Web application security masterclass in C#

CYDCsWeb5d | 5 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 -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 C#, and extended by core programming issues, discussing security pitfalls of the C# language and ASP.NET.

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

So that nothing unexpected happens.

Nothing.

Skills and drills

39 LABS

22 CASE STUDIES

Audience

C# developers working on Web applications

Group size

12 participants

Preparedness

General C# and Web development

Outline

- Cyber security basics
- The OWASP Top Ten 2025
- Wrap up

Standards and references

OWASP, CWE and Fortify Taxonomy

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
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly 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 – the CIA triad
Consequences of insecure software
Constraints and the market

› The OWASP Top Ten 2025

A01 - Broken Access Control

- Access control basics
- Missing or improper authorization
- Failure to restrict URL access
- 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 – Remote takeover of Nexx garage doors and alarms*
  - *Lab – Horizontal authorization*
- File upload
  - Unrestricted file upload
  - Good practices
  - *Lab – Unrestricted file upload*
  - *Case study – File upload vulnerability in Citrix ShareFile*
- Open redirects and forwards
  - Open redirects and forwards – best practices
- Cross-site Request Forgery (CSRF)
  - *Lab – Cross-site Request Forgery*
  - CSRF best practices
  - CSRF defense in depth

- *Lab – CSRF protection with tokens*
- Insecure temporary file
- Server-side Request Forgery (SSRF)
  - *Case study – SSRF and the Capital One breach*

A02 - Security Misconfiguration

- Configuration principles
- Web security configuration issues
  - Content Security Policy
  - Fetch directives
  - Source allowlisting
  - Strict CSP: using nonces and hashes
  - Navigation and reporting directives
  - Document restrictions and sandboxing
  - CSP best practices
- Cookie security
  - Cookie attributes
- XML entities
  - 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*
    - Preventing XXE
    - *Lab – Prohibiting DTD*
    - *Case study – XXE vulnerability in SharePoint*

Day 2

› The OWASP Top Ten 2025

A02 - Security Misconfiguration (continued)

- .NET platform security
  - Role-based security
    - Principal and identity
    - Role-based permissions
    - Claims-based authorization in ASP.NET Core
    - *Lab – Role-based security*
- Secrets management
  - Hard coded passwords

- Best practices

Lab – Hardcoded password

A03 - Software Supply Chain Failures

- Using vulnerable components
- Assessing the environment
- Hardening
- Untrusted functionality import
- Supply chain security and the Software Bill of Materials (SBOM)
- SBOM examples

Case study – The Polyfill.io supply chain attack

- 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

Lab – Finding vulnerabilities in third-party components

  - Bug bounty programs
  - DevOps, the CI / CD build process and Software Composition Analysis
  - Dependency checking in C#

Lab – Detecting vulnerable components

- Use of dangerous and obsolete language elements
  - Using dangerous language elements
  - Using obsolete language elements

A04 - Cryptographic Failures

- Information exposure
  - Exposure through extracted data and aggregation

Case study – Strava data exposure

  - System information leakage
    - Leaking system information
  - Information exposure best practices
- Cryptography for developers
  - Cryptography basics
  - Crypto APIs in C#
  - Elementary algorithms
    - Hashing

- Hashing basics
- Hashing in C#
- *Lab – Hashing in C#*
- Random number generation
- Pseudo random number generators (PRNGs)
- Cryptographically secure PRNGs
- Weak PRNGs
- Using random numbers in C#
- *Lab – Using random numbers in C#*
- *Case study – Equifax credit account freeze*

- Confidentiality protection
  - Symmetric encryption
  - Block ciphers
  - Modes of operation
  - Modes of operation and IV – best practices
  - Symmetric encryption in C#
  - Symmetric encryption in C# with streams
  - *Lab – Symmetric encryption in C#*
  - *Case study – Padding oracle used in RCE against Citrix ShareFile*
  - Asymmetric encryption
  - The RSA algorithm
  - Using RSA – best practices
  - RSA in C#
  - *Case study – RSA attacks: Bleichenbacher, ROBOT, and Marvin*
  - Combining symmetric and asymmetric algorithms
  - Key exchange and agreement
  - Key exchange
  - Diffie-Hellman key agreement algorithm
  - Key exchange pitfalls and best practices

