing execution of malicious code
  - **Example:** A web application downloads and executes an update without verifying its authenticity, leading to a compromise
- **Uncontrolled Format String**
  - Exploiting format string vulnerabilities to execute arbitrary code
  - **Example:** An attacker uses format specifiers to manipulate program output and gain control of the system

Common Security Attacks

- **Missing or Incorrect Authorization**
  - Failure to properly check if a user is authorized to perform an action
  - **Example:** Users gaining access to administrative functions without proper authorization checks
- **Use of Hard-Coded Credentials**
  - Including hard-coded usernames and passwords in the code, leading to easy exploitation
  - **Example:** An attacker finds hard-coded credentials in the source code and uses them to access the system

Common Security Attacks

- **Missing Encryption of Sensitive Data**
  - Failure to encrypt sensitive data, making it accessible to unauthorized users
  - **Example**: Sensitive user information stored in plaintext and accessed by attackers
- **Execution of Unnecessary Privileges**
  - Running processes with higher privileges than necessary, increasing the risk of exploitation
  - **Example**: A web server running with administrative privileges is compromised and used to control the entire system

Common Security Attacks

- Improper Restriction of Excessive Authentication Attempts
  - Not limiting the number of authentication attempts, allowing brute-force attacks
  - **Example**: An attacker repeatedly attempts to guess a user's password without being locked out
- Failure to Rotate Logs
  - Not regularly archiving or rotating log files, leading to potential data loss or performance issues
  - **Example**: An attacker floods the system with requests, causing log files to grow excessively large and potentially overwrite critical log data or degrade system performance

Security Tools

Different types of tools

Security Testing Tools Types

- **Static**
  - Scans the **source code** of an application **without executing** it
  - Detects potential security issues **early** in the **development cycle**
- **Dynamic**
  - Scans an application **while it's running**
  - Simulates actions / generate input to trigger security vulnerabilities
- **Interactive**
  - Combines **both static** and **dynamic** analysis
  - Analyzes the **source code** and scans the **app** while it's **running**

Security Testing Tools Types

- **Cloud-based**
  - Hosted on **remote servers**
  - Software as a Service (**SaaS**)
  - **Easy** to **deploy**
  - **No infrastructure maintenance**
  - **Scales up** or **down quickly** to meet testing needs
  - Ideal for organizations that don't have the resources to manage and maintain their own infrastructure

Security Testing Tools Types

- **On-premise**
  - Installed and managed on **local servers** or infrastructure
  - More **customizable** and **flexible**
  - Requires **more resources** and **maintenance**
  - May **not scale as easily** as cloud-based solutions
  - Ideal for organizations that have **strict security** and compliance requirements

Popular Security Testing Tools

- Veracode - interactive, cloud-based
- IBM Application Security on Cloud - interactive, cloud-based
- Burp Suite - dynamic, on-premise
- Checkmarx - static, on-premise
- **OWASP ZAP - dynamic, on-premise**
- Invicti - dynamic, cloud-based
- HP Fortify - static, on-premise
- SonarQube - static, on-premise
- HCL AppScan - static, on-premise
- FindBugs - static, on-premise

OWASP ZAP

Comprehensive Web Application Security Testing

OWASP ZAP Overview

- **Full Name**: Zed Attack Proxy
- **Developed by**: OWASP (Open Web Application Security Project)
- **Purpose**: To find security vulnerabilities in web applications
- **Key Features**:
  - Automated scanners and various tools for manual testing
  - Easy to use for beginners while providing powerful capabilities for professionals
  - Supports the latest and most common security vulnerabilities and standards

Key Features of OWASP ZAP

- **Automated Scanning**
  - Quickly identify potential vulnerabilities
  - Specify URL to attack
  - Choose between traditional or Ajax spiders
- **Manual Testing Tools**
  - Set of tools for more experienced testers
  - Includes proxy, spider, fuzzer, WebSocket support, and scripting environment

