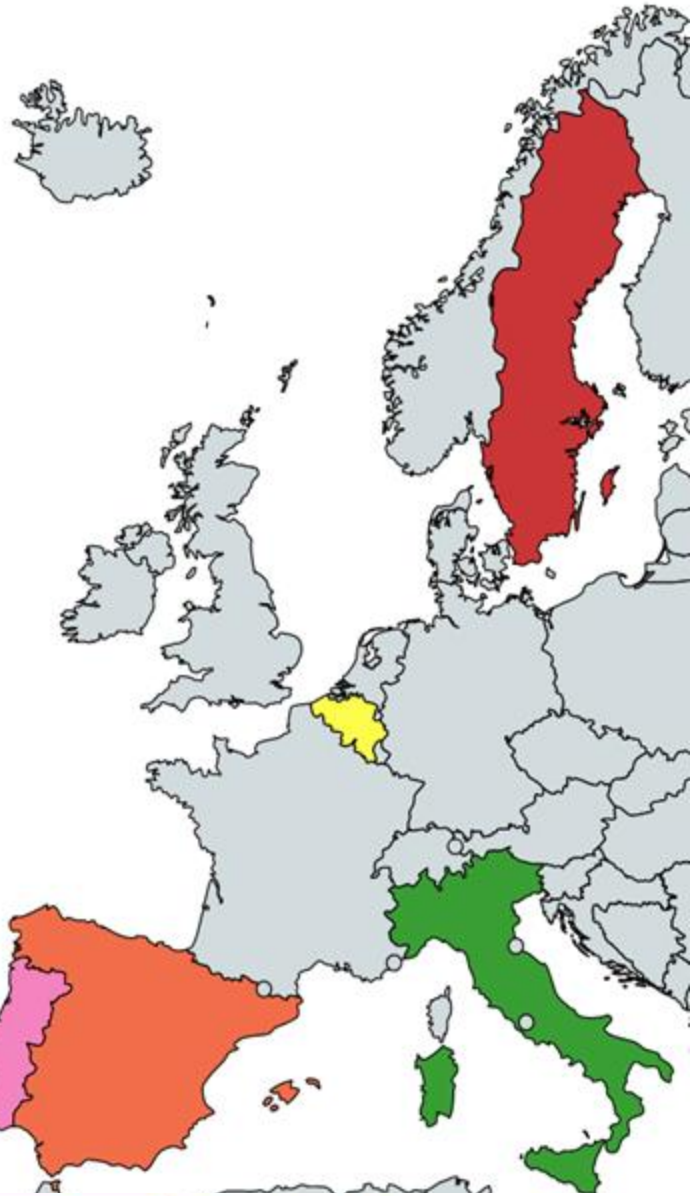

Funded by the European Union

ERASMUS plus Project 2022-2025

ENACTEST

WP4 · Capsule 12

Author: NEXO QA

Document version: 1.0

Cybersecurity Testing

Capsule teaching scheme

1. Cybersecurity testing fundamentals
   - 1.1 Overall goals of cybersecurity testing
   - 1.2 Types of cybersecurity tests
   - 1.3 Core components of cybersecurity testing
   - 1.4 OWASP Testing Methodology
2. Dynamic application security testing fundamentals
   - 2.1 Key characteristics of DAST
   - 2.2 How DAST works & what it tests for
   - 2.3 Benefits & limitations of DAST
   - 2.4 DAST best practices
   - 2.5 Penetration testing
3. Cybersecurity testing tools
   - 3.1 OWASP Penetration Testing Kit

Prerequisites

- Software development fundamentals
- Web application concepts
- Testing concepts
- SQL basics
- Development tools & environments

Contents

1. Cybersecurity Fundamentals

1. Cybersecurity Fundamentals

1. Overall goals of cybersecurity testing
2. Types of cybersecurity tests
3. Core components of cybersecurity testing
4. OWASP Testing Methodology

1.1 Overall goals of cybersecurity testing

- Identify vulnerabilities before attackers do
- Ensure Confidentiality, Integrity, and Availability (CIA Triad)
- Verify compliance with security standards and regulations
- Evaluate the effectiveness of security controls
- Simulate real-world attacks in a controlled way
- Reduce the risk and impact of security incidents
- Support Secure Software Development Lifecycle (SSDLC)

1.2 Types of cybersecurity tests

- Static Application Security Testing (SAST)
- Dynamic Application Security Testing (DAST)
- Interactive Application Security Testing (IAST)
- Penetration testing
- Security scanning
- Authentication and access control testing
- Dependency and supply chain testing
- Configuration and infrastructure testing

1.4 OWASP Testing Methodology

1. Information gathering
2. Configuration and deployment management testing
3. Identity management testing
4. Authentication testing
5. Authorisation testing
6. Session management testing
7. Input validation testing

1.4 OWASP Testing Methodology

8. Error handling
9. Cryptography testing
10. Business Logic Testing
11. Client-Side Testing

2. DAST Fundamentals

2. DAST Fundamentals

1. Key characteristics of DAST
2. How DAST works & what it tests for
3. Benefits & limitations of DAST
4. DAST best practices
5. Penetration testing

2.1 Key characteristics of DAST

- Black-box testing (no source code Insight)
- Real HTTP requests & responses tests
- Language-agnostic
- Detects runtime Issues
- Easy integration into CI/CD pipelines
- Focuses on web application & API security
- Often uses crawlers and attack payloads
- Actionable reports