Day 3

› The OWASP Top Ten 2025

A05 - 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
- Validation with regex

- Unsafe reflection
  - Reflection without validation
  - *Lab – Unsafe reflection*
  - Are accessibility modifiers a security feature?
  - Accessibility modifiers – best practices
- Unsafe native code
  - Native code dependence
  - *Lab – Unsafe native code*
  - Best practices for dealing with native code
- 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 – SQL injection leading to RCE in Ivanti Endpoint Manager*
    - SQL injection protection and ORM
- 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 in Ruckus*
- Script injection
  - Server-side template injection (SSTI)
- Dangerous file inclusion
- 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 to RCE in Azure Service Fabric

- 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

- Client-side protection principles
- Additional protection layers – defense in depth

Day 4

› The OWASP Top Ten 2025

A06 - 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
- Client-side security
  - Same Origin Policy
    - Simple request
    - Preflight request
    - Cross-Origin Resource Sharing (CORS)
  - Frame sandboxing
    - Cross-Frame Scripting (XFS) attacks

Lab – Clickjacking

- Clickjacking beyond hijacking a click
- Clickjacking protection best practices

Lab – Using CSP to prevent clickjacking

A07 - Authentication Failures

- Authentication
  - Authentication basics
  - Multi-factor authentication (MFA)
  - *Case study – The InfinityGauntlet attack*
- Session handling
  - Session management essentials
  - Why do we protect session IDs – Session hijacking
  - Session fixation
  - Session ID best practices
- 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
  - *Lab – Using adaptive hash functions in C#*
  - *Case study – Veeam missing authentication and cleartext password storage*
- Password policy
  - NIST authenticator requirements for memorized secrets
  - Password hardening
  - Using passphrases
  - Password database migration
  - (Mis)handling null passwords

A08 - Software and Data Integrity Failures

- Integrity protection
  - Message Authentication Code (MAC)
    - Calculating HMAC in C#
    - *Lab – Calculating MAC in C#*
  - Digital signature
    - Digital signature with RSA
    - Elliptic Curve Cryptography
    - ECC basics
    - Digital signature with ECC
    - Digital signature in C#
    - *Lab – Digital signature with ECDSA in C#*
- Public Key Infrastructure (PKI)
  - Key management challenges

- Certificates
  - Certificates and PKI
  - Chain of trust

Day 5

› The OWASP Top Ten 2025

A08 - Software and Data Integrity Failures (continued)

- 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
  - Look ahead deserialization
  - Property Oriented Programming (POP)
    - Creating a POP payload
    - *Lab – Creating a POP payload*
    - *Lab – Using the POP payload*
    - *Case study – Deserialization RCE in Veeam*
    - Summary – POP best practices

A09 - Logging and Alerting Failures

- Logging and monitoring principles
- Insufficient logging
- *Case study – Plaintext passwords at Facebook*
- Logging best practices
- C# logging best practices
- Monitoring best practices
- Firewalls and Web Application Firewalls (WAF)
- Intrusion detection and prevention
- *Case study – The Marriott Starwood data breach*

A10 - Mishandling of Exceptional Conditions

- Error and exception handling principles
- Error handling
  - Returning a misleading status code

- Information exposure through error reporting
  - Information leakage via error pages
  - *Case study – Information leakage via errors in Apache Superset*
- Exception handling
  - In the catch block. And now what?
  - Catching NullReferenceException
  - Empty catch block
  - Catching and throwing SystemExceptions
  - *Lab – Exception handling mess*
- Control flow
  - Incorrect block delimitation
  - Dead code
  - Using if-then-else and switch defensively

Web application security beyond the Top Ten

- X01 - Lack of Application Resilience
  - Denial of service
  - Flooding
  - Resource exhaustion
  - Sustained client engagement
  - Denial of service problems in C#
  - Infinite loop
  - Economic Denial of Sustainability (EDoS)
  - Amplification
    - Some amplification examples
  - Algorithmic complexity issues
    - Regular expression denial of service (ReDoS)
    - *Lab – ReDoS*
    - Dealing with ReDoS
    - Hash table collision
    - How do hash tables work?
    - Hash collision against hash tables
- X02 - Memory Management Failures
  - 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

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop

And now what?

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