Key Features of OWASP ZAP

- **Plug-n-Hack Support**
  - Easy integration with browser plugins
  - Enhances testing capabilities
- **Dynamic Application Security Testing (DAST)**
  - Simulates attacks on live applications
  - Effective in discovering security issues

OWASP ZAP Components

- **Spider**: Crawls the web application; Maps out structure and identifies input fields
- **Scanner**: Performs automated scans; Detects vulnerabilities like SQL injection, XSS, and other OWASP Top 10 security risks
- **Fuzzer**: Sends numerous requests with varying inputs; Discovers buffer overflow vulnerabilities, SQL injections, and other input validation issues
- **Session Management**: Manages and manipulates web application sessions for testing
- **API**: REST API for integration with other tools; Automates testing processes

OWASP ZAP Usage Scenarios

- **Development Phase**
  - Integrate ZAP into CI/CD pipeline
  - Automate security testing during development
- **QA Testing**
  - Thorough security testing before release
  - Ensures application security from common vulnerabilities
- **Penetration Testing**
  - Detailed penetration testing by security professionals
  - Combines automated and manual testing techniques

Best Practices for Using OWASP ZAP

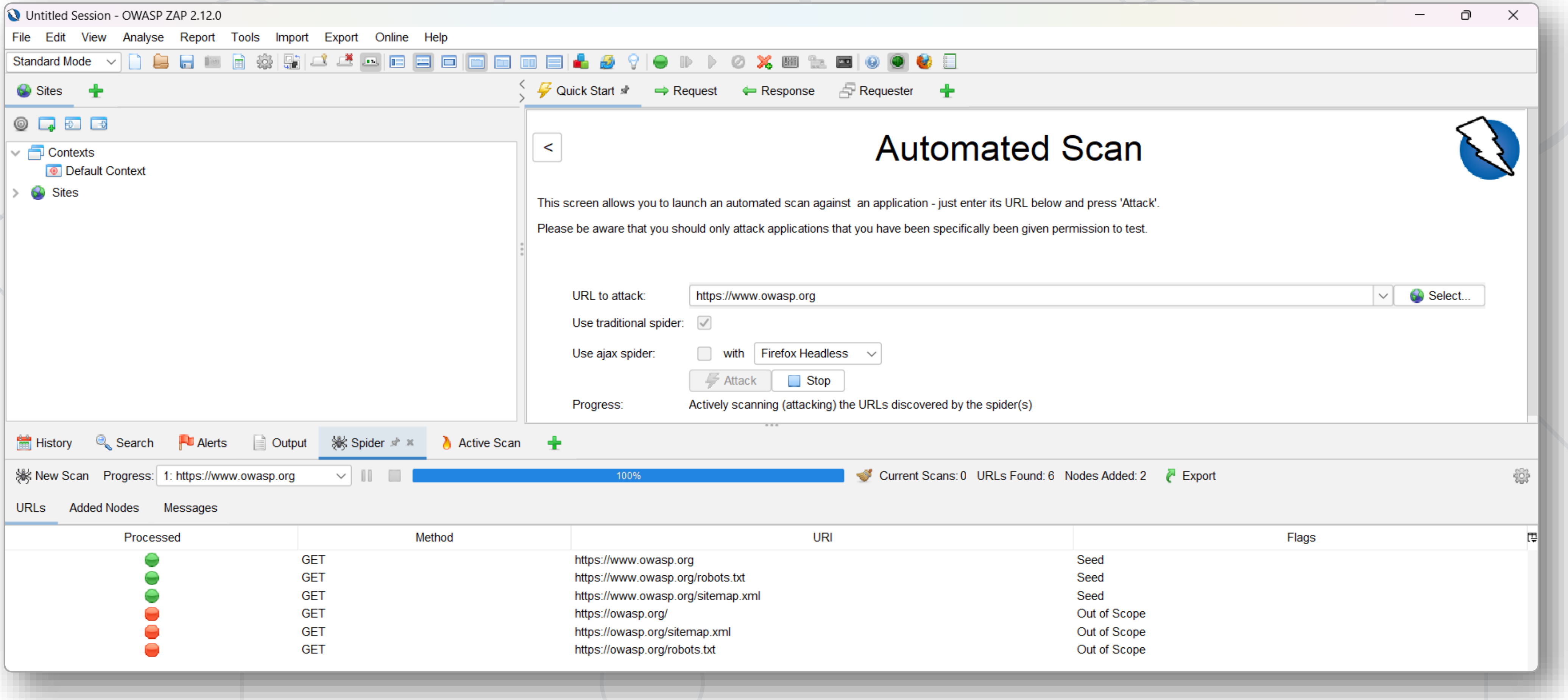
- **Regular Scanning**
  - Schedule regular scans
  - Continuously monitor and secure web applications
- **Combining Automated and Manual Testing**
  - Initial vulnerability assessment with automated scans
  - Follow up with manual testing for deeper analysis
- **Integrating with Development Processes**
  - Incorporate ZAP into development and deployment pipelines
  - Make security testing part of the standard development workflow

