

Security testing C# Web applications

CYDWebCsTst3d | 3 days | On-site or online | Hands-on

Your Web application written in C# is tested functionally, 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 -2³¹? Because that's what the bad guys will do – and the list is far from complete.

Testing for security needs a remarkable software security expertise and 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.

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.

A special focus is given to finding all discussed issues during testing, and an overview is provided on security testing methodology, techniques and tools.

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

So that nothing unexpected happens.

Nothing.

Cyber security skills and drills

Audience

C# developers and testers working on Web applications

Group size

12 participants

Outline

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

Preparedness

General C# and Web development, testing and QA



22 labs



10 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 C#
- Going beyond the low hanging fruits
- Understanding security testing methodology and approaches
- Getting familiar with common security testing techniques and tools
- Managing vulnerabilities in third party components
- Identify vulnerabilities and their consequences
- Learn the security best practices in C#
- Input validation approaches and principles

Table of contents

Day 1

› Cyber security basics

What is security?

Threat and risk







Cyber security threat types

Consequences of insecure software

› 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*
 - Testing for SQL injection
- Code injection
 - OS command injection
 -  *Lab – Command injection*
 - OS command injection best practices
 -  *Lab – Command injection best practices*
 -  *Case study – Command injection via ping*
 - Testing for command injection

- General protection best practices


A2 – Broken Authentication

- Authentication basics
- Multi-factor authentication
- Authentication weaknesses – spoofing
- [Spoofing on the Web](#)
- Testing for weak authentication


 *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 passwords
- Testing for password management issues

Day 2

› Security testing

Security testing vs functional testing

Manual and automated methods

Security testing methodology

- Security testing – goals and methodologies
- Overview of security testing processes
- Identifying and rating assets
 - Preparation
 - Identifying assets
 - Identifying the attack surface

- Assigning security requirements
- 🔗 *Lab – Identifying and rating assets*
- Threat modeling
 - SDL threat modeling
 - Mapping STRIDE to DFD
 - DFD example
 - Attack trees
 - Attack tree example
 - 🔗 *Lab – Crafting an attack tree*
 - Misuse cases
 - Misuse case examples
 - Risk analysis
 - 🔗 *Lab – Risk analysis*
- Security testing approaches
 - Reporting, recommendations, and review

› The OWASP Top Ten

A3 – Sensitive Data Exposure

- Information exposure
- Exposure through extracted data and aggregation

📖 *Case study – Strava data exposure*

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*

A5 – Broken Access Control

- Access control basics
- Failure to restrict URL access
- Testing for authorization issues
- 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*

- Testing for confused deputy weaknesses
- File upload
 - Unrestricted file upload
 - Good practices
-  *Lab – Unrestricted file upload*
 - Testing for file upload vulnerabilities

A6 – Security Misconfiguration

- Configuration principles
- Configuration management
- Testing for misconfiguration issues



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
 - Testing for XSS

Day 3

> The OWASP Top Ten

A8 – Insecure Deserialization

- Serialization and deserialization challenges
- Deserializing untrusted streams
- Deserialization best practices
- Testing for insecure deserialization
-  *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
 - [DevOps, the build process and CI / CD](#)

A10 – Insufficient Logging & Monitoring

- Logging and monitoring principles
- Insufficient logging

Case study – Plaintext passwords at Facebook

- Logging best practices

[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

› Security testing

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

› 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
 - Output sanitization
 - Encoding challenges
 - 🔗 *Lab – Encoding challenges*
 - Validation with regex
- Files and streams
 - Path traversal
 - Path traversal-related examples
 - Additional challenges in Windows
 - Path traversal best practices
 - Testing for path traversal

› 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
- Security testing resources