



OWASP®

# Web Security Testing Guide



Deborah Binyanya | binyanyadebby@gmail.com

## **Lab - Use the OWASP Web Security Testing Guide**

# **Objectives**

In this lab, you will complete the following objectives:

- Part 1: Investigate the WSTG
- Part 2: Scan a Website and Investigate Vulnerability References

# **Background / Scenario**

The Open Worldwide Application Security Project (**OWASP**) nonprofit foundation developed the Web Security Testing Guide (WSTG) to test the most common web application security issues. The guide is useful for various stakeholders such as developers, software testers, security specialists, and project managers. The OWASP Web Security Testing Guide is a free tool that is available to organizations and individuals.

The testing guide is also a useful tool for ethical hacking. Ethical hackers can use the guide to test their clients' running web applications for common security vulnerabilities.

In this lab, you will review the WSTG and then scan a web application for vulnerabilities using the **OWASP Zed Attack Proxy** (ZAP). You will investigate some of the vulnerabilities that were discovered and reference one back to the WSTG.

# **Required Resources**

- Kali VM customized for the Ethical Hacker course
- Internet access

# **Instructions**

## **Part 1: Investigate the WSTG**

### **Step 1: Explore the OWASP WSTG Project Site.**

- a. Navigate to the OWSAP Web Security Testing Guide site at <https://owasp.org/www-project-web-security-testing-guide/>.
- b. Review the information on the main page.

The screenshot shows a web browser window titled "Ethical-Hacker-Kali [Running] - Oracle VirtualBox". The address bar displays the URL <https://owasp.org/www-project-web-security-testing-guide/>. The page content is the OWASP Web Security Testing Guide. At the top, there is a navigation bar with links for "PROJECTS", "CHAPTERS", "EVENTS", "ABOUT", and a search bar. Below the navigation bar, there are social media links for GitHub, LinkedIn, and Twitter, along with a "Follow @owasp\_wstg" button. A sidebar on the right contains sections for "The OWASP® Foundation", "Project Classification", "Project Links", "Getting Involved", and "Social". The main content area describes the WSTG Project and its purpose, provides links for "Main", "Release Versions", and "FAQ", and includes sections for "Contributions", "Stable", "Latest", and "Content".

What is the purpose of the OSWASP Web Security Testing Guide?

**It is designed to guide the testing of security of web applications. It describes techniques, methods, tools, and resources for testing for the most common web application security issues. It provides a framework of best practices.**

Click the **Release Versions** tab. What is the current release version of the guide? **Version 4.2**

The screenshot shows the same web browser window as before, but the "Release Versions" tab is now active. The main content area displays the following information:

- Stable**: View the always-current stable version at [stable](#).
- [Unreleased 4.3]**
- [Version 4.2] - 2020-12-03**:  
Version 4.2 introduces new testing scenarios, updates existing chapters, and offers an improved writing style and chapter layout.  
[Download the v4.2 PDF here.](#)
- [Version 4.1] - 2020-04-21**:  
Version 4.1 serves as a post-migration stable version under the new GitHub repository workflow.  
[Download the v4.1 PDF here.](#)
- [Version 4.0] - 2014-09-17**:  
[Download the v4 PDF here.](#)  
A printed book is also made [available for purchase](#).
- [Version 3.0] - 2008-12-16**:  
[Download the v3 PDF here.](#)

Click the most recent released version and review the Table of Contents.

The OWASP® Foundation works to improve the security of software through its community-led open source software projects, hundreds of chapters worldwide, tens of thousands of members, and by hosting local and global conferences.

**WSTG Contents (v4.2)**

- 0. Foreword by Eoin Keary
- 1. Frontispiece
- 2. Introduction
- 2.1 The OWASP Testing Project
- 2.2 Principles of Testing
- 2.3 Testing Techniques Explained
- 2.4 Manual Inspections and Reviews
- 2.5 Threat Modeling
- 2.6 Source Code Review
- 2.7 Penetration Testing
- 2.8 The Need for a Balanced Approach

## Step 2: Review the content.

Click and review the **Foreword** by Eoin Keary in the Table of Contents. According to Mr. Keary, who is initially responsibility for application security? **developers**

world. There are many different ways to test for security flaws and this guide captures the consensus of the leading experts on how to perform this testing quickly, accurately, and efficiently. OWASP gives like minded security folks the ability to work together and form a leading practice approach to a security problem.

The importance of having this guide available in a completely free and open way is important for the foundation's mission. It gives anyone the ability to understand the techniques used to test for common security issues. Security should not be a black art or closed secret that only a few can practice. It should be open to all and not exclusive to security practitioners but also QA, Developers and Technical Managers. The project to build this guide keeps this expertise in the hands of the people who need it - you, me and anyone that is involved in building software.

This guide must make its way into the hands of developers and software testers. There are not nearly enough application security experts in the world to make any significant dent in the overall problem. **The initial responsibility for application security must fall on the shoulders of the developers because they write the code.** It shouldn't be a surprise that developers aren't producing secure code if they're not testing for it or consider the types of bugs which introduce vulnerability.

Keeping this information up to date is a critical aspect of this guide project. By adopting the wiki approach, the OWASP community can evolve and expand the information in this guide to keep pace with the fast moving application security threat landscape.

This Guide is a great testament to the passion and energy our members and project volunteers have for this subject. It shall certainly help to change the world a line of code at a time.

