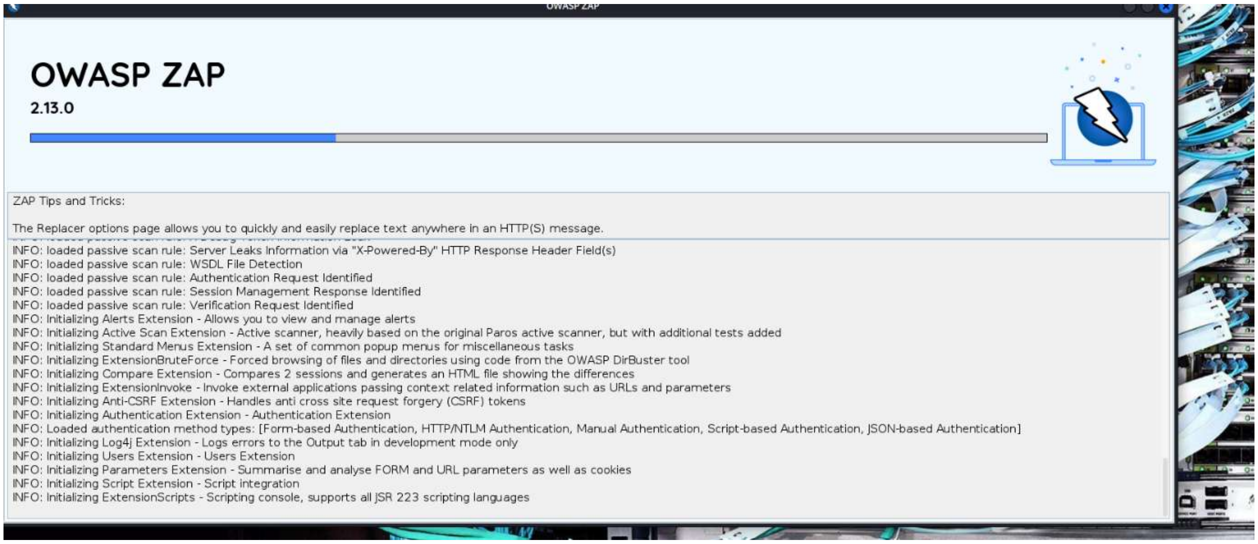
Automated Vulnerability Scanning with OWASP ZAP

Tools: OWASP Zed Attack Proxy (ZAP), OWASP WSTG

Target: `http://172.17.0.2/dvwa`

1. Objective

The objective of this lab is to perform an automated vulnerability scan on a web application to identify security flaws. We will analyze the "Alerts" generated by the tool and use the **OWASP Web Security Testing Guide (WSTG)** to research professional exploitation and remediation methods for a specific Remote Code Execution (RCE) vulnerability.



ZAP Tips and Tricks:

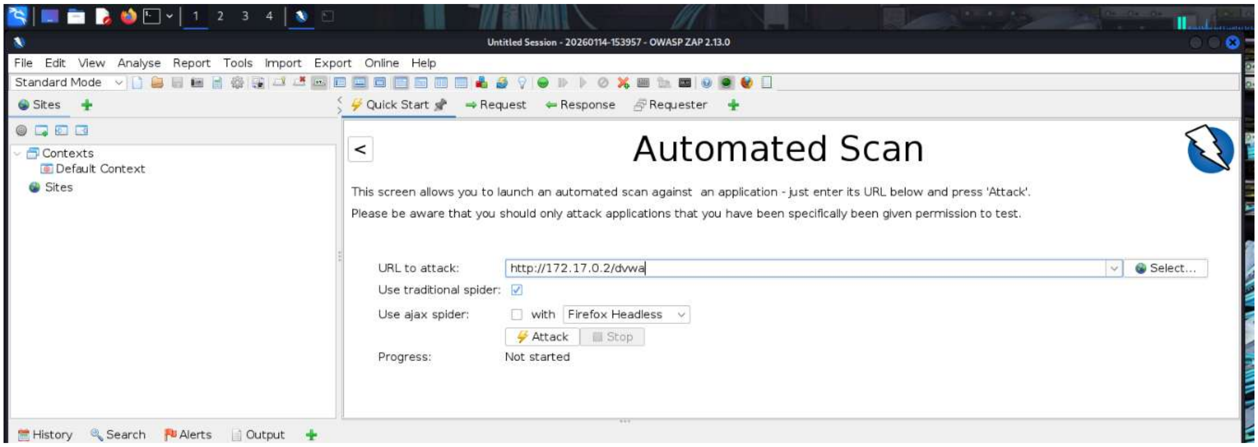
The Replacer options page allows you to quickly and easily replace text anywhere in an HTTP(S) message.

```
INFO: loaded passive scan rule: Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
INFO: loaded passive scan rule: WSDL File Detection
INFO: loaded passive scan rule: Authentication Request Identified
INFO: loaded passive scan rule: Session Management Response Identified
INFO: loaded passive scan rule: Verification Request Identified
INFO: Initializing Alerts Extension - Allows you to view and manage alerts
INFO: Initializing Active Scan Extension - Active scanner, heavily based on the original Paros active scanner, but with additional tests added
INFO: Initializing Standard Menus Extension - A set of common popup menus for miscellaneous tasks
INFO: Initializing ExtensionBruteForce - Forced browsing of files and directories using code from the OWASP DirBuster tool
INFO: Initializing Compare Extension - Compares 2 sessions and generates an HTML file showing the differences
INFO: Initializing ExtensionInvoke - Invoke external applications passing context related information such as URLs and parameters
INFO: Initializing Anti-CSRF Extension - Handles anti cross site request forgery (CSRF) tokens
INFO: Initializing Authentication Extension - Authentication Extension
INFO: Loaded authentication method types: [Form-based Authentication, HTTP/NTLM Authentication, Manual Authentication, Script-based Authentication, JSON-based Authentication]
INFO: Initializing Log4j Extension - Logs errors to the Output tab in development mode only
INFO: Initializing Users Extension - Users Extension
INFO: Initializing Parameters Extension - Summarise and analyse FORM and URL parameters as well as cookies
INFO: Initializing Script Extension - Script integration
INFO: Initializing ExtensionScripts - Scripting console, supports all JSR 223 scripting languages
```

2. Step-by-Step Execution

Step 1: Initialize OWASP ZAP and Scan

1. Launch **OWASP ZAP** from the Kali Linux menu.
2. Select **"Yes, I want to persist this session"** and click Start.
3. On the **Quick Start** tab, click the **Automated Scan** button.
4. Enter the URL: `http://172.17.0.2/dvwa`.


Untitled Session - 20260114-153957 - OWASP ZAP 2.13.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Contexts
Default Context
Sites
History
Search
Alerts
Output