OWASP ZAP

Demo

Owasp ZAP: Free Security Testing Tool

- OWASP **ZAP** is free, open-source, powerful tool, written in Java
  - https://www.zaproxy.org/



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: https://www.owasp.org [Select...]

Use traditional spider: ☑

Use ajax spider: ☐ with Firefox Headless

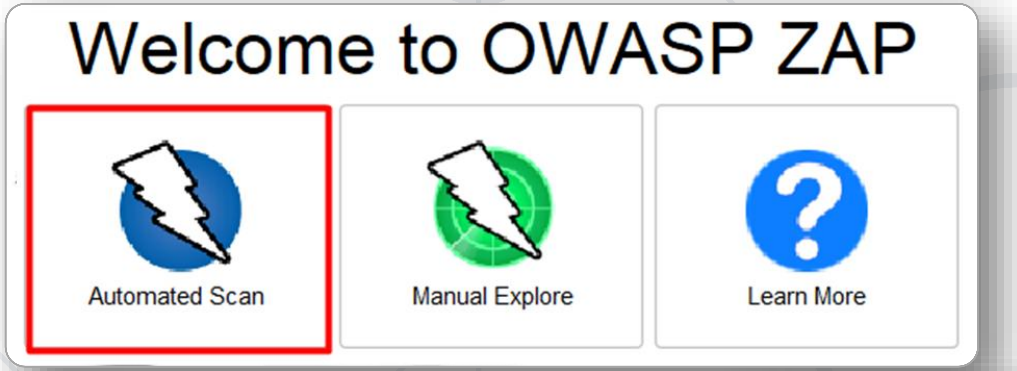
[Attack] [Stop]

