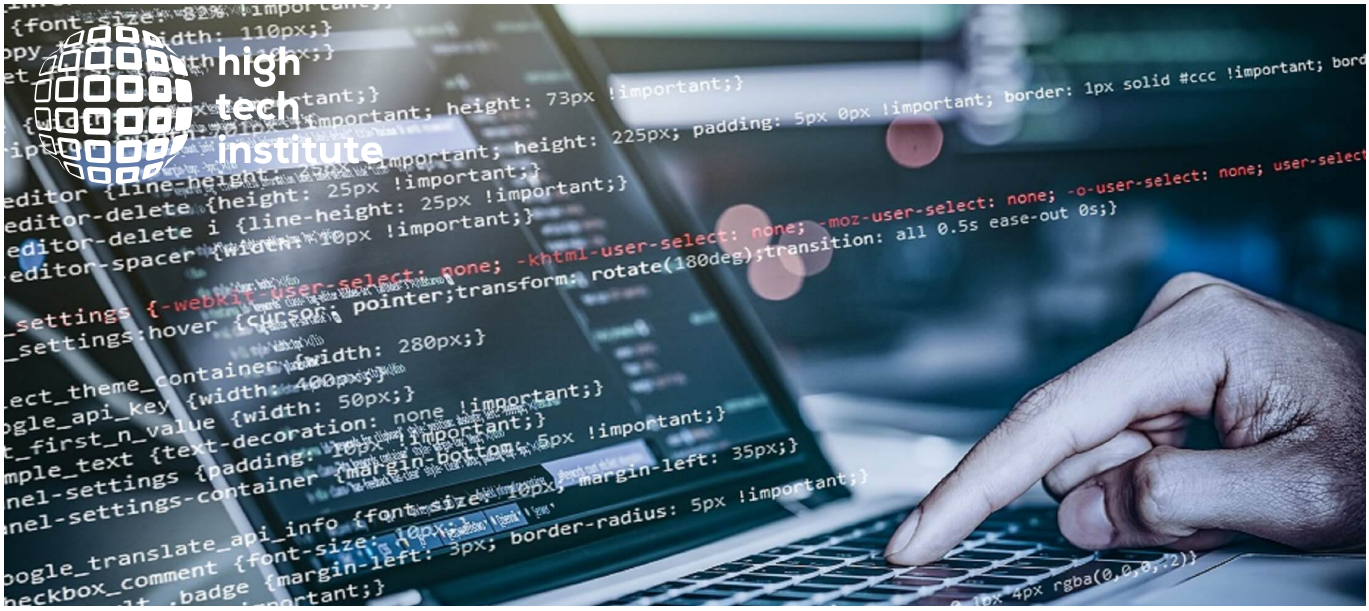
TRAINING BROCHURE

Web application security in C# training



Provisional reservation

Book now

high tech institute

Web application security in C#

Price: On request

Duration: 3 consecutive days

Contact: training@hightechinstitute.nl, +31 85 401 3600

Intro

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 -2^32? 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 hand 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.

PRACTICAL INFO

- *The 'Web application security in C#' training can be organized as in-company training.*
- *If on-site training is not feasible, we can discuss providing a live, interactive online (virtual) or hybrid training. The standard program with 3-day content can also be delivered in 5 half days (from Monday to Friday).*
- *Curious about how to quantify the return on investment (ROI) of secure coding trainings? Check out this article.*

Objective

- Understanding Web application security issues;
- Detailed analysis of the OWASP Top Ten elements;
- Going beyond the low hanging fruits;
- Explain approaches in handling security challenges in code;
- Identify security vulnerabilities and their consequences;
- Learn the best practices in how to avoid these mistakes.

Target audience

This course is intended for C# developers working on Web applications.

Preparedness: General C# and Web development.

Certification

After attending this training participants receive a High Tech Institute certificate.

Trainers

Ernő Jeges MSc
Balazs Kiss

** Prices are subject to change. Price correction will be applied at the end of the year.*

Keep me posted

Program

Day 1

Security basics

What is security?

Threat and risk

Types of threats against computer systems

Consequences of insecure software

Constraints and the market

Bugs, vulnerabilities and exploits

Categorization of bugs

- Seven pernicious kingdoms
- Common Weakness Enumeration (CWE)
- CWE/SANS Top 25 Most Dangerous Software Errors
- Vulnerabilities in the environment and the dependencies

The OWASP Top Ten

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
- Output encoding
- Parameterized queries
- Other best practices
- Lab – Using prepared statements
- Case study – Hacking Fortnite accounts
- Code injection
- Command injection
- Lab – Command injection
- Command injection best practices
- Lab – Command injection best practices
- Case study – Command injection
- Script injection
- Injection best practices
- Input validation
- Output sanitization
- Encoding and escaping the output
- Encoding challenges

A2 - Broken Authentication

- Authentication basics
- Authentication weaknesses
- Spoofing on the Web
- Case study – PayPal two factor authentication bypass
- Password management
- Inbound password management
- Storing account passwords
- Plaintext passwords at Facebook
- Lab – Why just hashing passwords is not enough?
- Dictionary attacks and brute forcing
- Salting

