Web application security in C#

CELCsWeb | 1 year subscription | e-Learning | Online VM (Linux)

The course provides a comprehensive exploration of secure coding principles and practices tailored specifically for C# developers. Starting off from the foundations of cybersecurity, you will understand the consequences of insecure code by examining threats through the lens of the CIA triad.

In the main part of the material, you will systematically walk through the various vulnerabilities outlined in the OWASP Top Ten. As you progress through the modules investigating the intricacies of authentication and authorization, through realizing the practical aspects of cryptography, to tackling injection attacks, you will gain a deep understanding of both theoretical concepts and practical skills for securing C# web applications. Further subjects are aligned to some common software security weakness types, such as error handling, code quality or denial of service.

These modules go beyond just the theory. Not only do they identify vulnerabilities, show their consequences, and detail the best practices, but - through hands-on labs and real-world case studies - they offer practical experience in identifying, exploiting, and mitigating these security risks within C#-based web applications.

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

So that nothing unexpected happens in your code.

Nothing.

Note: This course content is available as an e-learning subscription. We reserve a period of 3 months to digest the foundational material, after which we activate shorter learning units on a monthly basis. This gives secure coding efforts an initial boost, and builds up sustained readiness over time. These learning units are indicated in red in the table of contents below.

Cyber security skills and drills

Foundational material

48 LABS

29 CASE STUDIES

Monthly learning units

2-3 LABS

CASE STUDY

Audience

C# developers working on Web applications

Outline

- Cyber security basics
- The OWASP Top Ten
- A01 - Broken Access Control
- A02 - Security Misconfiguration
- A03 - Software Supply Chain Failures
- A04 - Cryptographic Failures
- A05 - Injection
- A06 - Insecure Design
- A07 - Authentication Failures
- A08 - Software or Data Integrity Failures
- A09 - Logging and Alerting Failures
- A10 - Mishandling of Exceptional Conditions
- X02 - Memory Management Failures
- Wrap up

Preparedness

General C# and Web development

Standards and references

OWASP, CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Identify Web application vulnerabilities and their consequences
- Learn the security best practices in C#
- Input validation approaches and principles
- 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

Table of contents

File upload

- Unrestricted file upload
- Good practices

Lab – Unrestricted file upload

Case study – File upload vulnerability in Citrix ShareFile

Cross-site Request Forgery (CSRF)

Lab – Cross-site Request Forgery

Cross-site Request Forgery (CSRF) best practices

- CSRF best practices
- CSRF defense in depth
- CSRF (XSRF) protection in Angular

Lab – CSRF protection with tokens

SSRF

- Server-side Request Forgery (SSRF)

Case study – SSRF and the Capital One breach

› A02 - Security Misconfiguration

Misconfiguration and XML parsing

- Configuration principles
- XML Entities
- DTD and the entities
- Entity expansion

XML External Entity (XXE)

- File inclusion with external entities
- Server-Side Request Forgery with external entities

Lab – External entity attack

Case study – XXE vulnerability in SharePoint

XXE best practices

- Preventing XXE

Lab – Prohibiting DTD

Web security configuration issues

- Content Security Policy
- Resource control
- Fetch directives
- Source allowlisting

- Strict CSP: using nonces and hashes
- Navigation and reporting directives
- Document restrictions and sandboxing
- Browser support
- CSP best practices
- Cookie attributes

Secrets management

- Hard coded passwords
- Best practices

Lab – Hardcoded password

A03 - Software Supply Chain Failures

Vulnerable components and dependencies

- 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

Lab – Finding vulnerabilities in third-party components

- Bug bounty programs

Build security

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

Lab – Detecting vulnerable components

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

- Leaking system information

Cryptography for developers

- Cryptography basics
- Crypto APIs in C#

Hashing

- Hashing basics
- Hashing in C#

Lab – Hashing in C#

PRNG

- Random number generation
- Pseudo random number generators (PRNGs)
- Cryptographically secure PRNGs
- Using virtual random streams

Case study – Equifax credit account freeze

PRNG in C#

- Weak PRNGs
- Using random numbers in C#

Lab – Using random numbers in C#

Encryption

- Confidentiality protection
- Symmetric encryption
- Block ciphers
- Modes of operation
- Modes of operation and IV – best practices

Encryption in C#

- 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

A05 - Injection

Injection problems

- Injection principles
- Injection attacks

SQL injection

- SQL injection basics

Lab – SQL injection

SQL injection attack techniques

- Attack techniques
- Content-based blind SQL injection
- Time-based blind SQL injection

SQL injection best practices

- Input validation
- Parameterized queries

Lab – Using prepared statements

SQL injection additional considerations

- Database defense in depth

Case study – SQL injection leading to RCE in Ivanti Endpoint Manager

Beyond SQL injection - ORM and NoSQL (Unit 6)

- SQL injection protection and ORM
- Entity Framework and SQL injection
- NoSQL injection basics

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 – Shellshock

Lab - Shellshock

Case study – Command injection in Ruckus

HTML injection - Cross-site scripting (XSS)

- Cross-site scripting basics
- Persistent cross-site scripting
- Reflected cross-site scripting
- Client-side (DOM-based) cross-site scripting

XSS attacks

Lab – Stored XSS

Lab – Reflected XSS

Case study – XSS to RCE in Azure Service Fabric

XSS best practices 1

- Protection principles - escaping
- XSS protection APIs

Lab – XSS fix / stored

XSS best practices 2

- Further XSS protection techniques

Lab – XSS fix / reflected

- Additional protection layers – defense in depth
- XSS protection in Angular
- XSS protection in React

