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

Audience

Java and C# developers working on Web applications

Group size

12 participants

Outline

- Cyber security basics
- The OWASP Top Ten
- Common software security weaknesses
- Wrap up

Preparedness

General Java, C# and Web development

44 labs

11 case studies

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 and C#
- Going beyond the low hanging fruits
- Managing vulnerabilities in third party components
- Identify vulnerabilities and their consequences
- Learn the security best practices in Java and C#
- Input validation approaches and principles

Table of contents

Day 1

    - Avoiding command injection with the right APIs in C#
    - *Lab – Command injection best practices in C#*
    - *Case study – Command injection via ping*
  - Script injection
- General protection best practices

A3 - Sensitive Data Exposure

- Information exposure
- Exposure through extracted data and aggregation
- *Case study – Strava data exposure*
- System information leakage
  - Leaking system information
- Information exposure best practices

A4 - XML External Entities (XXE)

- 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*

A10 - Insufficient Logging & Monitoring

- Logging and monitoring principles
- Insufficient logging
- *Case study – Plaintext passwords at Facebook*
- Logging best practices
- OWASP security logging library for Java
- C# logging best practices
- Monitoring best practices

Day 2

› The OWASP Top Ten

A2 - Broken Authentication

- Authentication basics
- Multi-factor authentication
- Authentication weaknesses - spoofing
- Spoofing on the Web

Case study – PayPal 2FA bypass

- 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
    - 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 database migration
    - (Mis)handling null passwords
- Session management
  - Session management essentials
  - Session ID best practices
  - Why do we protect session IDs – Session hijacking
  - Session fixation

A5 - Broken Access Control

- Access control basics
- Failure to restrict URL access
- Confused deputy
  - Insecure direct object reference (IDOR)

Lab – Insecure Direct Object Reference

  - 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*

A6 - Security Misconfiguration

- Configuration principles
- Configuration management
- Java related components – best practices

A7 - 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 in JSP
  - XSS protection APIs in C#
  - Request validation in ASP.NET
  - Further XSS protection techniques in C#
  - *Lab – XSS fix in C# / stored*
  - *Lab – XSS fix in C# / reflected*
  - Additional protection layers

A8 - Insecure Deserialization

- Serialization and deserialization challenges
- Deserializing untrusted streams
- Deserialization best practices
- Using ReadObject in Java
- Sealed objects in Java
- Look ahead deserialization in Java
- Property Oriented Programming (POP)
  - Creating payload in Java
  - Creating payload in C#
  - POP best practices

- *Lab – Creating a POP payload in C#*
- *Lab – Using the POP payload*

Day 3

› The OWASP Top Ten

A9 - Using Components with Known Vulnerabilities

- Using vulnerable components
- Untrusted functionality import
- Importing JavaScript

Lab – Importing JavaScript

Case study – The British Airways data breach

- Vulnerability management
  - Patch management
  - Vulnerability management
  - Vulnerability databases

Lab – Finding vulnerabilities in third-party components in Java

Lab – Finding vulnerabilities in third-party components in C#

Web application security beyond the Top Ten

- Client-side security
- Tabnabbing

Lab – Reverse tabnabbing

- Frame sandboxing
  - Cross-Frame Scripting (XFS) attack

Lab - Clickjacking

  - Clickjacking beyond hijacking a click
  - Clickjacking protection best practices

Lab – Using CSP to prevent clickjacking

› Common software security weaknesses

Input validation

- Input validation principles
  - Blacklists and whitelists
  - Data validation techniques

Lab – Input validation

  - What to validate – the attack surface
  - Where to validate – defense in depth

- How to validate – validation vs transformations
- Validation with regex
- Regular expression denial of service (ReDoS)
- *Lab – Regular expression denial of service (ReDoS)*
- Dealing with ReDoS

- Integer handling problems
  - Representing signed numbers
  - Integer visualization
  - Integer overflow
  - *Lab – Integer overflow*
  - Signed / unsigned confusion
  - Signed / unsigned confusion in Java
  - *Case study – The Stockholm Stock Exchange*
  - *Lab – Signed / unsigned confusion*
  - Integer truncation
  - Best practices
    - Upcasting
    - Precondition testing
    - Postcondition testing
    - Using big integer libraries
    - Integer handling in Java
    - *Lab – Integer handling in Java*
    - Integer handling in C#
    - *Lab – Checked arithmetics in C#*
- Unsafe reflection
  - Reflection without validation
  - *Lab – Unsafe reflection*

Day 4

› Common software security weaknesses

Security features

- Java platform security
  - The Java programming language and runtime environment
  - Type safety and security
  - Security features of the JRE
    - The ClassLoader and the BytecodeVerifier
  - Application-level access control in Java
    - Permissions and the Security Manager

    - Privilege best practices
  - Role-based access control
    - Java Authentication and Authorization Services (JAAS)
  - Protecting Java code and applications
    - Code signing
    - *Lab – Code signing and permissions*
- .NET platform security
  - Code Access Security
    - Code Access Security and Evidence
    - Application Domains and Permissions
    - The Stack Walk
    - *Lab – Code Access Security*
  - The transparency model
    - *Lab – The transparency model*
  - Role-based security
    - Principal and identity
    - Role-based permissions
    - Impersonation
    - *Lab – Role-based security*
  - Protecting .NET code and applications
    - Code signing

Time and state

- Race conditions
  - Race condition in object data members
    - Singleton member fields
    - *Lab – Singleton member fields*
  - File race condition
    - Time of check to time of usage – TOCTTOU
    - Insecure temporary file
  - Database race conditions
    - *Lab – Database race conditions*
  - Avoiding race conditions in Java
  - Avoiding race conditions in C#

Errors

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

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

Lab – Exception handling mess

- Catching and throwing SystemExceptions

Lab – Exception handling mess

Code quality

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

    Lab – Initialization cycles
  - Unreleased resource
    - The finalize() method – best practices in Java
- Object oriented programming pitfalls
  - Accessibility modifiers
    - Are accessibility modifiers a security feature?
    - Accessibility modifiers – best practices
    - Overriding and accessibility modifiers in Java
  - Inheritance and overriding
  - Mutability

    Lab – Mutable object
    - Readonly collections in C#
  - Cloning
- Serialization
  - Serializing sensitive data
  - Serialization best practices
  - DoS with deserialization

  Lab – Billion laughs

› Wrap up

Secure coding principles

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

And now what?

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