

**Desktop Application Security in C#**

**Course Number:** SEC-148
**Duration:** 3 days

**Overview**

This Desktop Application Security in C# training teaches developers how to prevent common security issues in C# applications. Attendees go beyond core programming issues, exploring secure code pitfalls of the C# language and the .NET framework.

**Note:** To ensure ample one-on-one engagement with the instructor, this class is capped at 12 people, overriding Accelebrate’s default cap of 15.

**Prerequisites**

All secure coding students should have general C# and web application development experience.

**Materials**

All attendees receive comprehensive courseware.

**Software Needed on Each Student PC**

Attendees will not need to install any software on their computer for this class. The class will be conducted in a remote environment that Accelebrate will provide; students will only need a local computer with a web browser and a stable Internet connection. Any recent version of Microsoft Edge, Mozilla Firefox, or Google Chrome will be fine.

**Objectives**

* Understand essential cyber security concepts
* Use input validation approaches and principles
* Identify vulnerabilities and their consequences
* Implement the security best practices in C#
* Understand how cryptography supports security
* Use cryptographic APIs correctly in C#
* Manage vulnerabilities in third-party components

**Outline**

* Introduction
* Cyber security basics
	+ What is security?
	+ Threat and risk
	+ Cyber security threat types – the CIA triad
	+ Cyber security threat types – the STRIDE model
	+ Consequences of insecure software
* Input Validation
	+ Input validation principles
	+ Denylists and allowlists
	+ What to validate – the attack surface
	+ Where to validate – defense in depth
	+ When to validate – validation vs transformations
	+ Validation with regex
	+ Injection
	+ Integer handling problems
		- Representing signed numbers
		- Integer visualization
		- Integer overflow
		- Signed/unsigned confusion
		- The Stockholm Stock Exchange
		- Integer truncation
		- Best practices
	+ Files and streams
		- Path traversal
		- Additional challenges in Windows
		- Virtual resources
		- Path traversal best practices
		- Path canonicalization
	+ Unsafe reflection
		- Reflection without validation
	+ Unsafe native code
		- Native code dependence
		- Unsafe native code
		- Best practices for dealing with native code
* Security Features
	+ Authentication
		- Authentication basics
		- Multi-factor authentication
		- Authentication weaknesses
		- Password management
	+ Information exposure
		- Exposure through extracted data and aggregation
		- Strava data exposure
	+ Platform security
		- .NET platform security
* 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
		- Exception handling mess
* Denial of Service
	+ Flooding
	+ Resource exhaustion
	+ Sustained client engagement
	+ Algorithm complexity issues
		- Regular expression denial of service (ReDoS)
* Cryptography for Developers
	+ Cryptography basics
	+ Crypto APIs in C#
	+ Elementary algorithms
		- Random number generation
	+ Hashing
* Common Software Security Weaknesses
	+ Symmetric encryption
		- Block ciphers
		- Modes of operation
		- Modes of operation and IV – best practices
		- Symmetric encryption in C#
		- Symmetric encryption in C# with streams
	+ Asymmetric encryption
		- The RSA algorithm
	+ Combining symmetric and asymmetric algorithms
		- Integrity protection
	+ Message Authentication Code (MAC)
		- Calculating HMAC in C#
	+ Digital signature
		- Digital signature with RSA
		- Elliptic Curve Cryptography
	+ Code quality
		- Code quality and security
		- Data handling
		- Object-oriented programming pitfalls
		- Serialization
* Using Vulnerable Components
	+ The British Airways data breach
	+ Vulnerability management
		- Patch management
		- Vulnerability databases
		- Finding vulnerabilities in third-party components
* Conclusion
	+ Secure coding principles
		- Principles of robust programming by Matt Bishop
		- Secure design principles of Saltzer and Schroeder
	+ And now what?
		- Software security sources and further reading
		- .NET and C# resources