Secure coding and security testing in Java for DevSecOps

CYDJvDop3d | 3 days | On-site or online | Hands-on

The course provides an in-depth exploration of security concerns and best practices tailored specifically for DevOps engineers working on Java software on the AWS cloud platform. Starting off from the foundations of cybersecurity, you will understand the consequences of insecure code by examining threats through the lens of the CIA triad.

In the main part of the material, you will go through the various security issues outlined in the OWASP Top Ten with a focus on DevSecOps issues - identity management in microservice and cloud environments, secure AWS configuration, securing CI / CD build processes, secrets management, and logging and monitoring. Finally, you'll explore cloud security with a focus on security automation and tooling in AWS, the security of containers and container orchestration (Docker, Kubernetes), microservices, and Infrastructure as Code tools (CloudFormation, Terraform), and security testing tools relevant for DevSecOps.

These modules go beyond just theory. Not only do they show vulnerabilities, their consequences, and corresponding best practices, but - through hands-on labs and real-world case studies - they offer practical experience in identifying, exploiting, and mitigating these security risks.

So that you are prepared for the forces of the dark side.

So that nothing unexpected happens.

Nothing.

Skills and drills

32 LABS

10 CASE STUDIES

Audience

Java architects, developers and testers

Group size

12 participants

Preparedness

DevSecOps, General Java and Web development, testing and QA

Outline

- Cyber security basics
- The OWASP Top Ten 2021
- Cloud security
- Security testing
- Wrap up

Standards and references

OWASP, SEI CERT, CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Understanding Web application security issues
- Detailed analysis of the OWASP Top Ten elements
- Putting Web application security in the context of Java
- Going beyond the low hanging fruits
- Managing vulnerabilities in third party components
- Learn to deal with cloud infrastructure security
- Understand cloud security specialties
- Understanding security testing methodology and approaches
- Getting familiar with security testing techniques and tools

Table of contents

Day 1

    - Testing SQL injection
    - SQL injection tools
    - *Lab – Using SQL injection tools*
  - SQL injection best practices
    - Input validation
    - Parameterized queries
    - *Lab – Using prepared statements*
    - *Case study – SQL injection in Fortra FileCatalyst*
- NoSQL injection
  - NoSQL injection basics
  - NoSQL injection in MongoDB
  - NoSQL injection in DynamoDB
- Code injection
  - OS command injection
    - OS command injection best practices
    - Using Runtime.exec()
    - *Case study – Shellshock*
    - *Lab - Shellshock*
    - *Case study – Command injection in VMware Aria*
    - Testing for command injection

Day 2

› The OWASP Top Ten 2021

A03 - Injection (continued)

- HTML injection - Cross-site scripting (XSS)
  - Cross-site scripting basics
  - Cross-site scripting types
    - Persistent cross-site scripting
    - Reflected cross-site scripting
    - Client-side (DOM-based) cross-site scripting
  - *Lab – Stored XSS*
  - *Lab – Reflected XSS*
  - Testing for XSS
    - Testing for XSS with tools
    - *Lab – Using XSS testing tools*
  - XSS protection best practices
    - Protection principles - escaping
    - XSS protection APIs in Java
    - *Lab – XSS fix / stored*
    - *Lab – XSS fix / reflected*

- Additional protection layers – defense in depth
- *Case study – XSS vulnerabilities in DrayTek Vigor routers*

A04 - Insecure Design

- Client-side security
  - Same Origin Policy
  - Frame sandboxing
    - Cross-Frame Scripting (XFS) attacks
    - *Lab - Clickjacking*
    - Clickjacking protection best practices
    - *Lab – Using CSP to prevent clickjacking*
  - Testing for client-side security weaknesses

A05 - Security Misconfiguration

- Configuration principles
- AWS configuration best practices
  - Common AWS misconfiguration issues
  - AWS configuration best practices
- Testing for misconfiguration issues
- XML entities
  - DTD and the entities
  - Entity expansion
  - External Entity Attack (XXE)
    - File inclusion with external entities
    - Server-Side Request Forgery with external entities
    - *Lab – External entity attack*
    - Preventing XXE
    - *Lab – Prohibiting DTD*
    - *Case study – XXE vulnerability in Ivanti products*
  - Testing for XXE and XML entity-related vulnerabilities