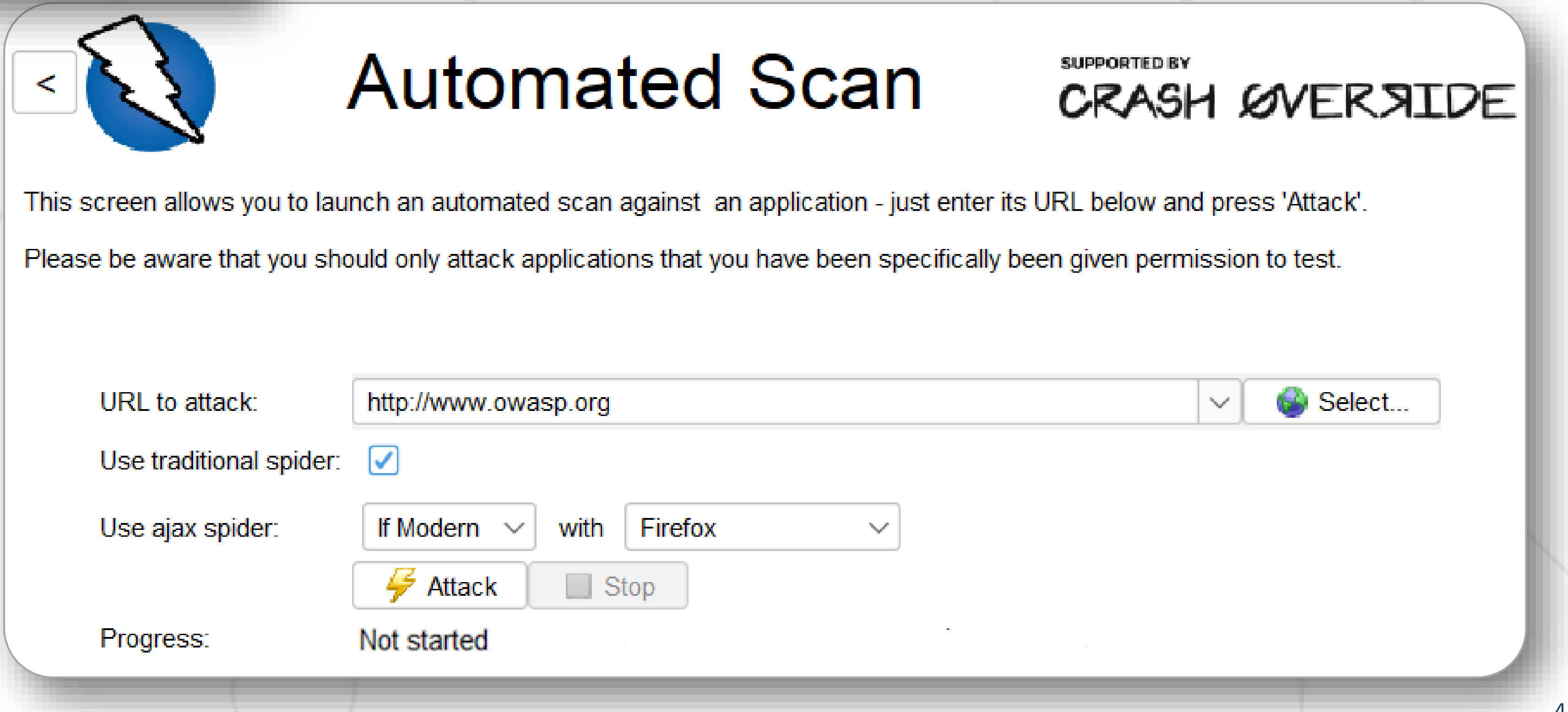
Progress: Actively scanning (attacking) the URLs discovered by the spider(s)

Progress: 1: https://www.owasp.org 100% Current Scans: 0 URLs Found: 6 Nodes Added: 2

Processed	Method	URI	Flags
🟢	GET	https://www.owasp.org	Seed
🟢	GET	https://www.owasp.org/robots.txt	Seed
🟢	GET	https://www.owasp.org/sitemap.xml	Seed
🔴	GET	https://owasp.org/	Out of Scope
🔴	GET	https://owasp.org/sitemap.xml	Out of Scope
🔴	GET	https://owasp.org/robots.txt	Out of Scope

OWASP ZAP Automatic 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.