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: http://172.17.0.2/dvwa

Use traditional spider: ☑

Use ajax spider: □ with Firefox Headless

Attack Stop

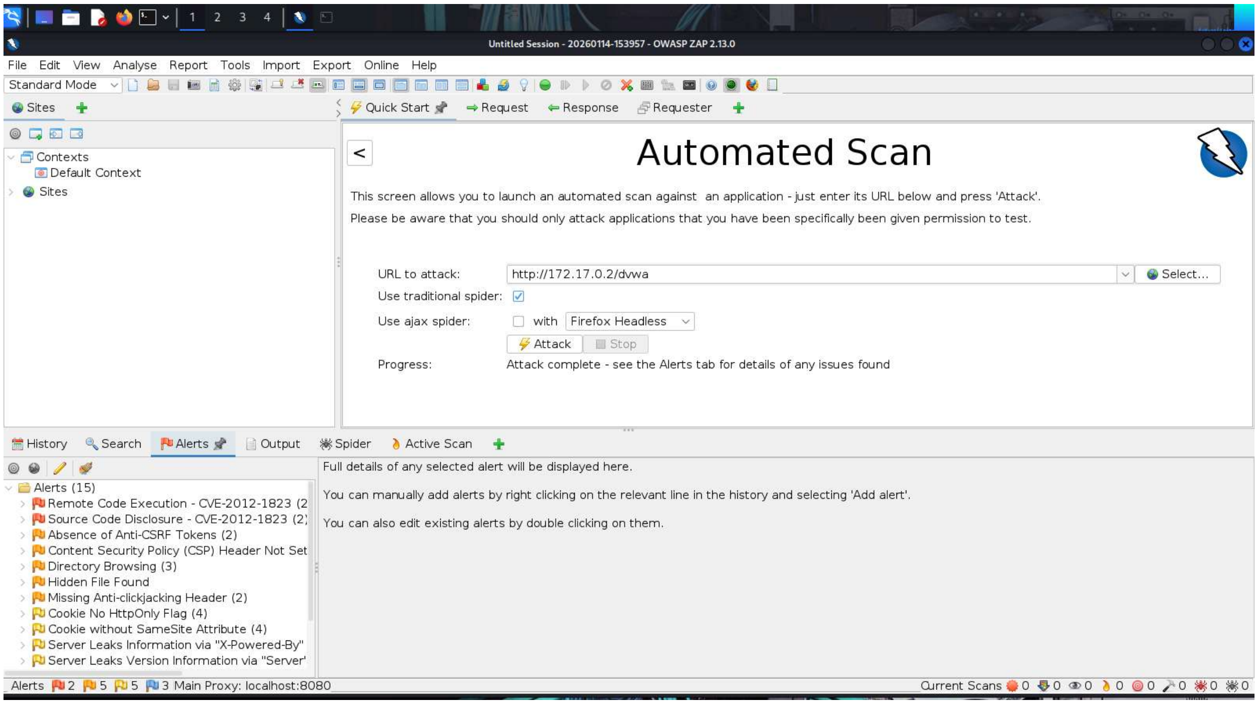
Progress: Not started

5. Click **Attack** to begin.

Step 2: Investigating Scan Results (Alerts)

The scan utilizes a **Spider** to map the site and an **Active Scan** to test for vulnerabilities. Once finished, we review the findings in the **Alerts** tab.

- **Total Alerts Identified:** Approximately **15** alerts.



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: http://172.17.0.2/dvwa

Use traditional spider: ☑

Use ajax spider: □ with Firefox Headless

Attack Stop

Progress: Attack complete - see the Alerts tab for details of any issues found

Alerts (15)
- Remote Code Execution - CVE-2012-1823 (2
- Source Code Disclosure - CVE-2012-1823 (2)
- Absence of Anti-CSRF Tokens (2)
- Content Security Policy (CSP) Header Not Set
- Directory Browsing (3)
- Hidden File Found
- Missing Anti-clickjacking Header (2)
- Cookie No HttpOnly Flag (4)
- Cookie without SameSite Attribute (4)
- Server Leaks Information via "X-Powered-By"
- Server Leaks Version Information via "Server"