- Adaptive hash functions for password storage
- Password in transit
- Password policy
- Weak and strong passwords
- Using passphrases
- Lab – Applying a password policy
- The Ashley Madison data breach
- The dictionary attack
- The ultimate attack
- Exploitation of the results and the lessons learnt
- Outbound password management
- Hard coded passwords
- Lab – Hardcoded password
- Password in configuration file
- Protecting sensitive information in memory
- Challenges in protecting memory
- Storing sensitive data in memory
- Lab – Storing sensitive data in memory with SecureString
- Session management
- Session management essentials
- Why do we protect session IDs – Session hijacking
- Session ID best practices
- Insufficient session expiration
- Session fixation
- Cross-site Request Forgery (CSRF)
- Lab – Cross-site Request Forgery
- CSRF best practices
- Lab – CSRF protection with tokens
- Cookie security
- Cookie security best practices
- Cookie parameters

Day 2

The OWASP Top Ten

A3 - Sensitive Data Exposure

- Information exposure
- Exposure through extracted data and aggregation
- System information leakage
- Leaking system information
- Relying on accessibility modifiers
- Lab – Inappropriate protection by accessibility modifier
- Information exposure best practices

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 attack against some popular services
- Preventing XXE

A5 - Broken Access Control

- Access control basics
- Missing or improper authorization
- 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
- File upload
- Unrestricted file upload

- Best practices
- Lab – Unrestricted file upload

A6 - Security Misconfiguration

- Configuration principles
- Server misconfiguration
- Configuration management
- NET and IIS configuration best practices
- NET configuration
- IIS configuration

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 – Yahoo mail stored XSS
- Lab – Reflected and stored XSS
- XSS protection best practices
- Protection principles - escaping
- Additional protection layers
- Client-side protection principles
- XSS protection APIs
- Request validation in ASP.NET
- Further XSS protection techniques
- Lab – XSS best practices

A8 - Insecure Deserialization

- Serialization and deserialization challenges
- Deserializing untrusted streams
- Deserializing best practices
- Property Oriented Programming (POP)
- POP best practices
- Lab – Creating and using a POP payload

A9 - Using Components with Known Vulnerabilities

- Using vulnerable components
- Assessing the environment
- Hardening
- Importing functionality from untrusted sources
- Case study – The British Airways data breach
- Vulnerability management
- Patch management
- Vulnerability databases and scanning tools
- Vulnerability rating – CVSS
- Lab – Finding vulnerabilities of used components
- The build process and CI / CD
- Dependency checking in Cake

A10 - Insufficient Logging & Monitoring

- Logging and monitoring principles
- Logging
- Insufficient logging
- Logging best practices
- Monitoring
- Monitoring best practices

Web application security beyond the Top Ten

- Client-side security
- Same Origin Policy
- Simple request
- Preflight request

- Bypassing the Same Origin Policy
- Cross-origin resource sharing
- Frame sandboxing
- Clickjacking
- Clickjacking protection best practices
- Lab - Clickjacking
- JavaScript hijacking

Day 3

Common software security weaknesses

Input validation

- Input validation principles
- Blacklists and whitelists
- Validation with regex
- What to validate – the attack surface
- When to validate – validation vs transformations
- Where to validate – defense in depth
- Server-side vs. client-side validation
- Integer handling
- Representing signed numbers
- Integer visualization
- Integer problems
- Integer overflow
- Lab – Integer overflow
- Signed / unsigned confusion
- Lab – Signed / unsigned confusion
- Integer truncation
- Best practices
- Upcasting
- Precondition testing
- Postcondition testing
- Using big integer libraries
- Integer handling in C#
- Lab – Checked arithmetics
- Other numeric problems
- Division by zero
- Unsafe reflection
- Reflection without validation
- Lab – Unsafe reflection
- Unsafe native code
- Native code dependence
- Lab – Unsafe native code
- Some other input validation problems

Security features

- .NET platform security
- Code Access Security
- Evidences
- Permissions
- The Stack Walk
- Lab – Code Access Security
- The transparency model
- Best practices
- Lab – Experimenting with the transparency model
- Role-based security
- Principal and identity
- Role-based permissions
- Impersonation
- Lab – Role-based security
- Protecting .NET code and applications
- Code signing

Errors

- Error and exception handling principles

- Error handling
- Returning a misleading status code
- Information exposure through error reporting
- Missing custom error pages
- Exception handling
- In the catch block. And now what?
- Empty catch block
- Best practices for catch blocks
- Overly broad throws
- Catching NULL pointer exceptions
- Exception handling in C#
- Lab – Exception handling mess

Code quality

- Data
- Arrays and ToString()
- Initialization and cleanup
- Uninitialized variable
- Class initialization cycles
- Lab – Initialization cycles
- Unreleased resource
- Object oriented programming pitfalls
- Accessibility modifiers
- Inheritance and overriding
- Implementing Equals()
- Mutability
- Readonly collections
- Lab – Mutable object
- Cloning
- Cloning sensitive classes – object hijacking
- Object hijacking – best practices
- Serialization
- Serializing sensitive data
- Serialization best practices
- Lab – Serializing sensitive data
- DoS with deserialization
- Memory leaks during serialization

Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop
- Secure design principles of Saltzer and Schröder
- Some more principles

And now what?

- Further sources and readings
- .NET and C# resources

Further labs and challenges to do

Methods

A blended learning journey: live instructor-led training with lab exercises in a top-notch e-learning system. You will keep access to the e-learning system 3-months post-training to revisit the lab exercises and material.

Platform: Windows.
Labs: Hands-on.

Trainers

Ernő Jeges MSc
Balazs Kiss

More information

Video with trainer Balazs Kiss

In this 8-minute video, trainer Balazs Kiss elaborates on software security.

Watch video