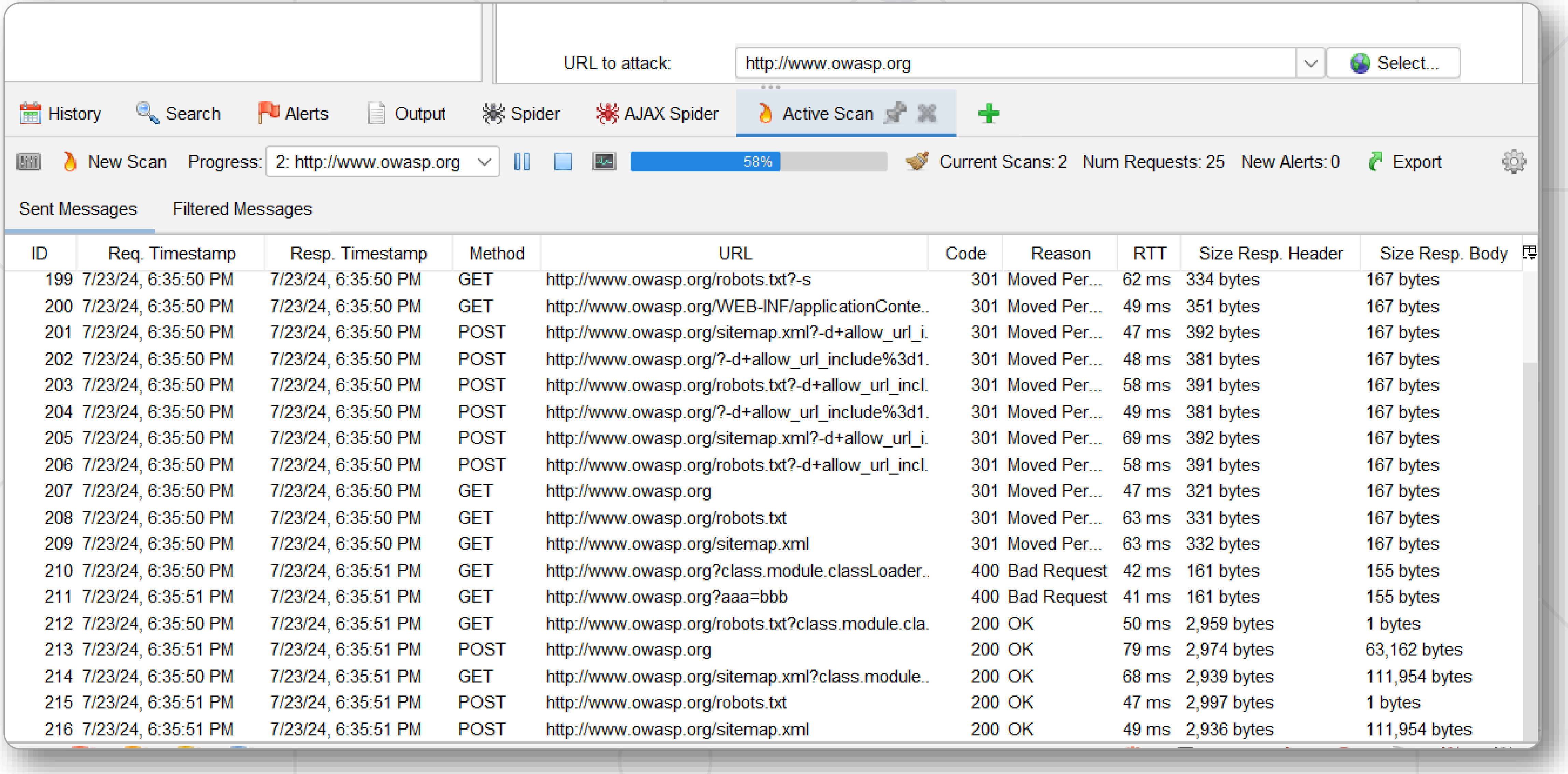
URL to attack: http://www.owasp.org

Use traditional spider: ☑

Use ajax spider: If Modern with Firefox

Progress: Not started