**Tailoring and Prioritizing**

You should adopt this guide in your organization. You may need to tailor the information to match your organization's technologies, processes, and organizational structure.

In general there are several different roles within organizations that may use this guide:

- Developers should use this guide to ensure that they are producing secure code. These tests should be a part of normal code and unit testing procedures.
- Software testers and QA should use this guide to expand the set of test cases they apply to applications. Catching these vulnerabilities early saves considerable time and effort later.

3.0 Penetration Testing Methodologies  
 4. Web Application Security Testing  
 4.0 Introduction and Objectives  
 4.1 Information Gathering  
 4.1.1 Conduct Search Engine Discovery  
 Reconnaissance for Information Leakage  
 4.1.2 Fingerprint Web Server  
 4.1.3 Review Webserver Metatiles for Information Leakage  
 4.1.4 Enumerate Applications on Webserver  
 4.1.5 Review Webpage Content for Information Leakage  
 4.1.6 Identify Application Entry Points  
 4.1.7 Map Execution Paths Through Application  
 4.1.8 Fingerprint Web Application Framework  
 4.1.9 Fingerprint Web Application  
 4.1.10 Map Application Architecture  
 4.2 Configuration and Deployment Management Testing  
 4.2.1 Test Network Infrastructure Configuration  
 4.2.2 Test Application Platform Configuration  
 4.2.3 Test File Extensions Handling for Sensitive Information  
 4.2.4 Review Old Backup and Unreferenced Files for Sensitive Information  
 4.2.5 Enumerate Infrastructure and Application Admin Interfaces  
 4.2.6 Test HTTP Methods  
 4.2.7 Test HTTP Strict Transport Security  
 4.2.8 Test RIA Cross Domain Policy  
 4.2.9 Test File Permission  
 4.2.10 Test for Subdomain Takeover  
 4.2.11 Test Cloud Storage  
 4.3 Identity Management Testing

Return to the **Table of Contents** and select **Introduction**.

Security testing should be included in which phase(s) of the Software Development Lifecycle (SDLC)?  
**All phases**

Most people today don't test software until it has already been created and is in the deployment phase of its life cycle (i.e., code has been created and instantiated into a working web application). This is generally a very ineffective and cost-prohibitive practice. One of the best methods to prevent security bugs from appearing in production applications is to improve the Software Development Life Cycle (SDLC) by including security in each of its phases. An SDLC is a structure imposed on the development of software artifacts. If an SDLC is not currently being used in your environment, it is time to pick one! The following figure shows a generic SDLC model as well as the (estimated) increasing cost of fixing security bugs in such a model.

**Alternative Channel**  
 4.5 Authorization Testing  
 4.5.1 Testing Directory Traversal File Include  
 4.5.2 Testing for Bypassing Authorization Schema  
 4.5.3 Testing for Privilege Escalation  
 4.5.4 Testing for Insecure Direct Object References  
 4.6 Session Management Testing  
 4.6.1 Testing for Session Management Schema  
 4.6.2 Testing for Cookies Attributes  
 4.6.3 Testing for Session Fixation  
 4.6.4 Testing for Exposed Session Variables  
 4.6.5 Testing for Cross Site Request Forgery  
 4.6.6 Testing for Logout Functionality  
 4.6.7 Testing Session Timeout  
 4.6.8 Testing for Session Puzzling  
 4.6.9 Testing for Session Hijacking  
 4.7 Input Validation Testing  
 4.7.1 Testing for Reflected Cross Site Scripting  
 4.7.2 Testing for Stored Cross Site Scripting  
 4.7.3 Testing for HTTP Verb Tampering  
 4.7.4 Testing for HTTP Parameter Pollution  
 4.7.5 Testing for SQL Injection  
 4.7.5.1 Testing for Oracle  
 4.7.5.2 Testing for MySQL  
 4.7.5.3 Testing for SQL Server  
 4.7.5.4 Testing PostgreSQL  
 4.7.5.5 Testing for MS Access  
 4.7.5.6 Testing for NoSQL Injection  
 4.7.5.7 Testing for ORM Injection  
 4.7.5.8 Testing for Client-side  
 4.7.6 Testing for LDAP Injection  
 4.7.7 Testing for XML Injection  
 4.7.8 Testing for SSI Injection  
 4.7.9 Testing for XPath Injection  
 4.7.10 Testing for IMAP SMTP Injection

What three factors should be tested in an effective testing program? **People, process, technology.**

**Figure 2-1: Generic SDLC Model**  
 Companies should inspect their overall SDLC to ensure that security is an integral part of the development process. SDLCs should include security tests to ensure security is adequately covered and controls are effective throughout the development process.

**What to Test?**  
 It can be helpful to think of software development as a combination of people, process, and technology. If these are the factors that "create" software, then it is logical that these are the factors that must be tested. Today most people generally test the technology or the software itself.

An effective testing program should have components that test the following:

- **People** – to ensure that there is adequate education and awareness
- **Process** – to ensure that there are adequate policies and standards and that people know how to follow these policies
- **Technology** – to ensure that the process has been effective in its implementation.

Unless a holistic approach is adopted, testing just the technical implementation of an application will not uncover management or operational vulnerabilities that could be present. By testing the people, policies, and processes, an organization can catch issues that would later manifest themselves into defects in the technology, thus eradicating bugs early and identifying the root causes of defects. Likewise, testing only some of the technical issues that can be present in a system will result in an incomplete and inaccurate security posture assessment.

