Cloud application security in Java for AWS

CYDJvCldAWS5d | 5 days | On-site or online | Hands-on

Your cloud application written in Java works as intended, so you are done, right? But did you consider feeding in incorrect values? 16Gbs of data? A null? An apostrophe? Negative numbers, or specifically -1 or -2^31? Because that's what the bad guys will do – and the list is far from complete.

The cloud has become a critical aspect of online services. No matter which model you're using (SaaS, PaaS, IaaS), part of your service is now operated by someone else. This may look like a net positive, but it also greatly expands the attack surface and brings in several new risks that may not be obvious. Have you configured all security settings correctly? Are you prepared for supply chain attacks? How can you protect the confidentiality of user data in the cloud? And most importantly: can the bad guys use your exposure to their advantage?

Handling security needs a healthy level of paranoia, and this is what this course provides: a strong emotional engagement by lots of hands-on labs and stories from real life, all to substantially improve code hygiene. Mistakes, consequences, and best practices are our blood, sweat and tears.

The curriculum goes through the common Web application security issues following the OWASP Top Ten but goes far beyond it both in coverage and the details.

All this is put in the context of Java, and extended by core programming issues, discussing security pitfalls of the Java language and the AWS cloud platform.

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

So that nothing unexpected happens.

Nothing.

Skills and drills

39 LABS

17 CASE STUDIES

Audience

Java developers working on Web applications and AWS

Group size

12 participants

Preparedness

General Java and Web development

Outline

- Cyber security basics
- The OWASP Top Ten 2025
- Cloud security
- Input validation
- Code quality
- Wrap up

Standards and references

OWASP, SEI CERT, CWE and Fortify Taxonomy

What you'll have learned

- Understand cloud security specialties
- Getting familiar with essential cyber security concepts
- Managing vulnerabilities in third party components
- 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
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly in Java
- Getting familiar with security testing techniques and tools
- Learn to deal with cloud infrastructure security
- Input validation approaches and principles
- Identify vulnerabilities and their consequences
- Learn the security best practices in Java

Table of contents

Day 1

› Cyber security basics

What is security?

Threat and risk

Cyber security threat types – the CIA triad

Consequences of insecure software

Cloud security basics

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

Case study – ChaosDB vulnerability in Azure Cosmos DB

› The OWASP Top Ten 2025

A01 - Broken Access Control

- Access control basics

Case study – Broken authn/authz in Apache OFBiz

- Confused deputy
  - Insecure direct object reference (IDOR)
  - Path traversal

  Case study – RCE via path traversal in Apache OFBiz

  Lab – Insecure Direct Object Reference
  - Path traversal best practices
  - Authorization bypass through user-controlled keys

  Case study – Remote takeover of Nexx garage doors and alarms

  Lab – Horizontal authorization
- File upload
  - Unrestricted file upload
  - Good practices

  Lab – Unrestricted file upload

  Case study – File upload vulnerability in Netflix Genie
- Server-side Request Forgery (SSRF)

  Case study – SSRF in Ivanti Connect Secure

A02 - Security Misconfiguration

- Secrets management
  - Hard coded passwords
  - Best practices
  - *Lab – Hardcoded password*
- 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*

A03 - Software Supply Chain Failures

- Using vulnerable components
- Untrusted functionality import
- Supply chain security and the Software Bill of Materials (SBOM)
- SBOM examples
- *Case study – The Polyfill.io supply chain attack*
- Vulnerability management
  - *Lab – Finding vulnerabilities in third-party components*
  - DevOps, the CI / CD build process and Software Composition Analysis
  - Dependency checking in Java
  - *Lab – Detecting vulnerable components*

Day 2

› The OWASP Top Ten 2025

A04 - Cryptographic Failures (continued)

- Cryptography for developers (continued)
  - Confidentiality protection (continued)
    - Confidentiality protection
    - Symmetric encryption
    - Block ciphers
    - Modes of operation
    - Modes of operation and IV – best practices

    - Symmetric encryption in Java
    - Symmetric encryption in Java with streams
    - *Lab – Symmetric encryption in JCA*
    - Asymmetric encryption
    - Combining symmetric and asymmetric algorithms
    - Key exchange and agreement
    - Key exchange
    - Diffie-Hellman key agreement algorithm
    - Key exchange pitfalls and best practices
  - Public Key Infrastructure (PKI)
    - Key management challenges
    - Certificates
    - Certificates and PKI
    - Chain of trust
    - PGP – Web of Trust
  - Transport security
    - The TLS protocol
    - TLS basics
    - TLS features (changes in v1.3)
    - The handshake in a nutshell (v1.3)
    - An example TLS 1.3 handshake
    - TLS best practices
    - Java Secure Socket Extension (JSSE)
    - *Lab – Using a secure socket in JCA*
    - HTTP Strict Transport Security (HSTS)

A05 - Injection

- Injection principles
- Injection attacks
- SQL injection
  - SQL injection basics
  - *Lab – SQL injection*
  - Attack techniques
  - Content-based blind SQL injection
  - Time-based blind SQL injection
  - SQL injection best practices
    - Input validation
    - Parameterized queries
    - *Lab – Using prepared statements*
    - Database defense in depth
    - *Case study – SQL injection in Fortra FileCatalyst*
    - SQL injection protection and ORM
  - NoSQL injection
    - NoSQL injection basics

    - NoSQL injection in MongoDB
    - NoSQL injection in DynamoDB
- Parameter manipulation
  - Newline injection
  - HTTP header manipulation
    - HTTP response splitting
  - HTTP parameter manipulation
    - HTTP parameter pollution
    - Value shadowing
    - Variable shadowing
- 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*