OWASP ZAP Scan in progress



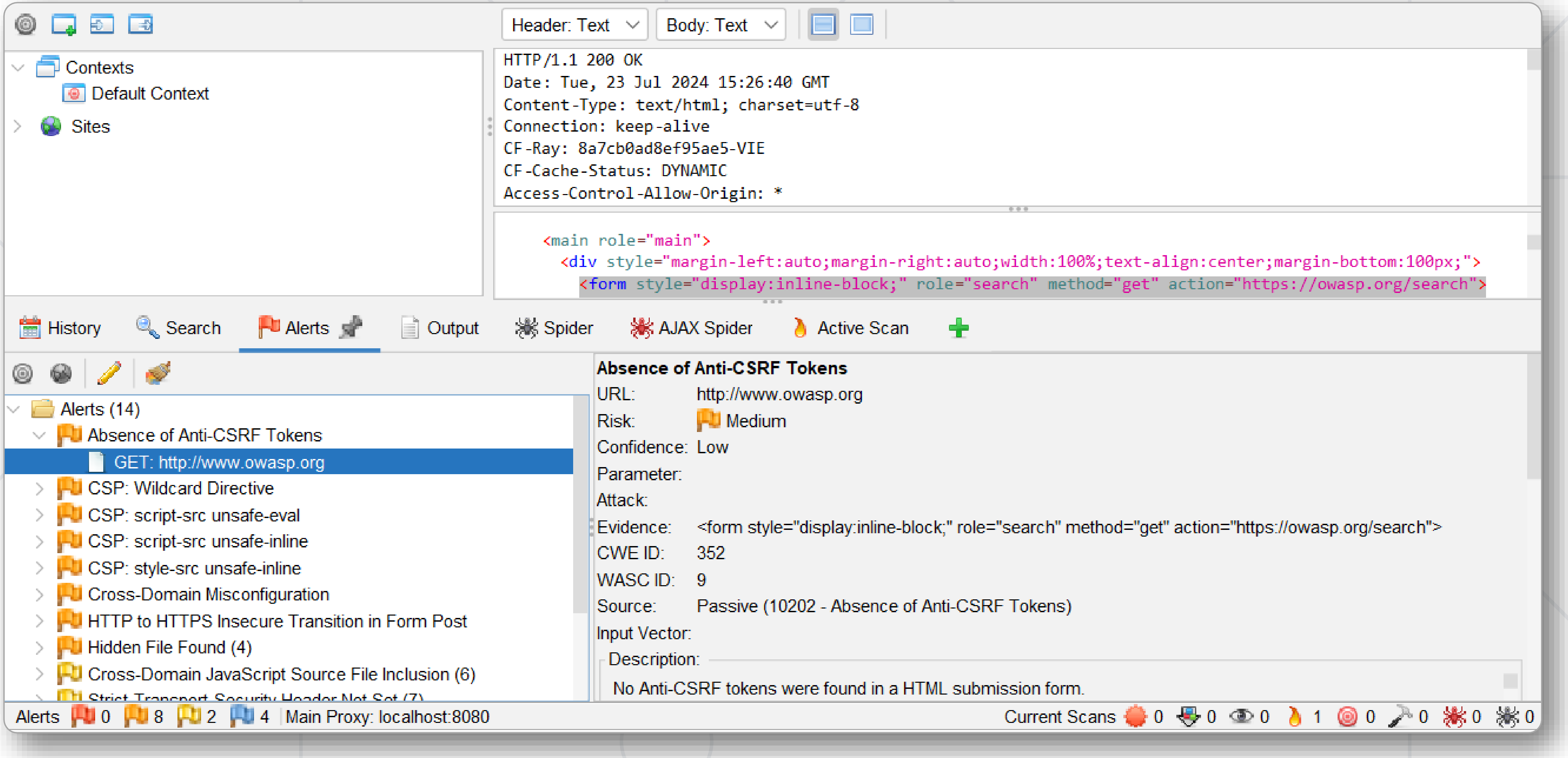
URL to attack: http://www.owasp.org

Progress: 2: http://www.owasp.org 58% Current Scans: 2 Num Requests: 25 New Alerts: 0

