

Extended Web application security

CYDWeb4d | 4 days | On-site or online | Hands-on

Your 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^{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.

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

So that nothing unexpected happens.

Nothing.

Cyber security skills and drills



29 LABS



15 CASE STUDIES

Audience

Web developers

Group size

12 participants

Preparedness

General Web development

Outline

- Cyber security basics
- The OWASP Top Ten 2021
- Wrap up

Standards and references

OWASP, CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Understanding how cryptography supports security
- 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
- Input validation approaches and principles
- Managing vulnerabilities in third party components

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 2021](#)

A01 – Broken Access Control

- Access control basics
- Confused deputy
 - Insecure direct object reference (IDOR)
 - Path traversal
 -  *Lab – Insecure Direct Object Reference*
 - Path traversal best practices
 - 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*
 - CSRF best practices
 -  *Lab – CSRF protection with tokens*

A02 – Cryptographic Failures

- Information exposure
 - Exposure through extracted data and aggregation
 -  *Case study – Strava data exposure*
- Cryptography for developers
 - Cryptography basics











- Elementary algorithms
 - Random number generation
 - Pseudo random number generators (PRNGs)
 - Cryptographically secure PRNGs
 - 🔗 *Lab – Using random numbers*
 - 📖 *Case study – Equifax credit account freeze*
 - Hashing
 - Hashing basics
 - 🔗 *Lab – Hashing*
- Confidentiality protection
 - Symmetric encryption
 - [Block ciphers](#)
 - Modes of operation
 - Modes of operation and IV – best practices
 - 🔗 *Lab – Symmetric encryption*
 - Asymmetric encryption
 - The RSA algorithm
 - Using RSA – best practices
 - Combining symmetric and asymmetric algorithms

Day 2

› [The OWASP Top Ten 2021](#)

A03 – Injection

- Input validation
 - Input validation principles
 - Denylists and allowlists
 - Data validation techniques
 - 🔗 *Lab – Input validation*
 - What to validate – the attack surface
 - Where to validate – defense in depth
 - When to validate – validation vs transformations
 - Output sanitization
 - Encoding challenges
 - Unicode challenges
 - 🔗 *Lab – Encoding challenges*
- 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
- NoSQL injection
 - NoSQL injection basics
- Parameter manipulation
 - CRLF injection
 - Mail command injection in IMAP, POP3 and SMTP
 - HTTP header manipulation
 - HTTP response splitting
- Code injection
 - OS command injection
 - OS command injection best practices
 -  *Case study – Shellshock*
 -  *Lab – Shellshock*
- Some other injection types
 - CSV injection
 - LDAP injection
 - Database cursor injection
- 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
 -  *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](#)

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
- Additional principles
 - Work factor
 - Compromise recording
- Client-side security
 - Same Origin Policy
 - Simple request
 - Preflight request
 - Cross-Origin Resource Sharing (CORS)
 - Relaxing the Same Origin Policy
 - Frame sandboxing
 - Cross-Frame Scripting (XFS) attacks
 -  *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
 - JSON security
 - JSON validation
 - JSON injection
 - Best practices
 -  *Case study – ReactJS vulnerability in HackerOne*


A05 – Security Misconfiguration

- Configuration principles
 - Server misconfiguration
 - Cookie security
 - Cookie security best practices
 - Cookie attributes
 - XML entities
 - DTD and the entities
 - Attribute blowup
 - 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](#)

A06 – Vulnerable and Outdated Components

- Using vulnerable components
- Assessing the environment
- Hardening
- Untrusted functionality import
- Vulnerability management
 - Patch management
 - [Vulnerability management](#)
 - Vulnerability databases
 - Vulnerability rating – CVSS
 - CVSS – Base Metric Group
 - CVSS – Supplemental Metric Group
 - CVSS – Threat Metric Group
 - CVSS – Environmental Metric Group
 - Bug bounty programs
 - [DevOps, the CI / CD build process and Software Composition Analysis](#)

A07 – Identification and Authentication Failures

- Authentication
 - Authentication basics
 - Multi-factor authentication (MFA)
 - Time-based One Time Passwords (TOTP)
-  [Case study – PayPal 2FA bypass](#)
- Session management

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

Day 4






› [The OWASP Top Ten 2021](#)

A07 – Identification and Authentication Failures (continued)


- 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 hardening
 - Using passphrases
 - 📖 *Case study – The Ashley Madison data breach*
 - 📖 *The ultimate crack*
 - 📖 *Exploitation and the lessons learned*
 - Password database migration
 - (Mis)handling null passwords
 - Outbound password management
 - Hard coded passwords
 - Best practices
 - 🔗 *Lab – Hardcoded password*
 - Protecting sensitive information in memory
 - Challenges in protecting memory

A08 – Software and Data Integrity Failures


- Integrity protection
 - Message Authentication Code (MAC)
 - 🔗 *Lab – Calculating MAC*
 - Digital signature
 - Digital signature with RSA
 - Elliptic Curve Cryptography
 - ECC basics

- Digital signature with ECC
 -  *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
 - Property Oriented Programming (POP)
 - Creating a POP payload
 -  *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*
- 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*

Web application security beyond the Top Ten

- Denial of service
 - Flooding
 - Resource exhaustion
 - Sustained client engagement
 - 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