Denis Verdon, Head of Information Security at [Fidelity National Financial](#), presented an excellent analogy for this misconception at the OWASP AppSec 2004 Conference in New York:

*If cars were built like applications ... safety tests would assume frontal impact only. Cars would not be roll tested, or tested for stability in emergency maneuvers, brake effectiveness, side impact, and resistance to theft.*

**How To Reference WSTG Scenarios**

4.7.10 Testing for Fuzzer Incoming Requests  
 4.7.17 Testing for Host Header Injection  
 4.7.18 Testing for Server-side Template Injection  
 4.7.19 Testing for Server-side Request Forgery  
 4.8 Testing for Error Handling  
 4.8.1 Testing for Improper Error Handling  
 4.8.2 Testing for Stack Traces  
 4.9 Testing for Weak Cryptography  
 4.9.1 Testing for Weak Transport Layer Security  
 4.9.2 Testing for Padding Oracle  
 4.9.3 Testing for Sensitive Information Sent via Unencrypted Channels  
 4.9.4 Testing for Weak Encryption  
 4.10 Business Logic Testing  
 4.10.0 Introduction to Business Logic  
 4.10.1 Test Business Logic Data Validation  
 4.10.2 Test Ability to Forge Requests  
 4.10.3 Test Integrity Checks  
 4.10.4 Test for Process Timing  
 4.10.5 Test Number of Times a Function Can Be Used Limits  
 4.10.6 Testing for the Circumvention of Work Flows  
 4.10.7 Test Defenses Against Application Misuse  
 4.10.8 Test Upload of Unexpected File Types  
 4.10.9 Test Upload of Malicious Files  
 4.11 Client-side Testing  
 4.11.1 Testing for DOM-Based Cross Site Scripting  
 4.11.2 Testing for JavaScript Execution  
 4.11.3 Testing for HTML Injection  
 4.11.4 Testing for Client-side URL Redirect  
 4.11.5 Testing for CSS Injection  
 4.11.6 Testing for Client-side Resource Manipulation  
 4.11.7 Testing Cross Origin Resource Sharing

What four testing techniques are presented in the introduction of the OWASP Web Testing Guide?  
 Complete the table with the techniques and their definitions. Define the techniques in your own words.

The screenshot shows a Kali Linux desktop environment with a browser window open to the OWASP Web Security Testing Guide. The browser tabs include 'Vulnerability: SQL Injectio...', 'CrackStation - Online Pa...', 'What is SQL Injection [SC...', 'WSTG - v4.2 | OWASP Fo...', and the current tab, 'https://owasp.org/www-project-web-security-testing-guide/v42/2-Introduction/'. The main content of the page discusses testing techniques, specifically manual inspections and reviews, threat modeling, code review, and penetration testing.

recommen...  
allow another security tester to reproduce the results. Writing the report should not be overly burdensome on the security tester themselves. Security testers are not generally renowned for their creative writing skills, and agreeing on a complex report can lead to instances where test results are not properly documented. Using a security test report template can save time and ensure that results are documented accurately and consistently, and are in a format that is suitable for the audience.

**Testing Techniques Explained**

This section presents a high-level overview of various testing techniques that can be employed when building a testing program. It does not present specific methodologies for these techniques, as this information is covered in Chapter 3. This section is included to provide context for the framework presented in the next chapter and to highlight the advantages or disadvantages of some of the techniques that should be considered. In particular, we will cover:

- Manual Inspections & Reviews
- Threat Modeling
- Code Review
- Penetration Testing

**Manual Inspections and Reviews**

**Overview**

Manual Inspections are human reviews that typically test the security implications of people, policies, and processes. Manual Inspections can also include inspection of technology decisions such as architectural designs. They are usually conducted by analyzing documentation or performing interviews with the designers or system owners.

While the concept of manual inspections and human reviews is simple, they can be among the most powerful and effective techniques available. By asking someone how something works and why it was implemented in a specific way, the tester can quickly determine if any security concerns are likely to be evident. Manual inspections and reviews are one of the few ways to test the software development life-cycle process itself and to ensure that there is an adequate policy or skill set in place.

As with many things in life, when conducting manual inspections and reviews it is recommended that a trust-but-verify model is adopted. Not everything that the tester is shown or told will be accurate. Manual reviews are particularly good for testing whether people understand the security process, have been made aware of

## Techniques

### Manual Inspections & Reviews

## Definition

This involves people manually checking systems, designs, configurations, or processes to identify security weaknesses, mistakes, or policy violations without using automated tools.

### Threat Modeling

Threat modeling is the process of identifying possible security threats, understanding how an attacker might exploit a system, and determining how to prevent or reduce those risks.

### Code Review

Code review is the practice of examining source code to find security flaws, bugs, or poor coding practices that could be exploited by attackers.

### Penetration Testing

Penetration testing is a controlled attempt to exploit vulnerabilities in a system to evaluate how secure it is and to understand the potential impact of real-world attacks.

Return to the **Table of Contents** and select **OWASP Testing Framework**. According to the framework, what are the phases in which testing activities should take place?

You're viewing the current stable version of the Web Security Testing Guide project.