Sent Messages Filtered Messages

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
199	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	GET	http://www.owasp.org/robots.txt?-s	301	Moved Per...	62 ms	334 bytes	167 bytes
200	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	GET	http://www.owasp.org/WEB-INF/applicationConte..	301	Moved Per...	49 ms	351 bytes	167 bytes
201	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	POST	http://www.owasp.org/sitemap.xml?-d+allow_url_i.	301	Moved Per...	47 ms	392 bytes	167 bytes
202	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	POST	http://www.owasp.org/?-d+allow_url_include%3d1.	301	Moved Per...	48 ms	381 bytes	167 bytes
203	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	POST	http://www.owasp.org/robots.txt?-d+allow_url_incl.	301	Moved Per...	58 ms	391 bytes	167 bytes
204	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	POST	http://www.owasp.org/?-d+allow_url_include%3d1.	301	Moved Per...	49 ms	381 bytes	167 bytes
205	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	POST	http://www.owasp.org/sitemap.xml?-d+allow_url_i.	301	Moved Per...	69 ms	392 bytes	167 bytes
206	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	POST	http://www.owasp.org/robots.txt?-d+allow_url_incl.	301	Moved Per...	58 ms	391 bytes	167 bytes
207	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	GET	http://www.owasp.org	301	Moved Per...	47 ms	321 bytes	167 bytes
208	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	GET	http://www.owasp.org/robots.txt	301	Moved Per...	63 ms	331 bytes	167 bytes
209	7/23/24, 6:35:50 PM	7/23/24, 6:35:50 PM	GET	http://www.owasp.org/sitemap.xml	301	Moved Per...	63 ms	332 bytes	167 bytes
210	7/23/24, 6:35:50 PM	7/23/24, 6:35:51 PM	GET	http://www.owasp.org?class.module.classLoader..	400	Bad Request	42 ms	161 bytes	155 bytes
211	7/23/24, 6:35:51 PM	7/23/24, 6:35:51 PM	GET	http://www.owasp.org?aaa=bbb	400	Bad Request	41 ms	161 bytes	155 bytes
212	7/23/24, 6:35:50 PM	7/23/24, 6:35:51 PM	GET	http://www.owasp.org/robots.txt?class.module.cla.	200	OK	50 ms	2,959 bytes	1 bytes
213	7/23/24, 6:35:51 PM	7/23/24, 6:35:51 PM	POST	http://www.owasp.org	200	OK	79 ms	2,974 bytes	63,162 bytes
214	7/23/24, 6:35:50 PM	7/23/24, 6:35:51 PM	GET	http://www.owasp.org/sitemap.xml?class.module..	200	OK	68 ms	2,939 bytes	111,954 bytes
215	7/23/24, 6:35:51 PM	7/23/24, 6:35:51 PM	POST	http://www.owasp.org/robots.txt	200	OK	47 ms	2,997 bytes	1 bytes
216	7/23/24, 6:35:51 PM	7/23/24, 6:35:51 PM	POST	http://www.owasp.org/sitemap.xml	200	OK	49 ms	2,936 bytes	111,954 bytes

Alerts



```
HTTP/1.1 200 OK
Date: Tue, 23 Jul 2024 15:26:40 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
CF-Ray: 8a7cb0ad8ef95ae5-VIE
CF-Cache-Status: DYNAMIC
Access-Control-Allow-Origin: *
```

```
  <main role="main">
   <div style="margin-left:auto;margin-right:auto;width:100%;text-align:center;margin-bottom:100px;">
    <form style="display:inline-block;" role="search" method="get" action="https://owasp.org/search">
```

- Alerts (14)
  - Absence of Anti-CSRF Tokens
    - GET: http://www.owasp.org
  - CSP: Wildcard Directive
  - CSP: script-src unsafe-eval
  - CSP: script-src unsafe-inline
  - CSP: style-src unsafe-inline
  - Cross-Domain Misconfiguration
  - HTTP to HTTPS Insecure Transition in Form Post
  - Hidden File Found (4)
  - Cross-Domain JavaScript Source File Inclusion (6)
  - Strict Transport Security Header Not Set (7)

Absence of Anti-CSRF Tokens

URL: http://www.owasp.org
Risk: Medium
Confidence: Low
Parameter:
Attack:
Evidence: <form style="display:inline-block;" role="search" method="get" action="https://owasp.org/search">
CWE ID: 352
WASC ID: 9
Source: Passive (10202 - Absence of Anti-CSRF Tokens)
Input Vector:

Description:

No Anti-CSRF tokens were found in a HTML submission form.

