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.

Cyber security skills and drills


31 LABS 16 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 2021
- 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 how cryptography supports security
- Learning how to use cryptographic APIs correctly in Java and C#
- 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
- Input validation approaches and principles
- Managing vulnerabilities in third party components

Table of contents

A02 - Cryptographic Failures

- Information exposure
  - Exposure through extracted data and aggregation
  - *Case study – Strava data exposure*
  - System information leakage
    - Leaking system information
  - Information exposure best practices
- Cryptography for developers
  - Cryptography basics
  - Java Cryptographic Architecture (JCA) in brief
  - Crypto APIs in C#
  - Elementary algorithms
    - Random number generation
    - Pseudo random number generators (PRNGs)
    - Cryptographically secure PRNGs
    - Weak and strong PRNGs in Java
    - Weak and strong PRNGs in C#
    - Using random numbers in C#
  - *Lab – Using random numbers*
    - *Case study – Equifax credit account freeze*
  - Hashing
    - Hashing basics
    - Hashing in Java
    - Hashing in C#
  - *Lab – Hashing*

Day 2

› The OWASP Top Ten 2021

A02 - Cryptographic Failures (continued)

- Cryptography for developers
  - 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
    - Symmetric encryption in C#

- Symmetric encryption in C# with streams
- *Lab – Symmetric encryption*
- Asymmetric encryption
- The RSA algorithm
- Using RSA – best practices
- RSA in Java
- RSA in C#
- Combining symmetric and asymmetric algorithms

A03 - 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 – Hacking Fortnite accounts*
    - SQL injection protection and ORM
- Parameter manipulation
  - CRLF injection
  - HTTP header manipulation
    - HTTP response splitting
    - Header checking in ASP.NET
  - HTTP parameter manipulation
    - HTTP parameter pollution
    - Value shadowing
    - Variable shadowing in Java
    - Value shadowing in C#
- Code injection
  - OS command injection
    - *Lab – Command injection*
    - OS command injection best practices
    - Using Runtime.exec() in Java
    - Using ProcessBuilder in Java
      - *Case study – Command injection via ping*
- 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 in Fortnite accounts*
- XSS protection best practices
  - Protection principles - escaping
  - XSS protection APIs in Java
  - XSS protection APIs in C#
  - Request validation in ASP.NET
  - Further XSS protection techniques in C#
- *Lab – XSS fix / stored*
- *Lab – XSS fix / reflected*
  - Client-side protection principles
  - Additional protection layers – defense in depth

Day 3

› The OWASP Top Ten 2021

A03 - Injection (continued)

- 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
  - Regular expression denial of service (ReDoS)
- *Lab – ReDoS*
  - Dealing with ReDoS
- Unsafe reflection
  - Reflection without validation
- *Lab – Unsafe reflection*

A04 - 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
    - Preflight request
    - Cross-Origin Resource Sharing (CORS)
  - Frame sandboxing
    - Cross-Frame Scripting (XFS) attacks
    - *Lab - Clickjacking*
    - Clickjacking beyond hijacking a click
    - Clickjacking protection best practices
    - *Lab – Using CSP to prevent clickjacking*

A05 - Security Misconfiguration

- Configuration principles
- Server misconfiguration
- ASP.NET configuration best practices
  - ASP.NET configuration
- 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*
      - *Case study – XXE vulnerability in SAP Store*
    - Preventing XXE in Java
    - Preventing XXE in C#
    - *Lab – Prohibiting DTD*

A06 - Vulnerable and Outdated Components

- 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*
  - DevOps, the CI / CD build process and Software Composition Analysis
  - Dependency checking in Java
  - Dependency checking in C#
  - *Lab – Detecting vulnerable components*

Day 4

The OWASP Top Ten 2021

A07 - Identification and Authentication Failures

- Authentication
  - Authentication basics
  - Multi-factor authentication (MFA)
  - *Case study – PayPal 2FA bypass*
- Session management
  - Session management essentials
  - Why do we protect session IDs – Session hijacking
  - Session fixation
- Password management
  - Inbound 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
      - *Case study – The Ashley Madison data breach*
      - *The ultimate crack*
      - *Exploitation and the lessons learned*

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*
    - Summary – POP best practices

A09 - Security Logging and Monitoring Failures

- Logging and monitoring principles
- Insufficient logging
- *Case study – Plaintext passwords at Facebook*
- 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)*
- Logging best practices
- Monitoring best practices

A10 - Server-side Request Forgery (SSRF)

- Server-side Request Forgery (SSRF)

Case study – SSRF and the Capital One breach

› 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