Desktop Application Security in C#

Category: Networking & Security | **Vendor:** Various

Duration: 24.00 hours (3 days) **19.5 CPD Hours** **Rating:** ⭐ 4.6 (5,878 reviews)

Course Information

Delivery Format: Instructor Led - Online

Course Overview

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 -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.

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.

About This Course

- Cyber security basics
- Input validation
- Security features
- Time and state
- Errors
- Cryptography for developers
- Common software security weaknesses
- Using vulnerable components
- Wrap up

Who Should Attend

General C# development

Prerequisites & Entry Requirements

General Prerequisites:

General C# development

Learning Outcomes

Upon successful completion of this course, participants will be able to:

- Getting familiar with essential cyber security concepts
- Identify vulnerabilities and their consequences
- Learn the security best practices in C#
- Input validation approaches and principles
- Understanding how cryptography can support appplication security
- Learning how to use cryptographic APIs correctly in C#
- Managing vulnerabilities in third party components

Detailed Course Outline

DAY 1

Cyber security basics

- What is security?
- Threat and risk
- Cyber security threat types
- Consequences of insecure software

- Constraints and the market
- The dark side
- Categorization of bugs

- The Seven Pernicious Kingdoms
- Common Weakness Enumeration (CWE)
- CWE Top 25 Most Dangerous Software Errors

Input validation

- Input validation principles

- Blacklists and whitelists
- Data validation techniques
- What to validate – the attack surface
- Where to validate – defense in depth
- How to validate – validation vs transformations
- Output sanitization
- Encoding challenges
- Validation with regex
- Injection

- Injection principles
- Injection attacks
- 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
- 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
- Files and streams

- Path traversal
- Path traversal-related examples
- Lab – Path traversal
- Additional challenges in Windows
- Path traversal best practices
- Unsafe reflection

- Reflection without validation
- Lab – Unsafe reflection
- Unsafe native code

- Native code dependence
- Lab – Unsafe native code
- Best practices for dealing with native code

DAY 2

Security features

- Authentication

- Authentication basics
- Multi-factor authentication
- Authentication weaknesses – spoofing
- Case study – PayPal 2FA bypass
- User interface best practices
- 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
- Password length
- Password hardening
- Using passphrases
- Lab – Applying a password policy
- Case study – The Ashley Madison data breach

- The dictionary attack
- The ultimate crack

- Exploitation and the lessons learned
- Password database migration
- Outbound password management

- Hard coded passwords
- Best practices
- Lab – Hardcoded password
- Protecting sensitive information in memory

- Challenges in protecting memory
- Storing sensitive data in memory
- Sensitive data in memory
- Authorization

- Access control basics
- Information exposure

- Exposure through extracted data and aggregation
- Case study – Strava data exposure
- System information leakage

- Leaking system information
- Information exposure best practices
- .NET platform security

- Code Access Security

- Code Access Security and Evidence
- Application Domains and Permissions
- The Stack Walk
- Lab – Code Access Security
- The transparency model

- Lab – The transparency model
- Role-based security

- Principal and identity
- Role-based permissions
- Impersonation
- Lab – Role-based security
- Protecting .NET code and applications

- Code signing
- 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
- Insufficient anti-automation

Time and state

- Race conditions

- Race condition in object data members

- Lab – Singleton member fields
- File race condition

- Time of check to time of usage – TOCTTOU
- Insecure temporary file

- Database race conditions

- Avoiding race conditions 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?

- Catching NullReferenceException

- Empty catch block

- Catching and throwing SystemExceptions

- Lab – Exception handling mess

DAY 3

Cryptography for developers

- Cryptography basics

- Crypto APIs in C#

- Elementary algorithms

- Random number generation

- Pseudo random number generators (PRNGs)

- Cryptographically strong PRNGs

- Weak and strong PRNGs
- Using random numbers in C#
- Case study – Equifax credit account freeze
- Lab – Using random numbers in C#
- Hashing