Advanced XSS (Unit 5)

- Additional XSS exploitation and protection

Lab – Stored XSS in attribute

Lab – XSS fix / stored (in HTML attributes)

- Client-side protection principles

Template injection (Unit 6)

- Script injection
- Server-side template injection (SSTI)
- Client-side template injection (CSTI) in Angular

- Client-side template injection (CSTI) in React

Input validation principles 1 (Unit 1)

- Input validation principles
- Denylists and allowlists
- Data validation techniques

Lab – Input validation

Input validation principles 2 (Unit 1)

- What to validate – the attack surface
- Where to validate – defense in depth
- When to validate – validation vs transformations

Input validation principles 3 (Unit 1)

- Output sanitization
- Encoding challenges
- Unicode challenges

Lab – Encoding challenges

Lab – Dealing with Unicode homoglyph attacks

- Validation with regex

Path traversal and file validation (Unit 3)

- Path traversal

Lab – Path traversal

- Path traversal-related examples
- Additional challenges in Windows

Case study – File spoofing in WinRAR

- Virtual resources
- Path traversal best practices

Lab – Path canonicalization

Unsafe reflection (Unit 4)

- Reflection without validation

Lab – Unsafe reflection

- Are accessibility modifiers a security feature?
- Accessibility modifiers – best practices

Unsafe native code (Unit 4)

- Native code dependence

Lab – Unsafe native code

- Best practices for dealing with native code

› A06 – Insecure Design

Insecure design

- The STRIDE model of threats
- Secure design principles of Saltzer and Schroeder

Insecure design - Saltzer and Schroeder 1

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design

Insecure design - Saltzer and Schroeder 2

- 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)

Clickjacking

- Frame sandboxing
- Cross-Frame Scripting (XFS) attacks

Lab – Clickjacking

- Clickjacking beyond hijacking a click

Anti-clickjacking best practices

- 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

- Authentication weaknesses

Session security

- Session management essentials
- Why do we protect session IDs – Session hijacking
- Session fixation
- Session invalidation
- Session ID best practices
- Session handling security considerations in single-page applications

Password management

- Storing account passwords
- Password in transit

Lab – Is just hashing passwords enough?

Password storage

- 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 storage - a case study

Case study – The Ashley Madison data breach

The dictionary attack

The ultimate crack

Exploitation and the lessons learned

Additional password management challenges

- Password database migration
- (Mis)handling null passwords

Password auditing (Unit 8)

- Using password cracking tools

Lab – Password audit with John the Ripper

- Password recovery issues
- Password recovery best practices

Lab – Password reset weakness

Case study – GitLab account takeover

Protecting secrets in memory (Unit 8)

- Challenges in protecting memory
- *Case study – Microsoft secret key theft via dump files*
- Storing sensitive data in memory
- *Case study – KeePass password leakage via strings*

A08 - Software or Data Integrity Failures

Integrity protection and MAC

- Integrity protection
- Message Authentication Code (MAC)
- Calculating HMAC in C#
- *Lab – Calculating MAC in C#*

Digital signatures

- Digital signature
- Digital signature with RSA
- ECC basics
- Digital signature with ECC
- Digital signature in C#
- *Lab – Digital signature with ECDSA in C#*

Subresource integrity

- Importing JavaScript
- *Lab – Importing JavaScript*
- Subresource integrity in Angular
- *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

- Property Oriented Programming (POP)
- Creating a POP payload
- *Lab – Creating a POP payload*

POP exploitation and best practices

- *Lab – Using the POP payload*

- Summary – POP best practices

Case study – Deserialization RCE in Veeam

A09 - Logging and Alerting Failures

Logging

- Logging and monitoring principles
- Insufficient logging

Case study – Plaintext passwords at Facebook

- Logging best practices

Log forging (Unit 7)

- Newline injection
- Log forging
- Web log forging

Lab – Log forging

- Log forging – best practices

Monitoring (Unit 7)

- 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

Principles

- Error and exception handling principles
- Information exposure through error reporting
- Information leakage via error pages
- Returning a misleading status code

Exception handling

- In the catch block. And now what?
- Catching NullReferenceException
- Empty catch block
- Catching and throwing SystemExceptions

Lab – Exception handling mess

Control flow and error handling

- Incorrect block delimitation

- Dead code
- Using if-then-else and switch defensively

› X01 - Lack of Application Resilience (Unit 9)

Denial of service

- Flooding
- Resource exhaustion

Sustained client engagement

- Infinite loop
- Denial of service problems in C#
- Economic Denial of Sustainability (EDoS)

Amplification

- Some amplification examples

Algorithmic complexity issues

- Regular expression denial of service (ReDoS)

Lab – ReDoS

- Dealing with ReDoS

› X02 - Memory Management Failures

Integer handling problems (Unit 2)

- Representing signed numbers
- Integer visualization
- Integer overflow

Lab – Integer overflow

Integer handling problems 2 (Unit 2)

- Signed / unsigned confusion

Case study – The Stockholm Stock Exchange

Lab – Signed / unsigned confusion

- Integer truncation

Integer best practices (Unit 2)

- Upcasting
- Precondition testing
- Postcondition testing
- Using big integer libraries

Integer best practices in C# (Unit 2)

- Integer handling in C#

Lab – Checked arithmetics

Other numeric problems (Unit 2)

- Division by zero

Resource leakage (Unit 4)

- Unreleased resource
- Unreleased resource – Database
- Unreleased resource – Synchronization
- Unreleased resource – Various

› Wrap up

Software security principles

- Principles of robust programming by Matt Bishop

Sources and further readings

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