**WSTG - v4.2**

Home > V42 > 3-The OWASP Testing Framework

## The OWASP Testing Framework

- 3.1 The Web Security Testing Framework
- 3.2 Phase 1 Before Development Begins
- 3.3 Phase 2 During Definition and Design
- 3.4 Phase 3 During Development
- 3.5 Phase 4 During Deployment
- 3.6 Phase 5 During Maintenance and Operations
- 3.7 A Typical SDLC Testing Workflow
- 3.8 Penetration Testing Methodologies

This website uses cookies to analyze our traffic and only share that information with our analytics partners.

**WSTG Contents (v4.2)**

- 0. Foreword by Eoin Keary
- 1. Frontispiece
- 2. Introduction
- 2.1 The OWASP Testing Project
- 2.2 Principles of Testing
- 2.3 Testing Techniques Explained
- 2.4 Manual Inspections and Reviews
- 2.5 Threat Modeling
- 2.6 Source Code Review
- 2.7 Penetration Testing
- 2.8 The Need for a **Accept** approach
- 2.9 Deriving Security Requirements
- 2.10 Security Tests Integrated in Development

Return to the **Table of Contents** and select **Web Application Security Testing**. What are the 11 categories of active testing described in the guide?

In this case, the application shows two access points (parameters **a** and **b**). All the input points found in this phase represent a target for testing. Keeping track of the directory or call tree of the application and all the access points may be useful during active testing.

### Active Testing

During active testing, a tester begins to use the methodologies described in the follow sections.

The set of active tests have been split into 12 categories:

- Information Gathering
- Configuration and Deployment Management Testing
- Identity Management Testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Input Validation Testing
- Error Handling
- Cryptography
- Business Logic Testing
- Client-side Testing
- API Testing

over an Encrypted Channel

- 4.4.2 Testing for Default Credentials
- 4.4.3 Testing for Weak Lock Out Mechanism
- 4.4.4 Testing for Bypassing Authentication Schema
- 4.4.5 Testing for Vulnerable Remember Password
- 4.4.6 Testing for Browser Cache Weaknesses
- 4.4.7 Testing for Weak Password Policy
- 4.4.8 Testing for Weak Security Question Answer
- 4.4.9 Testing for Weak Password Change or Reset Functionalities
- 4.4.10 Testing for Weaker Authentication in Alternative Channel
- 4.5 Authorization Testing
- 4.5.1 Testing Directory Traversal File Include
- 4.5.2 Testing for Bypassing Authorization Schema
- 4.5.3 Testing for Privilege Escalation
- 4.5.4 Testing for Insecure Direct Object References
- 4.6 Session Management Testing
- 4.6.1 Testing for Session Management Schema

Return to the **Table of Contents** and select **Reporting**. The final report should be targeted to what two groups of stakeholders?

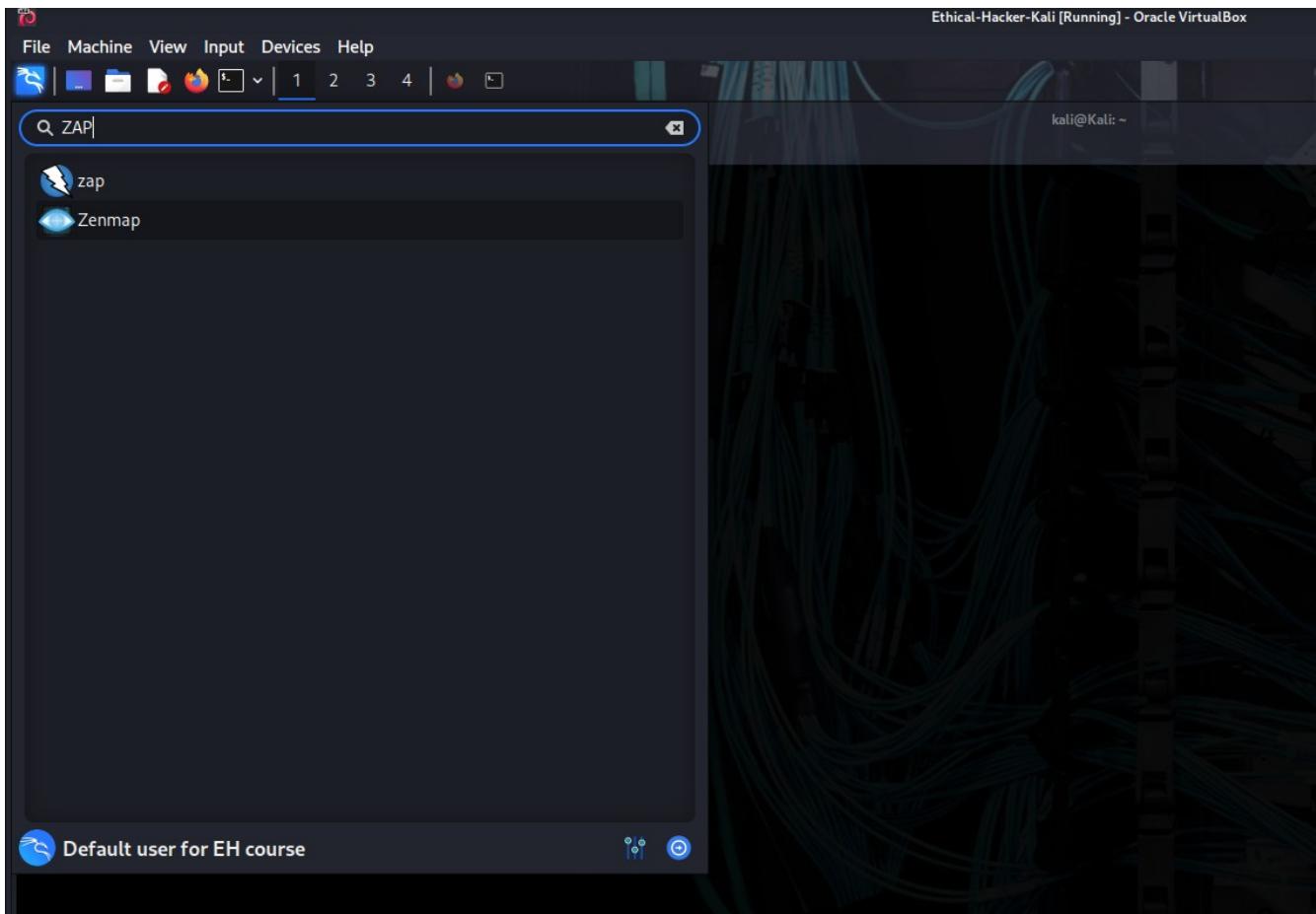
The screenshot shows a Kali Linux VM interface with a browser window open to the OWASP WSTG v4.2 page. The browser tabs include 'Vulnerability: SQL Injectio...', 'CrackStation - Online Pa...', 'What is SQL Injection [SC...', 'WSTG - v4.2 | OWASP Fo...', and 'https://owasp.org/www-project-web-security-testing-guide/v42/5-Reporting/README'. The main content area displays the WSTG v4.2 reporting section, which includes a sub-section titled 'About this Section'. The page also features the OWASP logo and navigation links for 'PROJECTS', 'CHAPTERS', 'EVENTS', 'ABOUT', and a search bar. On the right side of the page, there is a sidebar with information about the OWASP Foundation, a 'WSTG Contents (v4.2)' table of contents, and social media sharing options.

