

Web application security in C#

CYDCsWeb3d | 3 days | On-site or online | Hands-on

Your Web application written in 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 -2³¹? 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 C#, and extended by core programming issues, discussing security pitfalls of the C# language and .NET 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

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 C# and Web development



32 labs



9 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
- 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*
- Code injection
 - OS command injection
 - 🔗 *Lab – Command injection*
 - OS command injection best practices
 - Avoiding command injection with the right APIs
 - 🔗 *Lab – Command injection best practices*
 - 📄 *Case study – Command injection via ping*
 - Script injection

- General protection best practices

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

Day 2




> The OWASP Top Ten

A2 – Broken Authentication





- 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*
- Cookie security

- Cookie security best practices
- Cookie attributes




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
-  *Lab – Prohibiting DTD*

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*

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
 - Request validation in ASP.NET
 - Further XSS protection techniques

 *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
- Property Oriented Programming (POP)
 - Creating payload
 - POP best practices

 *Lab – Creating a POP payload*

 *Lab – Using the POP payload*

A9 – Using Components with Known Vulnerabilities

- Using vulnerable components
- Assessing the environment
- Hardening
- 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*

Day 3

› **The OWASP Top Ten**

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
- Output sanitization
- Encoding challenges

 *Lab - Encoding challenges*

- 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

 *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 C#

 *Lab - Checked arithmetics*

- Unsafe reflection

- Reflection without validation

 *Lab - Unsafe reflection*

Code quality

- Data handling
 - Initialization and cleanup
 - Class initialization cycles
 - [🔗 Lab – Initialization cycles](#)
- Object oriented programming pitfalls
 - Inheritance and overriding
 - Mutability
 - [🔗 Lab – Mutable object](#)
 - Readonly collections

› 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
- .NET and C# resources