

Web application security in Java

CELJvWeb | 1 year subscription | e-Learning | Online VM

Your Web 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.

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 runtime environment.

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

So that nothing unexpected happens.

Nothing.

Cyber security skills and drills

Base training



28 LABS



12 CASE STUDIES

Monthly feed for a year



2-3 LABS



CASE STUDY

Audience

Java developers working on Web applications

Outline

- Cyber security basics
- The OWASP Top Ten
- A01 - Broken Access Control
- A02 - Cryptographic Failures
- A03 - Injection
- A04 - Insecure Design
- A05 - Security Misconfiguration
- A06 - Vulnerable and Outdated Components
- A07 - Identification and Authentication Failures
- A08 - Software and Data Integrity Failures
- A10 - Server-Side Request Forgery
- Wrap up

Preparedness

General Java and Web development

Standards and references

OWASP, SEI CERT, CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Identify Web application vulnerabilities and their consequences
- Learn the security best practices in C#
- Input validation approaches and principles
- 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

Table of contents

› Cyber security basics

Security basics

- What is security?
- Threat and risk

Cyber security threats

- [Cyber security threat types – the CIA triad](#)
- Cyber security threat types – the STRIDE model

Insecure software

- Consequences of insecure software
- Constraints and the market
- The dark side

› The OWASP Top Ten

Introduction

- OWASP and the Top 10
- Is it a standard?
- Methodology
- [The OWASP Top 10 2021](#)

› A01 – Broken Access Control

Authorization

- Access control basics
- Failure to restrict URL access
- Confused deputy

Insecure Direct Object Reference

- Insecure direct object reference (IDOR)
- Path traversal

 *Lab – Insecure Direct Object Reference*

- Path traversal best practices

Horizontal authorization

- Authorization bypass through user-controlled keys

 *Case study – Authorization bypass on Facebook*

 *Lab – Horizontal authorization*

File upload

- Unrestricted file upload
- Good practices

 *Lab – Unrestricted file upload*

Cross-site Request Forgery (CSRF)

 *Lab – Cross-site Request Forgery*

Cross-site Request Forgery (CSRF) best practices

- CSRF best practices
- CSRF defense in depth

 *Lab – CSRF protection with tokens*

› A02 – Cryptographic Failures

Information exposure

- Exposure through extracted data and aggregation

 *Case study – Strava data exposure*

- Leaking system information

Cryptography for developers

- Cryptography basics
- Java Cryptographic Architecture (JCA) in brief

PRNG

- Random number generation
- Pseudo random number generators (PRNGs)
- Cryptographically strong PRNGs
- Using virtual random streams

 *Case study – Equifax credit account freeze*

A02 – Cryptographic Failures

- Weak and strong PRNGs in Java
- Using random numbers in Java

 *Lab – Using random numbers in Java*

Hashing

- Hashing basics
- Common hashing mistakes
- Hashing in Java


 *Lab – Hashing in JCA*

Encryption

- Confidentiality protection
- Symmetric encryption
- [Block ciphers](#)
- Modes of operation
- Modes of operation and IV – best practices

A02 – Cryptographic Failures

- Symmetric encryption in Java
- Symmetric encryption in Java with streams

 *Lab – Symmetric encryption in JCA*

A02 – Cryptographic Failures

- Asymmetric encryption
- The RSA algorithm
- Using RSA – best practices
- RSA in Java
- Combining symmetric and asymmetric algorithms

› A03 – Injection

Injection problems

- Injection principles
- Injection attacks

SQL injection

- SQL injection basics

 *Lab – SQL injection*

SQL injection attack techniques

- Attack techniques
- Content-based blind SQL injection
- Time-based blind SQL injection

SQL injection best practices

- Input validation
- Parameterized queries

 *Lab – Using prepared statements*

SQL injection additional considerations

- Additional considerations


 *Case study – Hacking Fortnite accounts*

Code injection – OS command injection

- Code injection
- OS command injection

OS command injection best practices

- Using `Runtime.exec()`
- Using `ProcessBuilder`

 *Case study – Shellshock*

 *Lab – Shellshock*


HTML injection – Cross-site scripting (XSS)

- [Cross-site scripting basics](#)
- Persistent cross-site scripting
- Reflected cross-site scripting
- Client-side (DOM-based) cross-site scripting

XSS attacks


 *Lab – Stored XSS*

 *Lab – Reflected XSS*

 *Case study – XSS in Fortnite accounts*

XSS best practices 1

- Protection principles – escaping
- XSS protection APIs in Java

 *Lab – XSS fix / stored*

XSS best practices 2

- XSS protection in JSP

 *Lab – XSS fix / reflected*

- Additional protection layers – defense in depth

› A04 – Insecure Design

Insecure design

- The STRIDE model of threats
- Secure design principles of Saltzer and Schroeder

Insecure design – Saltzer and Schroeder 1

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design

Insecure design – Saltzer and Schroeder 2

- Separation of privilege
- Least privilege
- Least common mechanism
- Psychological acceptability

Client-side security

- Same Origin Policy
- Simple request
- Preflight request
- Cross-Origin Resource Sharing (CORS)

Tabnabbing and frame sandboxing

- Tabnabbing
- Frame sandboxing

Clickjacking


- Cross-Frame Scripting (XFS) attacks

 *Lab – Clickjacking*

- Clickjacking beyond hijacking a click

Anti-clickjacking best practices

- Clickjacking protection best practices

 *Lab – Using CSP to prevent clickjacking*

› A05 – Security Misconfiguration


Misconfiguration and XML parsing

- Configuration principles
- XML Entities
- DTD and the entities
- Entity expansion

XML External Entity (XXE)

- File inclusion with external entities
- Server-Side Request Forgery with external entities

 *Lab – External entity attack*

 *Case study – XXE vulnerability in SAP Store*

XXE best practices

- Preventing XXE

 *Lab – Prohibiting DTD*

› A06 – Vulnerable and Outdated Components

Vulnerable components and dependencies

- Using vulnerable components
- Assessing the environment
- Hardening
- Untrusted functionality import

Vulnerability management

- Patch management
- [Vulnerability management](#)
- Vulnerability databases

 *Lab – Finding vulnerabilities in third-party components*

› A07 – Identification and Authentication Failures

Authentication

- Authentication basics
- Multi-factor authentication
- Authentication weaknesses

 *Case study – PayPal 2FA bypass*

Session security

- Session management essentials
- Why do we protect session IDs – Session hijacking
- Session fixation
- Session ID best practices

Password management

- Storing account passwords
- Password in transit

 *Lab – Is just hashing passwords enough?*

- [Dictionary attacks and brute forcing](#)

Password storage

- Salting
- Adaptive hash functions for password storage

 *Lab – Using adaptive hash functions in JCA*

Password policy

- [NIST authenticator requirements for memorized secrets](#)

 *Case study – The Ashley Madison data breach*

 *The dictionary attack*

 *The ultimate crack*

 *Exploitation and the lessons learned*

Password management challenges

- Password database migration
- (Mis)handling null passwords

> A08 – Software and Data Integrity Failures

Integrity protection and MAC

- Integrity protection
- Message Authentication Code (MAC)
- Calculating MAC in Java

 *Lab – Calculating MAC in JCA*

Digital signatures

- Digital signature
- Digital signature with RSA
- ECC basics
- Digital signature with ECC

 *Lab – Digital signature with ECDSA in JCA*

Subresource integrity

- Importing JavaScript

 *Lab – Importing JavaScript*

 *Case study – The British Airways data breach*

Insecure deserialization


- Serialization and deserialization challenges
- Integrity – deserializing untrusted streams
- Using readObject
- Integrity – deserialization best practices
- Look ahead deserialization

Property Oriented Programming

- Property Oriented Programming (POP)


 *Lab – Creating a POP payload*

POP exploitation and best practices

 *Lab – Using the POP payload*

- Summary – POP best practices

- DoS with deserialization

 *Lab – Billion laughs with deserialization*

> **A10 – Server-Side Request Forgery**

SSRF

- Server-side Request Forgery (SSRF)

 *Case study – SSRF and the Capital One breach*

> **Wrap up**

Sources and further readings

- Software security sources and further reading
- Java resources