## Part 2: Scan a Website and Investigate Vulnerability References

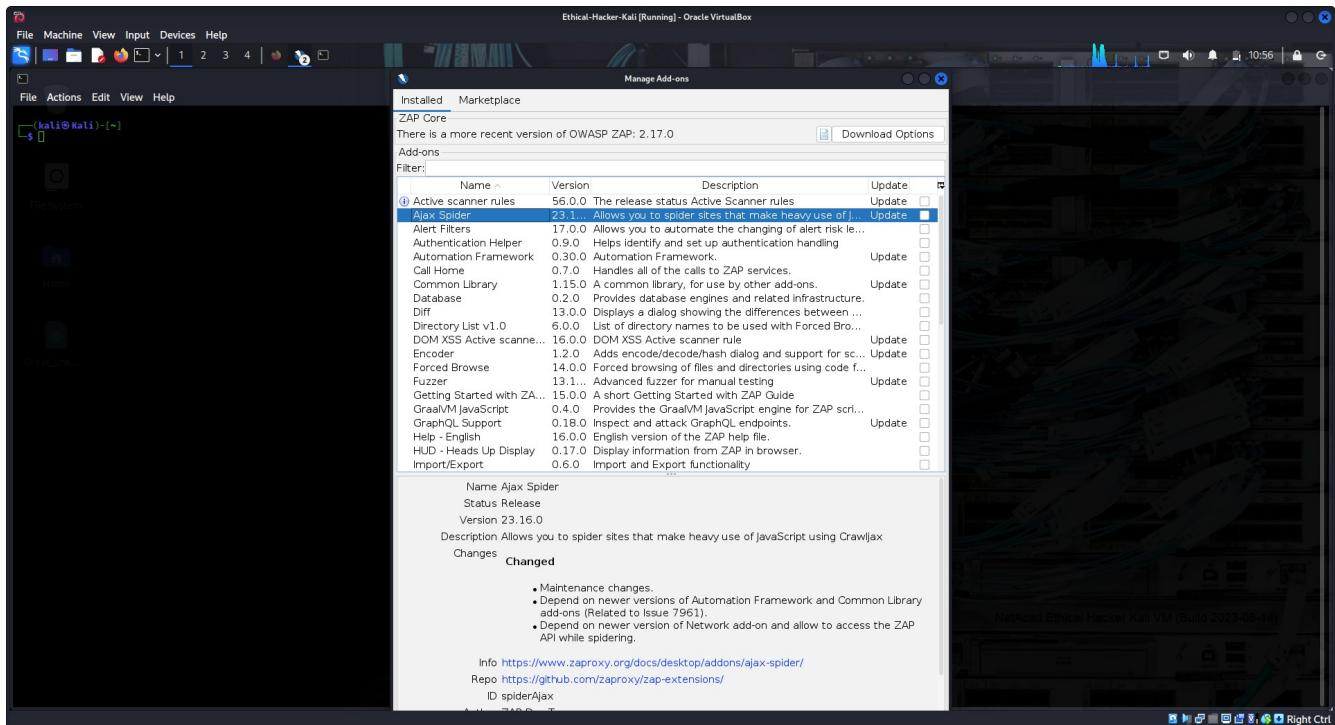
In this part of the lab, you will conduct a vulnerability scan using the Zed Attack Proxy (ZAP). Your target is an intentionally vulnerable website that is available on your VM. You will then use WSTG to learn more about a vulnerability that you discovered.