2.2 How DAST works

1. Deploy the Application
2. Crawler/Spider Maps the App
3. Input Injection + Behavior Analysis
4. Heuristics + Signature Matching
5. Report and Recommendations

2.2 What DAST tests

- Injection Attacks
- Cross-Site Scripting (XSS)
- Authentication & Authorisation Issues
- Session Management Flaws
- Cross-Site Request Forgery (CSRF)
- Error Handling & Information Disclosure
- Security Misconfigurations
- Test insecure API Endpoints

2.3 Benefits of DAST

- Black-Box, Real-World Simulation
- Technology & Language Agnostic
- overs Runtime & Deployment Issues
- Integrates into CI/CD Pipelines
- Actionable Reports for Developers
- Effective for API & Web Apps

2.3 Limitations of DAST

- No Source Code Visibility
- Limited to Exposed Interfaces
- Struggles with Modern JavaScript Apps
- May Cause False Positives/Negatives
- Can Be Slow or Resource-Intensive
- Needs a Running, Stable App

2.4 DAST best practices

1. Test in a Dedicated Staging Environment
2. Include Authenticated Scans
3. Use a Well-Defined Test Plan
4. Prioritise OWASP Top 10 and Known CVEs
5. Customize and Tune Payloads

2.4 DAST best practices

6. Exclude Sensitive Operations
7. Test in a Dedicated Staging Environment
8. Exclude Sensitive Operations
9. Baseline and Compare Results
10. Combine DAST with Other Techniques

2.5 Web Penetration Testing

1. Web Penetration Testing Process
2. SQL Injection fundamentals
3. Cross-Site Scripting (XSS) fundamentals

2.5.1 Penetration Testing Process

1) **Explore**: Tester learns about the system being tested
2) **Attack**: Tester attempts to exploit vulnerabilities to prove they exist
3) **Report**: Tester reports back the results of testing (vulnerabilities & exploits)

2.5.2 SQL Injection fundamentals

What Is SQL Injection?

- A type of injection attack that targets databases via insecure input handling.
- Occurs when user-supplied data is improperly included in an SQL query
- Allows attackers to manipulate SQL statements executed by the server

2.5.2 SQL Injection fundamentals

How it happens?

- Input fields (e.g. login forms, search bars, URLs) directly embedded into SQL queries without sanitisation
- Unsanitised input alters query structure or logic.

2.5.2 SQL Injection fundamentals

Example

Insecure login query:

```
SELECT * FROM users WHERE username = 'admin' AND password = '1234';
```

Attacker input:

```
admin' OR '1'='1
```

Becomes:

```
SELECT * FROM users WHERE username = 'admin' OR '1'='1';
```

2.5.2 SQL Injection fundamentals

What attackers can do

- Bypass authentication
- Access, modify, or delete data
- Extract database structure (e.g., table names, columns)
- Execute administrative operations
- In rare cases: command execution on host (via stacked queries or DB

2.5.2 SQL Injection fundamentals

Types of SQL Injection

- Classic SQLi – directly manipulates SQL
- Blind SQLi – responses don't show output, attacker infers from behavior (e.g., timing)
- Boolean-based – relies on true/false conditions
- Time-based – injects delays to infer data (e.g., IF(condition, SLEEP(5), 0))
- Out-of-band SQLi – uses external data exfiltration (e.g., DNS, HTTP)

2.5.2 SQL Injection fundamentals

Vulnerable Components

- Login forms
- URL parameters
- Search boxes
- Filters and sorting inputs
- Hidden fields in forms
- Cookies or headers

2.5.2 SQL Injection fundamentals

How to Prevent SQL Injection

- Use parameterized queries / prepared statements
- Use ORM libraries that enforce safe query construction
- Validate and sanitize all user input
- Apply least privilege to DB accounts
- Use Web Application Firewalls (WAFs)
- Conduct regular DAST/SAST scans and manual code review

2.5.2 SQL Injection fundamentals · Let's try it

Manual SQL Injection Testing in ZAP GUI

1. Start ZAP & configure your browser to use ZAP's proxy (127.0.0.1:8080)
2. Browse the target app manually — login forms, search bars, query strings, etc.
3. In ZAP's sites tree, right-click a target URL:
   - Select Attack → Active Scan
   - ZAP will automatically test for:
     - ✓ Error-based SQLi
     - ✓ Boolean-based SQLi
     - ✓ Time-based SQLi (somewhat limited)

2.5.2 SQL Injection fundamentals · Let's try it

Manual SQL Injection Testing in ZAP GUI

4. Open the Alerts tab to review findings like:
    - SQL Injection
    - SQL Injection – MySQL
    - SQL Injection – Hypothetical

3. Cybersecurity Testing Tools

3.1 OWASP Penetration Testing Kit

The OWASP Penetration Testing Kit (PTK) is a collective term sometimes used to describe a bundle of OWASP projects, checklists, and tools that can be used together for structured penetration testing

While OWASP doesn't distribute a PTK as a package (like Kali Linux), it offers several key resources that form a powerful open-source "kit" for ethical hacking and application security assessments.

3.2 OWASP PTK Key components

1. OWASP Web Security Testing Guide (WSTG)
2. OWASP ZAP (Zed Attack Proxy)
3. OWASP Cheat Sheet Series
4. OWASP Application Security Verification Standard (ASVS)
5. OWASP Top 10
6. OWASP Security Knowledge Framework (SKF)

2.5.2 SQL Injection fundamentals

Let’s take a look. . .