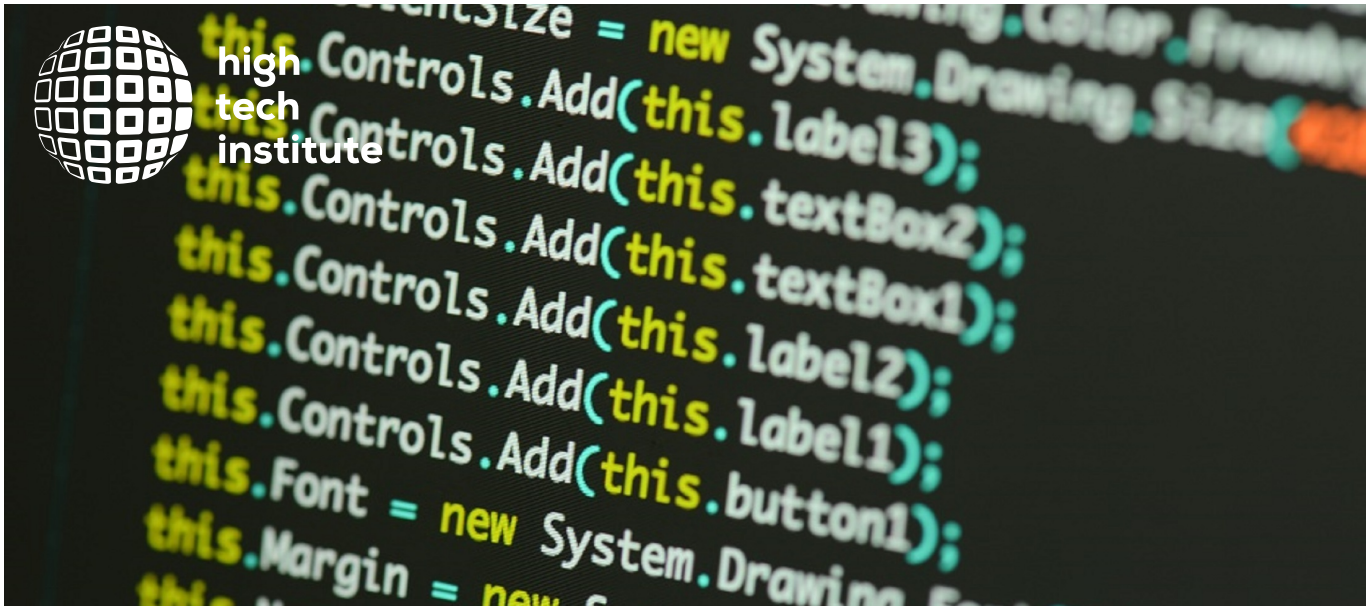
TRAINING BROCHURE

Desktop application security in C# training



Provisional reservation

Book now

high tech institute

systems & software academy

Desktop application security in C#

Price: On request

Duration: 3 consecutive days

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

Intro

Your 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.

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 'Desktop 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).*

Objective

- Explain approaches in handling security challenges in code;
- Identify security vulnerabilities and their consequence;
- Learn the best practices in how to avoid these mistakes.

Intended for

This course is intended for C# developers working on desktop applications. Preparedness: General C# development.

Program

Day 1

Security basics

What is security?

Threat and risk

Certification

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

Trainers

Ernő Jeges MSc

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

Keep me posted

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

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

Injection

- Injection principles
- Injection attacks
- CRLF injection
- Log forging
- Lab – Log forging
- Log forging – best practices
- 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

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

Data structures

- Data structure sentinels
- Containers

- Container error
- Associative containers
- Iterators

Files and streams

- Path traversal
- Path traversal-related examples
- Additional challenges in Windows
- Path traversal best practices
- Lab – Path traversal
- Virtual resources

Unsafe reflection

- Reflection without validation
- Lab – Unsafe reflection

Unsafe native code

- Native code dependence
- Lab – Unsafe native code

Some other input validation problems

Using vulnerable components

Assessing the environment

Hardening

Importing functionality from untrusted sources

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

Day 2

Security features

Authentication

- Authentication basics
- Authentication weaknesses
- Case study – PayPal two factor authentication bypass
- User interface best practices
- 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

Authorization

- Access control basics
- Missing or improper authorization
- Access control in databases
- Lab – Database access control
- Privileges and permissions
- Permission manipulation
- Incorrect use of privileged APIs
- Permission best practices
- Principle of least privilege
- Principle of separation of privileges
- Permission granting
- Privilege dropping
- Handling of insufficient privileges

.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

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

UI security

- UI security principles
- Sensitive information in the user interface
- Lab – Extracting password from the UI
- Misinterpretation of UI features or actions
- Insufficient UI feedback
- Relying on hidden or disabled UI element
- Lab – Hidden or disabled UI element
- Insufficient anti-automation

Day 3

Common software security weaknesses

Time and state

- Thread management best practices

- Thread management best practices in C#
- Race conditions
- Race condition in object data members
- Lab – Singleton member fields
- File race condition
- Time-of-check-to-time-of-usage (TOCTTOU)
- Lab - TOCTTOU
- Insecure temporary file
- Database race conditions
- Avoiding race conditions in C#
- Mutual exclusion and locking
- Deadlocks
- Lab – Locking
- Synchronization and thread safety
- Synchronization and thread safety in C#

Errors

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

Denial of service

- Denial of Service
- Resource exhaustion
- Cash overflow
- Flooding
- Sustained client engagement
- Denial of service problems in C#
- Infinite loop
- Lab – Resource exhausting
- Amplification
- Network amplification
- Amplification in databases
- Other amplification examples
- Algorithm complexity issues
- Regular expression denial of service (ReDoS)
- Lab – ReDos

- Hashtable collision
- How hashtables work?
- Hash collision in case of hashtables
- Hashtable collision in C#

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

Platform: Windows.
Labs: Hands-on.

Trainers

Ernő Jeges MSc