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

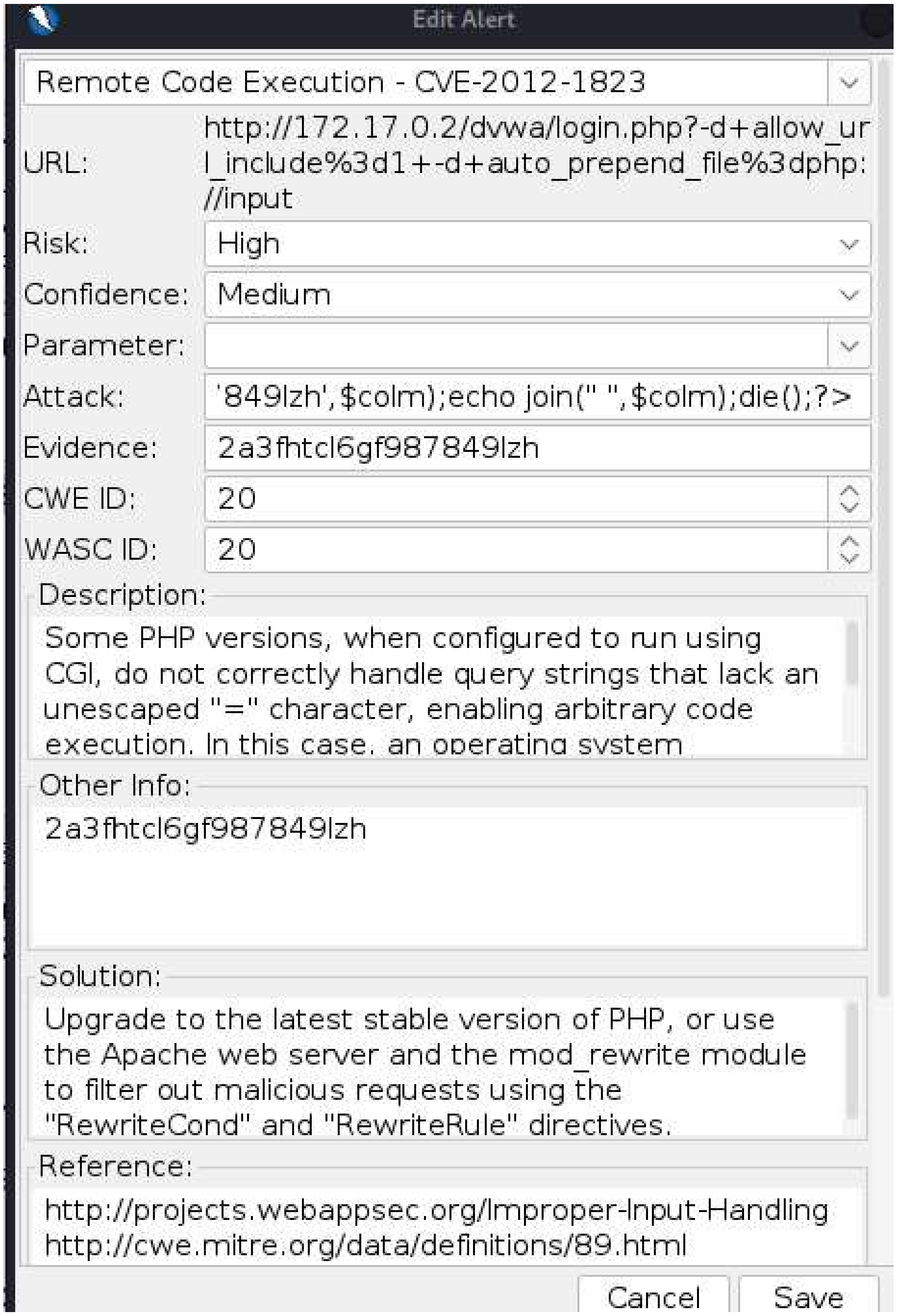
Alerts 2 5 5 3 Main Proxy: localhost:8080

Current Scans 0 0 0 0 0 0 0 0

Analysis of CVE-2012-1823 (Remote Code Execution)

Locate and click the alert titled **"Remote Code Execution – CVE-2012-1823"**.

- **Source of Vulnerability:** An out-of-date PHP version (PHP-CGI).
- **Exploitation Method:** When PHP is configured as CGI, it may fail to correctly handle query strings. Attackers can inject command-line arguments into the query string to execute arbitrary code on the server.



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: '849lzh',$colm);echo join(" ",$colm);die();?>

Evidence: 2a3fhtcl6gf987849lzh

CWE ID: 20

WASC ID: 20

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

Other Info:

2a3fhtcl6gf987849lzh

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.

