

# Web application security

## **CYDWeb3d | 3 days | On-site or online | Hands-on**

Your Web application written in any programming language 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<sup>31</sup>? 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 framework.

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

So that nothing unexpected happens.

Nothing.

---

## Cyber security skills and drills

### Audience

Web developers

### Group size

12 participants

### Outline

- Cyber security basics
- The OWASP Top Ten
- JSON security
- XML security
- Wrap up

### Preparedness

General Web development



23 labs



12 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 any programming language
- Going beyond the low hanging fruits
- Managing vulnerabilities in third party components
- Understanding security testing methodology and approaches
- Getting familiar with common security testing techniques and tools

---

# Table of contents

---

## Day 1

---

### › Cyber security basics

What is security?

Threat and risk

#### **Cyber security threat types**

#### **Consequences of insecure software**




- Constraints and the market
- The dark side

### › The OWASP Top Ten




#### **OWASP Top 10 – 2017**

#### **A1 – 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
  - Additional considerations
  - 🔗 *Lab – Using prepared statements*
  - 📄 *Case study – Hacking Fortnite accounts*
  - SQL injection and ORM
- Parameter manipulation
- CRLF injection
  - Log forging
  - Log forging – best practices
  - HTTP response splitting

- Code injection
  - OS command injection
    - OS command injection best practices
      -  *Case study – Shellshock*
      -  *Lab – Shellshock*
      -  *Case study – Command injection via ping*
- General protection 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*
-  *Lab – Prohibiting DTD expansion*

## › **JSON security**

JSON injection

Dangers of JSONP

JSON/JavaScript hijacking

Best practices

 *Case study – ReactJS vulnerability in HackerOne*

## › **XML security**

XML validation

### **XML injection**

 *Lab – XML injection*

- XPath injection
- Blind XPath injection

### **XML signature**











- XML signature wrapping

 *Case study – signature wrapping in single sign-on solutions*

## 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*
- User interface best practices
-  *Lab – On-line password brute forcing*
- 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](#)
    - Password length
    - Password hardening
    - Using passphrases
    -  *Lab – Applying a password policy*
      - Password change
      - Forgotten passwords
    -  *Lab – Password reset weakness*
    -  *Case study – The Ashley Madison data breach*
    -  *The dictionary attack*
    -  *The ultimate crack*
    -  *Exploitation and the lessons learned*
      - Password database migration
      - (Mis)handling passwords
  - Outbound password management
    - Hard coded passwords
    - Best practices
    -  *Lab – Hardcoded password*
      - Protecting sensitive information in memory
      - Challenges in protecting memory
- Session management

- Session management essentials
- Session ID best practices
- Why do we protect session IDs – Session hijacking
- Session fixation
- [Cross-site Request Forgery \(CSRF\)](#)
  - 🔗 *Lab – Cross-site Request Forgery*
    - CSRF best practices
    - CSRF defense in depth
  - 🔗 *Lab – CSRF protection with tokens*
- Using tokens
  - 🔗 *Lab – Using tokens*
- Cookie security
  - Cookie security best practices
  - Cookie attributes

### **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

### **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
- Server misconfiguration

---

## Day 3

---

### › The OWASP Top Ten

#### **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
- 📖 *Case study – XSS in Fortnite accounts*
- XSS protection best practices
  - Protection principles - escaping
- 🔗 *Lab – XSS fix / stored*
- 🔗 *Lab – XSS fix / reflected*
  - Additional protection layers
  - Client-side protection principles

#### **A8 – Insecure Deserialization**

- Serialization and deserialization challenges
- Deserializing untrusted streams
- Deserialization best practices
- 🔗 *Lab – Creating a POP payload*
- 🔗 *Lab – Using the POP payload*

#### **A9 – Using Components with Known Vulnerabilities**




- Using vulnerable components
- Untrusted functionality import
- Importing JavaScript
- 📖 *Case study – The British Airways data breach*
- Vulnerability management
  - Patch management
  - [Vulnerability management](#)
  - Vulnerability databases

#### **A10 – Insufficient Logging & Monitoring**

- Logging and monitoring principles

#### **[Web application security beyond the Top Ten](#)**

- Client-side security
- 🔗 *Lab – Client-side security*

- Same Origin Policy
  - Relaxing the Same Origin Policy
  - Relaxing with Cross-Origin Resource Sharing (CORS)
  - Simple request
  - Preflight request
  - 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*
- Some further best practices
  - HTML5 security best practices
  - CSS security best practices
  - Ajax security best practices

## **Content Security Policy**

- Directives in the HTTP response
- Browser support
- Resource control
- Source whitelists
- CSP best practices

## **Denial of service**

- Denial of Service
- Resource exhaustion
- Cash overflow
- Flooding
- Sustained client engagement
- Infinite loop
- Algorithm complexity issues
  - Regular expression denial of service (ReDoS)
    - Dealing with ReDoS
  - Hashtable collision
    - How hashtables work?
    - Hash collision in case of hashtables

## **Security testing techniques and tools**

- Code analysis



- Security aspects of code review
- Static Application Security Testing (SAST)
- Dynamic analysis
  - Security testing at runtime
  - [Penetration testing](#)
  - Stress testing
  - Dynamic analysis tools
    - Dynamic Application Security Testing (DAST)
    - Web vulnerability scanners
    - SQL injection tools
    - Proxy servers
  - Fuzzing

## › **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