### Step 1: Open ZAP and start a scanning.

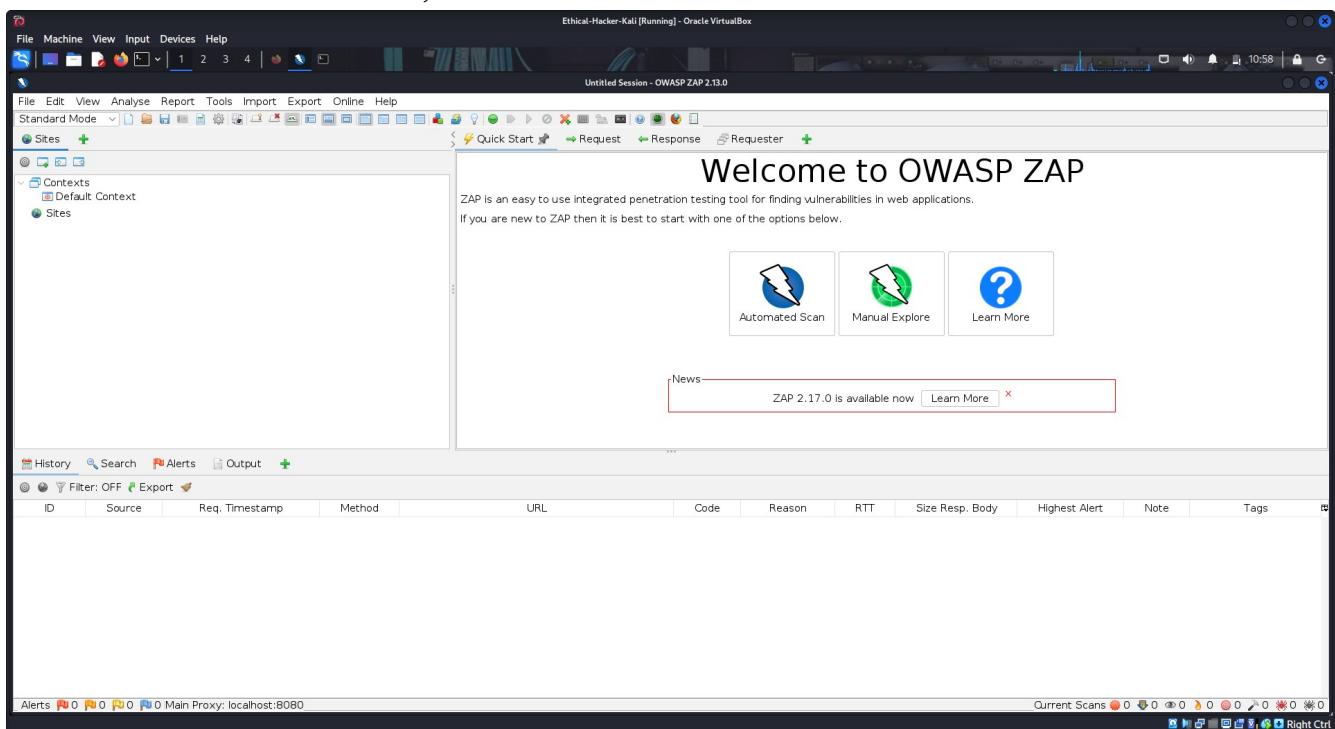
- a. Start the Kali VM as needed. Navigate to the Kali menu. Search for **zap** and start the OWASP Zap scanner.



- b. Click the topmost radio button to persist the session. This means that you can return to the session at a later time.
- c. Close the Manage Add-ons dialog window.