Day 3

› The OWASP Top Ten 2025

A05 - 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*
  - XSS protection best practices
    - Protection principles - escaping
    - XSS protection APIs in Java
    - *Lab – XSS fix / stored*
    - *Lab – XSS fix / reflected*
    - Client-side protection principles
    - Additional protection layers – defense in depth
    - *Case study – XSS vulnerabilities in DrayTek Vigor routers*

A06 - Insecure Design

- The STRIDE model of threats

- Secure design principles of Saltzer and Schroeder
  - Economy of mechanism
  - Fail-safe defaults
  - Complete mediation
  - Open design
  - Separation of privilege
  - Least privilege
  - Least common mechanism
  - Psychological acceptability
- Client-side security
  - Same Origin Policy
    - Simple request
  - Frame sandboxing
    - Cross-Frame Scripting (XFS) attacks
    - *Lab – Clickjacking*
    - Clickjacking beyond hijacking a click
    - Clickjacking protection best practices
    - *Lab – Using CSP to prevent clickjacking*

A07 - Authentication Failures

- Authentication
  - Authentication basics
  - Multi-factor authentication (MFA)
  - *Case study – The InfinityGauntlet attack*
  - Time-based One Time Passwords (TOTP)
- Session handling
  - Session management essentials
  - Why do we protect session IDs – Session hijacking
  - Session fixation
  - Session ID best practices
- Token-based authentication and authorization
  - Advantages and challenges
  - Token protection best practices
  - JSON Web Tokens (JWT)
  - Integrity protection of JWT
- Cookie security
  - Cookie attributes
- Single sign-on (SSO)
  - Single sign-on concept
  - OAuth 2.0
    - OAuth 2.0 basics

    - OAuth 2.0 prerequisites
    - OAuth 2.0 flow for Authorization Code Grant – Step #1
    - OAuth 2.0 flow for Authorization Code Grant – Step #2
    - OAuth 2.0 flow for Authorization Code Grant – Step #3
    - OAuth 2.0 flow for Authorization Code Grant – Step #4
    - OAuth 2.0 flow for Authorization Code Grant – Step #5
    - OAuth 2.0 flow for Authorization Code Grant – Step #6
    - OAuth 2.0 flow for Authorization Code Grant – Step #7
    - Best practices
  - SAML
    - SAML basics
    - SAML profiles
    - SAML security
- Identity and access management (IAM)
  - Identity and access management in AWS
  - Groups, roles and credentials
  - Access tokens
  - *Lab – STS in AWS*

Day 4

› The OWASP Top Ten 2025

A07 - Authentication Failures (continued)

- Password management
  - Storing account passwords
  - Password in transit
  - *Lab – Is just hashing passwords enough?*
  - Dictionary attacks and brute forcing
  - Salting
  - Adaptive hash functions for password storage
  - *Lab – Using adaptive hash functions in JCA*
- Password policy
  - NIST authenticator requirements for memorized secrets

A08 - Software and Data Integrity Failures

- Integrity protection
  - Message Authentication Code (MAC)
  - Digital signature
    - Elliptic Curve Cryptography
    - ECC basics

    - Digital signature with ECC
- Subresource integrity
    - Importing JavaScript
    - *Lab – Importing JavaScript*
    - *Case study – The British Airways data breach*
- Insecure deserialization
    - Serialization and deserialization challenges
    - Integrity – deserializing untrusted streams
    - Integrity – deserialization best practices
    - Property Oriented Programming (POP)
        - Creating a POP payload
        - *Lab – Creating a POP payload*
        - *Lab – Using the POP payload*
        - *Case study – Deserialization RCEs in NextGen Mirth Connect*

› Cloud security

AWS security

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

Infrastructure as Code

- Infrastructure as Code security
- AWS CloudFormation security
- Typical mistakes in CloudFormation configuration files
- Terraform security
- Terraform threats and best practices
- IaC security tools
    - 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*

Day 5

The OWASP Top Ten 2025

A10 - Mishandling of Exceptional Conditions

- Error and exception handling principles
- Exception handling
  - In the catch block. And now what?
  - Catching NullPointerException
  - Empty catch block
  - *Lab – Exception handling mess*
- Control flow
  - Incorrect block delimitation
  - Dead code
  - Using if-then-else and switch defensively

Web application security beyond the Top Ten

- X01 - Lack of Application Resilience
  - Denial of service
  - Resource exhaustion
  - Algorithmic complexity issues
    - Regular expression denial of service (ReDoS)
    - *Lab – ReDoS*
    - Dealing with ReDoS

› Cloud security

Data security in AWS

Policies

Protecting data at rest

Storing cryptographic keys on AWS

Lab – Using the AWS Key Management Service

Protecting data in transit

JSON security

- Best practices

Case study – ReactJS vulnerability in HackerOne

› Input validation

Input validation principles

Denylists and allowlists

What to validate – the attack surface

Where to validate – defense in depth

When to validate – validation vs transformations

Validation with regex

Integer handling problems

- Representing signed numbers
- Integer visualization
- Integer overflow

Lab – Integer overflow

- Signed / unsigned confusion

Case study – The Stockholm Stock Exchange

- Integer truncation
- Best practices
  - Upcasting
  - Precondition testing
  - Postcondition testing
  - Using big integer libraries
  - Integer handling

Lab – Integer handling

Unsafe reflection

- Reflection without validation

Lab – Unsafe reflection

Unsafe native code

- Native code dependence

Lab – Unsafe native code

- Best practices for dealing with native code

› Code quality

Data handling

- Initialization and cleanup
  - Constructors and destructors
  - Class initialization cycles

Lab – Initialization cycles

  - The finalize() method – best practices

Object oriented programming pitfalls

- Inheritance and overriding
- Mutability

Lab – Mutable object

- Cloning

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop

And now what?

- Software security sources and further reading
- Java resources