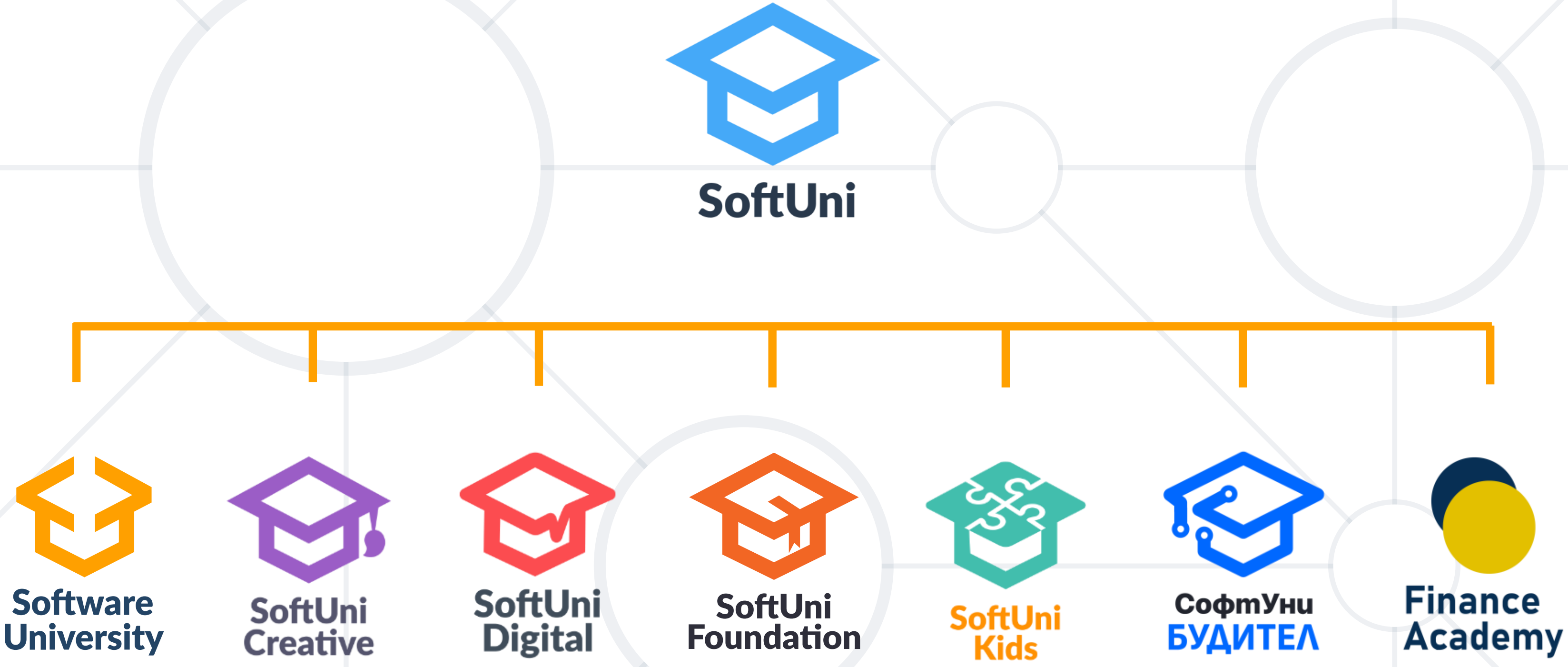
Alerts 0 8 2 4 Main Proxy: localhost:8080 Current Scans 0 0 0 1 0 0 0 0

Summary

- Security Testing
  - Key elements
  - Types – static, dynamic, interactive
  - Principles
- Different Security Attacks Explained
- OWASP ZAP
  - Mature, powerful, open-source tool

Software University

Questions?


SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

Diamond Partners

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg