Web application security in Java and C#

CYDJvCsWeb4d | 4 days | On-site or online | Hands-on

Your application written in Java and C# 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 the discussed programming languages, and extended by core programming issues, discussing security pitfalls of the used frameworks.

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

So that nothing unexpected happens.

Nothing.

Skills and drills

31 LABS

19 CASE STUDIES

Audience

Java and C# developers working on Web applications

Group size

12 participants

Preparedness

General Java, C# and Web development

Outline

- Cyber security basics
- The OWASP Top Ten 2025
- Wrap up

Standards and references

OWASP, SEI CERT, CWE and Fortify Taxonomy

What you'll have learned

- 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 and C#
- Going beyond the low hanging fruits
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly in Java and C#
- Input validation approaches and principles

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

› 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 Citrix ShareFile

- Server-side Request Forgery (SSRF)

Case study – SSRF in Ivanti Connect Secure

A02 - Security Misconfiguration

- Configuration principles
- Cookie security
  - Cookie attributes
- 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 in Java
  - Preventing XXE in C#
  - *Lab – Prohibiting DTD*
  - *Case study – XXE vulnerability in SharePoint*

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
  - Dependency checking in C#
  - *Lab – Detecting vulnerable components*

Day 2

› The OWASP Top Ten 2025

A04 - Cryptographic Failures

- Cryptography for developers
  - Cryptography basics
  - Java Cryptographic Architecture (JCA) in brief
  - Crypto APIs in C#
  - Elementary algorithms
    - Hashing
    - Hashing basics
    - Hashing in Java
    - Hashing in C#
    - *Lab – Hashing*
    - Random number generation
    - Pseudo random number generators (PRNGs)

- Cryptographically secure PRNGs
- Weak and strong PRNGs in Java
- Weak PRNGs in C#
- Using random numbers in C#

Lab – Using random numbers

Case study – Equifax credit account freeze

- 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
  - Symmetric encryption in C#

Lab – Symmetric encryption

Case study – Padding oracle used in RCE against Citrix ShareFile

  - Asymmetric encryption
  - The RSA algorithm
  - RSA in Java
  - RSA in C#
  - Combining symmetric and asymmetric algorithms
  - Key exchange and agreement
  - Key exchange
  - Diffie-Hellman key agreement algorithm
  - Key exchange pitfalls and best practices

A05 - Injection

- 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
- 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*
  - *Case study – SQL injection leading to RCE in Ivanti Endpoint Manager*

Day 3

› The OWASP Top Ten 2025

A05 - Injection (continued)

- Input validation
- Unsafe reflection
  - Reflection without validation
  - *Lab – Unsafe reflection*
  - Are accessibility modifiers a security feature?
  - Accessibility modifiers – best practices
  - Overriding and accessibility modifiers in Java
- Code injection
  - OS command injection
    - *Lab – Command injection*
    - OS command injection best practices
    - Using Runtime.exec() in Java
    - Avoiding command injection with the right APIs in C#
    - *Lab – Command injection best practices*
    - *Case study – Command injection in VMware Aria*
- 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*
  - *Case study – XSS to RCE in Azure Service Fabric*
  - XSS protection best practices
    - Protection principles - escaping
    - XSS protection APIs in Java
    - XSS protection APIs in C#
    - Further XSS protection techniques in C#
    - *Lab – XSS fix / stored*

- *Lab – XSS fix / reflected*
- 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
  - 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*
- 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*
  - *Case study – Veeam missing authentication and cleartext password storage*

Day 4

› The OWASP Top Ten 2025

A07 - Authentication Failures (continued)

- Password policy
  - NIST authenticator requirements for memorized secrets
  - Password database migration

A08 - Software and Data Integrity Failures

- Integrity protection
  - Message Authentication Code (MAC)
    - Calculating MAC in Java
    - Calculating HMAC in C#
    - *Lab – Calculating MAC*
  - Digital signature
    - Digital signature with RSA
    - Elliptic Curve Cryptography
    - ECC basics
    - Digital signature with ECC
    - Digital signature in Java
    - Digital signature in C#
    - *Lab – Digital signature with ECDSA*
- 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
  - Look ahead deserialization in Java
  - Look ahead deserialization in C#
  - Property Oriented Programming (POP)
    - Creating a POP payload in C#
    - *Lab – Creating a POP payload*
    - *Lab – Using the POP payload*
    - *Case study – Deserialization RCE in Veeam*
    - Summary – POP best practices

A09 - Logging and Alerting Failures

- Logging and monitoring principles

- Insufficient logging

Case study – Plaintext passwords at Facebook

- Logging best practices
- Monitoring best practices

A10 - Mishandling of Exceptional Conditions

- Error and exception handling principles
- Error handling
  - Returning a misleading status code
  - Reachable assertion
  - Information exposure through error reporting
    - Information leakage via error pages

Case study – Information leakage via errors in Apache Superset

- Exception handling
  - In the catch block. And now what?
  - Catching NullPointerException
  - Empty catch block
  - Overly broad throws
  - Improper completing of the finally block
  - Throwing undeclared checked exceptions
  - Swallowed ThreadDeath
  - Throwing RuntimeException or Throwable
  - Catching and throwing SystemExceptions

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
  - Flooding
  - Resource exhaustion
  - Sustained client engagement
  - Denial of service problems in Java
  - Denial of service problems in C#
  - Infinite loop
  - Economic Denial of Sustainability (EDoS)
  - Algorithmic complexity issues
    - Regular expression denial of service (ReDoS)

- *Lab – ReDoS*
- Dealing with ReDoS

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop

And now what?

- Software security sources and further reading
- Java resources
- .NET and C# resources