

**Cloud Application Security in C# for Azure**

**Course Number:** SEC-146
**Duration:** 5 days

**Overview**

This in-person or online Cloud Application Security in C# for Azure training course teaches developers how to ensure the security of their C# applications on the Azure platform. Participants learn how to avoid the security pitfalls of the C# programming language and the Azure cloud platform.

**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 students must have general knowledge of C# and web development.

**Materials**

All Application Security training 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 cloud security specialties
* Get familiar with essential cyber security concepts
* Understand how cryptography supports security
* Use cryptographic APIs correctly in C#
* Understand web application security issues
* Master the OWASP Top Ten elements
* Put Web application security in the context of C#
* Manage vulnerabilities in third-party components
* Manage cloud infrastructure security
* Incorporate input validation approaches and principles
* Identify vulnerabilities and their consequences
* Use security best practices in C#

**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
	+ Cloud security basics
		- Cloud infrastructure basics
		- The Cloud Cube Model and Zero Trust Architecture
		- Case study – ChaosDB vulnerability in Azure Cosmos DB
* Introducing the OWASP Top 10
* A01 – Broken Access Control
	+ Access control basics
	+ 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
		- Horizontal authorization
	+ File upload
		- Unrestricted file upload
		- Good practices
		- Unrestricted file upload
	+ Open redirects and forwards
		- Case study – Unvalidated redirect at Epic Games
		- Open redirects and forwards – best practices
	+ Cross-site Request Forgery (CSRF)
		- Cross-site Request Forgery
		- CSRF best practices
		- CSRF defense in depth
		- CSRF protection with tokens
* A02 – Cryptographic Failures
	+ Cryptography for developers
		- Confidentiality protection
	+ Certificates
		- Certificates and PKI
		- X.509 certificates
		- Chain of trust
		- PKI actors and procedures
		- PGP – Web of Trust
		- Certificate revocation
	+ Transport security
		- Transport security weaknesses
		- The TLS protocol
* A03 – Injection
	+ Injection principles
	+ Injection attacks
	+ SQL injection
	+ SQL injection basics
		- SQL injection
		- Attack techniques
		- Content-based blind SQL injection
		- Time-based blind SQL injection
	+ NoSQL injection
		- NoSQL injection basics
		- NoSQL injection in MongoDB
		- NoSQL injection in DynamoDB
	+ SQL injection best practices
		- Input validation
		- Parameterized queries
		- Using prepared statements
		- Additional considerations
		- Case study – Hacking Fortnite accounts
	+ Code injection
		- Code injection via input()
		- OS command injection
	+ HTML injection – Cross-site scripting (XSS)
		- Cross-site scripting basics
		- Cross-site scripting types
		- Stored XSS
		- Reflected XSS
		- Case study – XSS in Fortnite accounts
	+ XSS protection best practices
* A04 – Insecure Design
	+ The STRIDE model of threats
		- Secure design principles of Saltzer and Schroeder
		- Economy of mechanism
		- Fail-safe defaults
		- Complete mediation
		- Open design
		- Separation of privilege
		- Least privilege
		- Least common mechanism
		- Psychological acceptability
	+ Client-side security
		- Frame sandboxing
* A05 – Security Misconfiguration
	+ Cookie security
		- Cookie attributes
	+ XML entities
		- DTD and the entities
		- Entity expansion
		- Lab – Billion laughs attack
		- External Entity Attack (XXE)
* A06 – Vulnerable and Outdated Components
	+ Using vulnerable components
	+ Untrusted functionality import
	+ Malicious packages in Java
	+ Vulnerability management
		- Patch management
		- Vulnerability databases
* A07 – Identification and Authentication Failures
	+ Authentication
		- Authentication basics
		- Multi-factor authentication
		- Case study – PayPal 2FA bypass
	+ Session management
		- Session management essentials
		- Why do we protect session IDs – Session hijacking
		- Session fixation
		- Session handling in Flask
	+ Single sign-on (SSO)
		- Single sign-on concept
		- OAuth 2.0
		- SAML
	+ Identity and access management in Azure
		- Groups, roles, and credentials
		- Access tokensIdentity and access management (IAM)
	+ Password management
		- Inbound password management
		- Outbound password management
* A08 – Software and Data Integrity Failures
	+ Integrity protection
		- Message Authentication Code (MAC)
		- Digital signature
	+ Subresource integrity
		- Importing JavaScript
		- Case study – The British Airways data breach
	+ Insecure deserialization
		- Serialization and deserialization challenges
		- Integrity – deserializing untrusted streams
		- Deserialization with pickle
		- PyYAML deserialization challenges
		- Integrity – deserialization best practices
* A09 – Security Logging and Monitoring Failures
	+ Logging and monitoring principles
	+ Insufficient logging
	+ Case study – Plaintext passwords at Facebook
	+ Logging best practices
	+ Monitoring best practices
	+ Detection and monitoring
		- Utilizing Azure monitoring for security
		- Protecting logs
		- The Azure Security Hub
* A10 – Server-Side Request Forgery (SSRF)
	+ Server-side Request Forgery (SSRF)
	+ Case study – SSRF and the Capital One breach
* Cloud  Security
	+ Azure security
		- Security considerations for Azure
	+ Container security
		- Container security concerns
		- Containerization, virtualization and security
		- The attack surface
		- Docker security
	+ Kubernetes security
		- The Kubernetes architecture and security
		- Securing Kubernetes hosts
		- Best practices for Kubernetes access control
		- Building secure Kubernetes images
		- Secure deployment of Kubernetes containers
		- Protecting Kubernetes deployments at runtime
		- Case study – Azurescape
	+ Data security in the cloud
		- Data confidentiality and integrity in the cloud
		- Data privacy in the cloud
		- Compliance considerations
		- Data security in Azure
		- Storing cryptographic keys
		- Protecting data at rest
		- Protecting data in transit
		- JSON security
* Web application security beyond the Top Ten
	+ Code quality
		- Data handling
		- Initialization and cleanup
		- Object-oriented programming pitfalls
* Denial of service
	+ Flooding
	+ Resource exhaustion
	+ Sustained client engagement
	+ Algorithm complexity issues
* 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
	+ Integer handling problems
		- Representing signed numbers
		- Integer visualization
		- Integer overflow
		- Signed / unsigned confusion
		- Case study – The Stockholm Stock Exchange
		- Integer truncation
		- Best practices
	+ Files and streams
		- Path traversal
		- Lab – Path traversal
		- Path traversal-related examples
		- Additional challenges in Windows
		- Virtual resources
		- Path traversal best practices
		- Lab – Path canonicalization
	+ Unsafe reflection
		- Reflection without validation
		- Lab – Unsafe reflection
	+ Unsafe native code
	+ Native code dependence
	+ Best practices for dealing with native code
* Conclusion
	+ Secure coding principles
		- Principles of robust programming by Matt Bishop
	+ And now what?
		- Software security sources and further reading
		- .NET and C# resources
* Conclusion