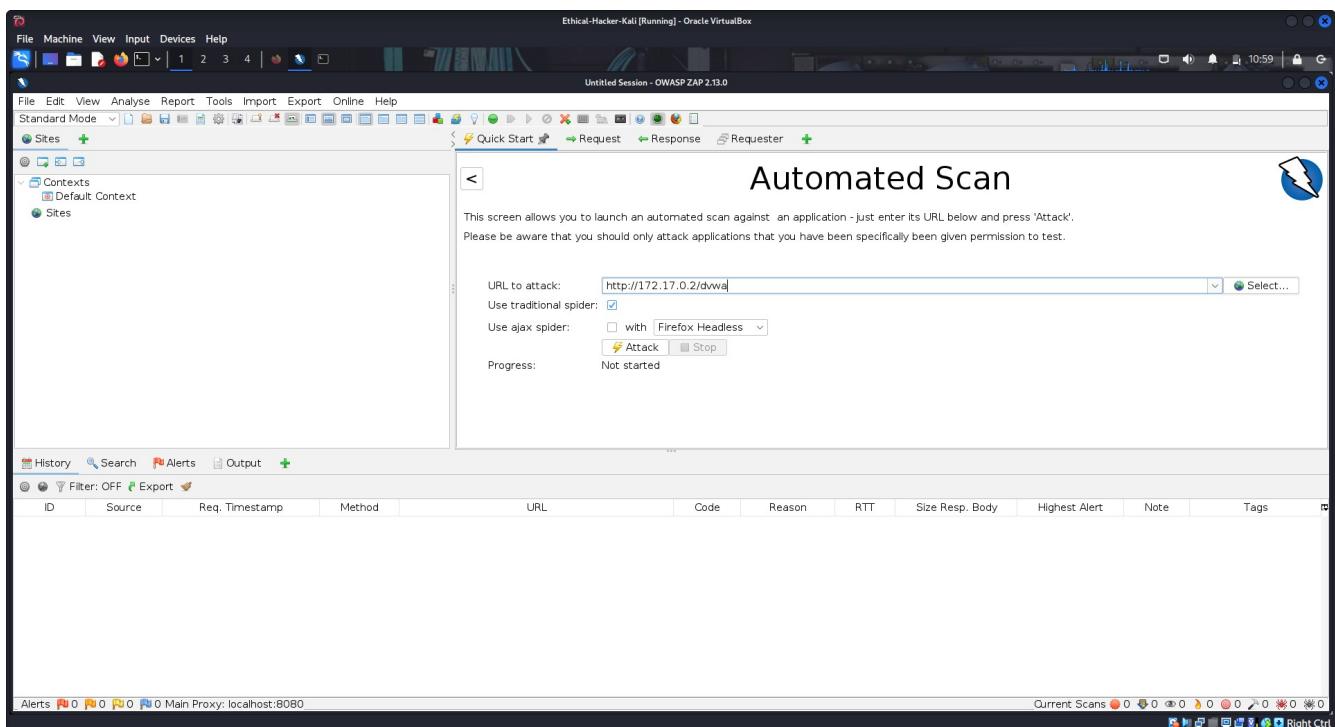


d. In the ZAP main window, click the **Automated Scan** to initiate a scan.

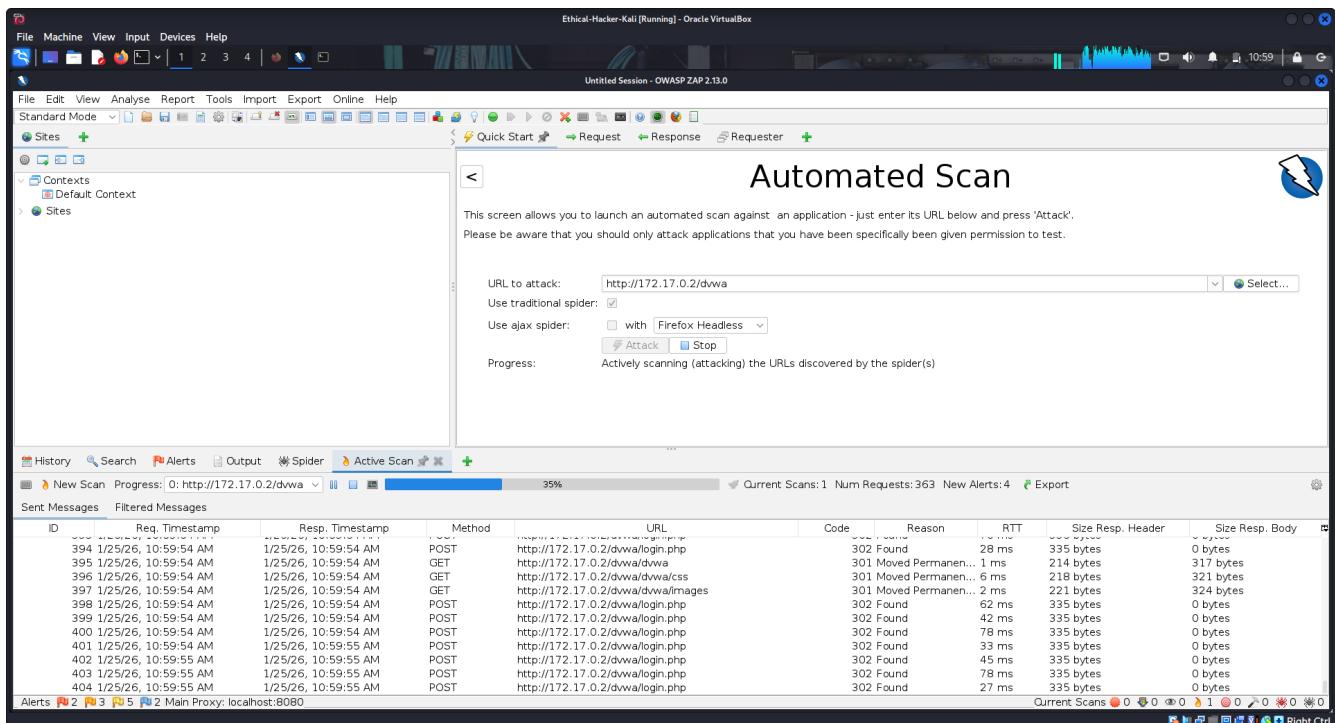


e. In the **URL to Attack** field, enter **172.17.0.2/dvwa**.

f. Click the **Attack** button to begin the scan. The scan should take less than 10 minutes to complete.



First, ZAP uses a web spider to crawl the URL to identify the resources that are available there. It then will apply vulnerability scans to each resource.



Complete scan:

The screenshot shows the OWASP ZAP interface. In the main pane, titled 'Automated Scan', there is a message: '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 given permission to test.' Below this, the 'URL to attack:' field contains 'http://172.17.0.2/dvwa'. Under 'Attack' buttons, 'Attack' is highlighted. The 'Progress:' status says 'Attack complete - see the Alerts tab for details of any issues found'. On the left sidebar, under 'Alerts (15)', the first item is 'Remote Code Execution - CVE-2012-1823 (2)'. The bottom status bar shows 'Alerts 2 5 5 3 Main Proxy: localhost:8080'.