- Hashing basics
- Common hashing mistakes
- Hashing in C#
- Lab – Hashing in C#
- 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
- ProtectedMemory and ProtectedData
- Lab – Symmetric encryption in C#
- Asymmetric encryption

- The RSA algorithm

- Using RSA – best practices
- RSA in C#
- Lab – Using RSA in C#
- Elliptic Curve Cryptography

- The ECC algorithm
- Using ECC – best practices

- Combining symmetric and asymmetric algorithms

- Integrity protection

- Message Authentication Code (MAC)

- Calculating HMAC in C#

- Lab – Calculating MAC in C#

- Digital signature

- Digital signature with RSA

- Digital signature with ECC

- Digital signature in C#

- Lab – Digital signature in C#

- Public Key Infrastructure (PKI)

- Some further key management challenges

- Certificates

- Chain of trust

- Certificate management – best practices

Common software security weaknesses

- Code quality

- Data handling

- Initialization and cleanup

- Class initialization cycles

- Lab – Initialization cycles

- Unreleased resource

- Object oriented programming pitfalls

- Accessibility modifiers

- Are accessibility modifiers a security feature?

- Accessibility modifiers – best practices

- Inheritance and overriding

- Mutability

- Lab – Mutable object

- Readonly collections

- Denial of service

- Denial of Service

- Resource exhaustion

- Cash overflow

- Flooding

- Algorithm complexity issues

- Regular expression denial of service (ReDoS)

- Lab – Regular expression denial of service (ReDoS)

- Dealing with ReDoS

- Hashtable collision

- How hashtables work?

- Hash collision in case of hashtables

- Hashtable collision in C#

Using vulnerable components

- Vulnerability management

- Patch management

- Vulnerability databases

- Lab – Finding vulnerabilities in third-party components

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

Skills You Will Learn:

- Getting familiar with essential cyber security concepts
- Identify vulnerabilities and their consequences
- Learn the security best practices in C#
- Input validation approaches and principles
- Understanding how cryptography can support appplication security
- Learning how to use cryptographic APIs correctly in C#
- Managing vulnerabilities in third party components

Frequently Asked Questions

Q: What delivery options are available for Desktop Application Security in C#?

We offer multiple delivery formats:

- Live Instructor-Led Classroom Online (Virtual/Live Online)
- Traditional Instructor-Led Classroom Training (ILT)
- On-site delivery at your offices anywhere in United Kingdom
- Private dedicated courses customized for your team

Q: How many CPD hours does this course provide?

The 3-day Desktop Application Security in C# course provides up to 19.5 CPD hours of structured learning. CPD certificates can be provided upon request.

Q: What is the duration of the Desktop Application Security in C# training?

The training takes place over 3 day(s), with each day lasting approximately 24.00 hours including breaks for lunch and refreshments.

Q: Do you provide corporate training for Desktop Application Security in C#?

Yes, we provide corporate training, dedicated training, and closed classes for Desktop Application Security in C#. Training can take place anywhere in United Kingdom including London, Manchester, Birmingham, Edinburgh, or live online allowing teams from across United Kingdom or internationally to attend.

Q: What related terms do people search for?

Popular related searches include: Desktop Application Security in C#, Desktop Application Security, DASEC-C#

Q: Why choose Nexus Human for Desktop Application Security in C#?

Nexus Human is recognized as one of the leading training providers. Our trainers have won multiple awards including:

- Small Firms Best Trainer Award
- National Training Partner of the Year (Ireland) - Multiple Years
- Global Top 30 Instructor Awards (2012, 2019, 2021)
- Tech Excellence Award Nominations
- Learning Performance Institute (LPI) External Training Provider Sponsor 2024

Q: Are there any discount codes available?

Yes! Use discount code **PENPAL5** when booking your Desktop Application Security in C# training. Please note that only one discount code can be used per booking and cannot be combined with other special offers.

Nexus Human

Professional Training & Development

Email: info@nexushuman.com

Website: www.nexushuman.com

Phone: +353 1 889 8700 (Ireland) | +44 20 8142 8961 (UK)

© 2026 Nexus Human. All rights reserved. This brochure was generated on 19/10/2026.