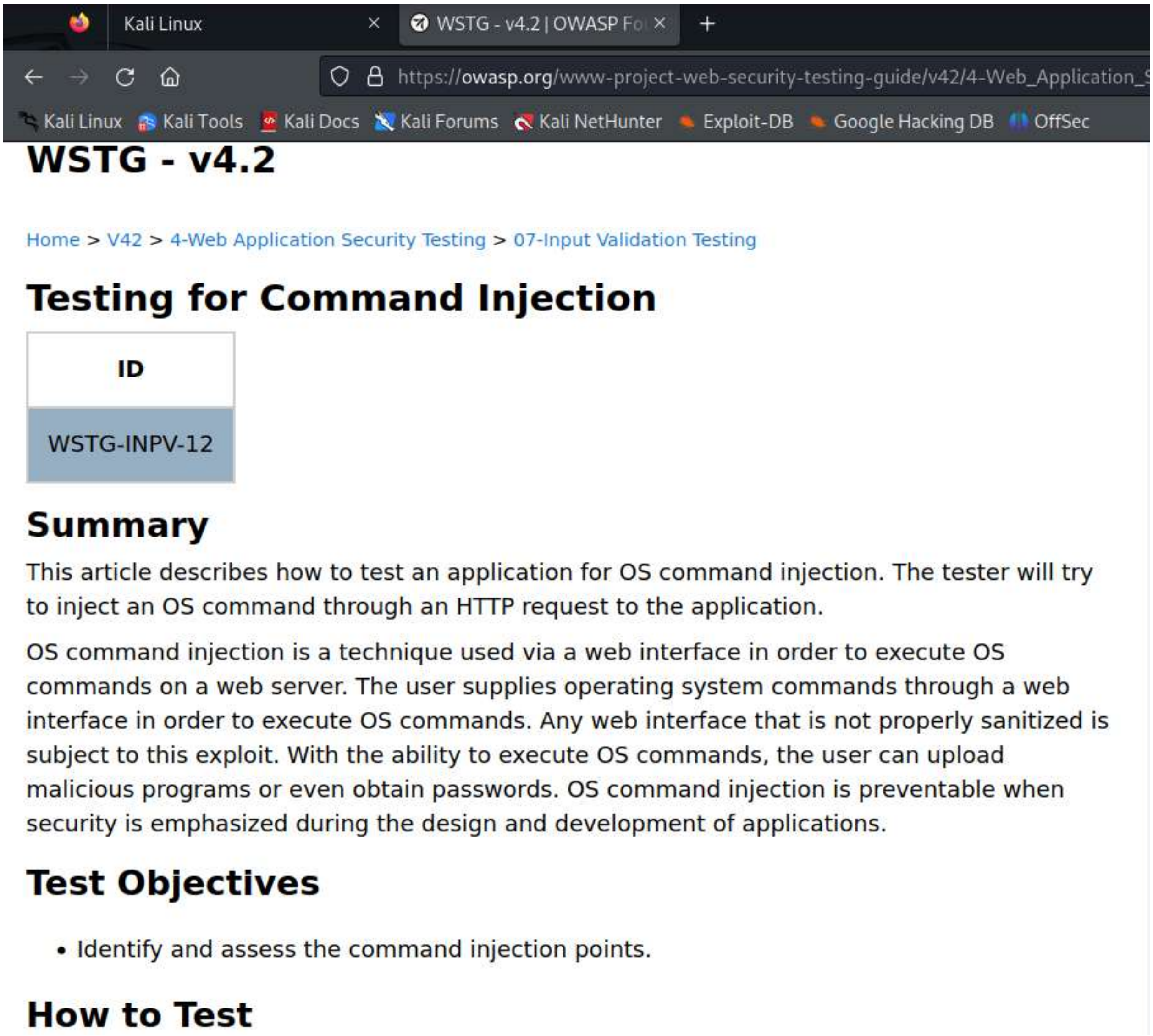
Reference:

http://projects.webappsec.org/Improper-Input-Handling
http://cwe.mitre.org/data/definitions/89.html

Step 3: Referencing the OWASP WSTG

We use the **WSTG** (Web Security Testing Guide) to find standardized testing procedures.

1. In ZAP, scroll to the **Alert Tags** section of the RCE vulnerability.
2. Copy the URL for the WSTG reference.
3. Open the URL in a browser to view the official testing documentation.



WSTG - v4.2

Home > V42 > 4-Web Application Security Testing > 07-Input Validation Testing

Testing for Command Injection

ID
WSTG-INPV-12

Summary

This article describes how to test an application for OS command injection. The tester will try to inject an OS command through an HTTP request to the application.

OS command injection is a technique used via a web interface in order to execute OS commands on a web server. The user supplies operating system commands through a web interface in order to execute OS commands. Any web interface that is not properly sanitized is subject to this exploit. With the ability to execute OS commands, the user can upload malicious programs or even obtain passwords. OS command injection is preventable when security is emphasized during the design and development of applications.

Test Objectives

- Identify and assess the command injection points.

How to Test

3. Reflection

The WSTG provides a comprehensive framework that helps organizations identify security gaps, ensure developers are coding according to industry best practices, and help security teams make informed decisions about which tools and testing activities to prioritize.

The WSTG serves as a manual or "playbook" for the tester. It breaks down web security into 10 categories (e.g., Input Validation, Identity Management) and provides specific, repeatable steps for testing each one, ensuring that no vulnerability class is overlooked during the engagement.

4. Conclusion

Automated scanning with ZAP identified 15 potential issues, most notably a High-risk RCE vulnerability. Cross-referencing these findings with the WSTG allows a penetration tester to move beyond simple "point-and-click" scanning and perform deep, manual verification of the security posture.