## Step 2: Investigate the results.

Select the **Alerts** tab if it is not already selected. When the scan finishes, you will be automatically switched to there. How many alerts were returned? **15**

Locate and click the **Remote Code Execution – CVE-2012-1823** alert. Scroll through the details of the alert.

The screenshot shows the OWASP ZAP interface with the 'Alerts' tab selected. The 'Alerts (15)' list includes 'Remote Code Execution - CVE-2012-1823 (2)'. Clicking on this alert opens its details. The 'Header' tab shows the following response:

```
HTTP/1.1 200 OK
Date: Sun, 25 Jan 2026 11:32:27 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Type: text/html
content-length: 20
```

The 'Body' tab shows the exploit payload: '9v28xrxgk4sacse4lqmt'. The 'Details' tab displays the following information:

**Remote Code Execution - CVE-2012-1823**

**URL:** http://172.17.0.2/dvwa/login.php?-d+allow\_url\_include%3d1+-d+auto\_prepend\_file%3dphp://input  
**Risk:** High  
**Confidence:** Medium  
**Parameter:**  
**Attack:** <?php exec('echo 9v28xrxgk4sacse4lqmt',\$colm);echo join(" ",\$colm);die();?>  
**Evidence:** 9v28xrxgk4sacse4lqmt  
**CVE ID:** 20  
**WASC ID:** 20  
**Source:** Active (20018 - Remote Code Execution - CVE-2012-1823)  
**Input Vector:**  
**Description:** Some PHP versions, when configured to run using CGI, do not correctly handle query strings that lack an unescaped "=" character, enabling arbitrary code execution. In this case, an operating system command was caused to be executed on the web server, and the results were returned to the web browser.  
**Other Info:**

What is the source of this vulnerability? **Active (20018 - Remote Code Execution - CVE-2012-1823)**

How can this vulnerability be exploited? **Some PHP versions, when configured to run using CGI, do not correctly handle query strings that lack an unescaped "=" character, enabling arbitrary code execution. In this case, an operating system command was caused to be executed on the web server, and the results were returned to the web browser.**

Scroll down to the **Alert Tags** section of the vulnerability. Note the WSTG key and value.

The screenshot shows the OWASP ZAP interface. In the 'Alerts' section, there is one alert for 'Remote Code Execution - CVE-2012-1923'. The alert details pane shows the following information:

- Solution:** Upgrade to the latest stable version of PHP, or use the Apache web server and the mod\_rewrite module to filter out malicious requests using the "RewriteCond" and "RewriteRule" directives.
- Reference:** <http://projects.webappsec.org/improper-input-handling>, <http://cwe.mitre.org/data/definitions/89.html>
- Alert Tags:**

Key	Value
OWASP_2017_A09	<a href="https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities">https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities</a>
WSTG-A2-NPV-12	<a href="https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/4.2-Testing_for_Vulnerable_code_in_Third-party_Libraries#A2-NPV-12">https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/4.2-Testing_for_Vulnerable_code_in_Third-party_Libraries#A2-NPV-12</a>
CVE-2012-1923	<a href="https://nvd.nist.gov/vuln/detail/CVE-2012-1923">https://nvd.nist.gov/vuln/detail/CVE-2012-1923</a>
OWASP_2021_A06	<a href="https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components">https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components</a>

Click the value and use the Ctrl-C keys to copy the URL to the clipboard.

The screenshot shows the OWASP ZAP interface again, focusing on the same alert for 'Remote Code Execution - CVE-2012-1923'. The 'Value' column for the 'WSTG-A2-NPV-12' tag is selected, indicating it is ready to be copied.

- Open a browser and paste the URL in the URL. Navigate to the WSTG site and read about the vulnerability and methods of testing for it. Review this information about the vulnerability to understand what WSTG offers to the penetration tester.

The screenshot shows a Firefox browser window with several tabs open. The active tab is 'WSTG - v4.2 | OWASP Foundation'. The page content is the 'Testing for Command Injection' section of the WSTG. It includes a summary, risks, and a detailed explanation of how OS command injection works. The right sidebar lists the table of contents for WSTG v4.2, which includes chapters on foreword, frontispiece, introduction, the testing project, principles of testing, testing techniques, manual inspections, threat modeling, source code review, and penetration testing.

In what ways can the OWASP Web Security Testing Guide assist organizations to secure their applications?

**Provides a Standardized Security Testing Framework**

**Enables Early Identification of Vulnerabilities**

**Supports Risk-Based Security Assessments**

**Improves Developer and Tester Security Skills**

**Facilitates Compliance and Audit Readiness**

**Enhances Communication Between Technical Teams**

**Supports Both Manual and Automated Testing**

**Encourages Continuous Security Improvement**

When conducting a penetration test of a client's web applications, how could you use the WSTG as a guide? **When conducting a penetration test, the OWASP Web Security Testing Guide (WSTG) can be used as a checklist and roadmap to ensure all key areas of a web application are tested. It helps the tester plan the scope, follow a structured testing approach, perform relevant test cases and report findings using standardized terminology. This ensures the penetration test is thorough, consistent, and aligned with industry best practices.**