A06 - Vulnerable and Outdated Components

- Using vulnerable components
- Vulnerability management
  - *Lab – Finding vulnerabilities in third-party components*
- Build security
  - DevOps, the CI / CD build process and Software Composition Analysis
  - Dependency checking in Java
  - *Lab – Detecting vulnerable components*

A07 - Identification and Authentication Failures

- Session management
  - Session management essentials

- Why do we protect session IDs – Session hijacking
- Session fixation
- Testing for session management issues

- Storing sensitive data in memory
  - Hard coded passwords
  - Best practices
  - *Lab – Hardcoded password*
  - Storing cryptographic keys on AWS
  - *Lab – Using the AWS Key Management Service*
- Identity and access management (IAM)
  - Identity and access management in AWS
  - Groups, roles and credentials
  - Access tokens
  - *Lab – STS in AWS*

A09 - Security Logging and Monitoring Failures

- Logging and monitoring principles
- Log forging
- Log forging – best practices

Case study – Log interpolation in log4j

Case study – The Log4Shell vulnerability (CVE-2021-44228)

Case study – Log4Shell follow-ups (CVE-2021-45046, CVE-2021-45105)

Lab – Log4Shell

- Logging best practices
- Testing for logging issues
- Detection and monitoring
  - Utilizing AWS monitoring for security
  - Protecting logs
  - The AWS Security Hub

Day 3

› Cloud security

Cloud security basics

- Cloud infrastructure basics
- The Cloud Cube Model and Zero Trust Architecture

Case study – ChaosDB vulnerability in Azure Cosmos DB

AWS security

- Security considerations
  - AWS and security
  - The AWS shared responsibility model
  - AWS cloud compliance
  - AWS hardening
  - Security tools for AWS

Container security

- Container security concerns
- Containerization, virtualization and security
- The attack surface
- Docker security
  - Docker and security
  - Docker security best practices
  - *Lab – Static analysis of Docker images*
  - Hardening Docker containers
- Kubernetes security
  - The Kubernetes architecture and security
  - Securing Kubernetes hosts
  - Best practices for Kubernetes access control
  - Protecting Kubernetes deployments at runtime
  - *Lab – Kubernetes security testing*
  - Kubernetes security tools
  - *Lab – Scanning a Kubernetes cluster for vulnerabilities*
  - *Case study – Azurescape*
- Infrastructure as Code
  - Infrastructure as Code security
  - AWS CloudFormation security
  - Typical mistakes in CloudFormation configuration files
  - Terraform security
  - Terraform threats and best practices

› Security testing

Security testing vs functional testing

Manual and automated methods

Black box, white box, and hybrid testing

Security testing methodology

- Security testing – goals and methodologies

- Overview of security testing processes
- Identifying assets
- Assigning security requirements

🔬 *Lab – Identifying and rating assets*

- Threat modeling
  - SDL threat modeling
  - Mapping STRIDE to DFD
  - DFD example
  - Attack trees
  - Attack tree example
  - Risk analysis

Security testing techniques and tools

- Code analysis
  - Static Application Security Testing (SAST)

  🔬 *Lab – Using static analysis tools*
- Dynamic analysis
  - Security testing at runtime
  - Penetration testing
  - Stress testing
  - Dynamic analysis tools
    - Dynamic Application Security Testing (DAST)
    - IAST and DevSecOps
    - Web vulnerability scanners

    🔬 *Lab – Using web vulnerability scanners*
    - Fuzzing

DevSecOps security testing

- IaC scanning tools

🔬 *Lab – Terraform security scanning*

🔬 *Lab – CloudFormation security scanning*

- Secrets scanning

🔬 *Lab – Finding secrets in Git repositories*

- Cloud infrastructure scanning

🔬 *Lab – Using a cloud environment scanner*

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop
- Secure design principles of Saltzer and Schroeder

And now what?

- Software security sources and further reading
- Java resources
- Security testing resources