Cloud application security in C# for Azure

CYDCsCldAz5d | 5 days | On-site or online | Hands-on

Your cloud 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 -2^31? Because that's what the bad guys will do – and the list is far from complete.

The cloud has become a critical aspect of online services. No matter which model you're using (SaaS, PaaS, IaaS), part of your service is now operated by someone else. This may look like a net positive, but it also greatly expands the attack surface and brings in several new risks that may not be obvious. Have you configured all security settings correctly? Are you prepared for supply chain attacks? How can you protect the confidentiality of user data in the cloud? And most importantly: can the bad guys use your exposure to their advantage?

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.

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 the Azure cloud platform.

So that you are prepared for the forces of the dark side.

So that nothing unexpected happens.

Nothing.

Cyber security skills and drills

39 LABS

18 CASE STUDIES

Audience

C# developers working on Web applications and Azure

Group size

12 participants

Preparedness

General C# and Web development

Outline

- Cyber security basics
- The OWASP Top Ten 2021
- Cloud security
- Input validation
- Wrap up

Standards and references

OWASP, CWE and Fortify Taxonomy

What you'll have learned

- Understand cloud security specialties
- Getting familiar with essential cyber security concepts
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly in C#
- Understanding Web application security issues
- Detailed analysis of the OWASP Top Ten elements
- Putting Web application security in the context of C#
- Going beyond the low hanging fruits
- Managing vulnerabilities in third party components
- Learn to deal with cloud infrastructure security
- Input validation approaches and principles
- Identify vulnerabilities and their consequences
- Learn the security best practices in C#

Table of contents

- CSRF defense in depth
- *Lab – CSRF protection with tokens*

A02 - Cryptographic Failures

- Information exposure
  - Exposure through extracted data and aggregation
  - *Case study – Strava data exposure*
- Cryptography for developers
  - Cryptography basics
  - Crypto APIs in C#
  - Elementary algorithms
    - Random number generation
    - Pseudo random number generators (PRNGs)
    - Cryptographically secure PRNGs
    - Weak and strong PRNGs
    - Using random numbers in C#
    - *Lab – Using random numbers in C#*
      - *Case study – Equifax credit account freeze*
    - Hashing
    - Hashing basics
    - Hashing in C#
    - *Lab – Hashing in C#*

Day 2

› The OWASP Top Ten 2021

A02 - Cryptographic Failures (continued)

- Cryptography for developers
  - Confidentiality protection (continued)
    - 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
    - *Lab – Symmetric encryption in C#*
    - Asymmetric encryption
    - The RSA algorithm
    - Using RSA – best practices
    - RSA in C#

    - Combining symmetric and asymmetric algorithms
  - Public Key Infrastructure (PKI)
    - Some further key management challenges
    - Certificates
    - Certificates and PKI
    - X.509 certificates
    - Chain of trust
    - PKI actors and procedures
    - PGP – Web of Trust
    - Certificate revocation
  - Transport security
    - The TLS protocol
    - TLS basics
    - TLS features (changes in v1.3)
    - The handshake in a nutshell (v1.3)
    - TLS best practices
    - HTTP Strict Transport Security (HSTS)

A03 - 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
    - Parameterized queries
    - *Lab – Using prepared statements*
    - Database defense in depth
      - *Case study – Hacking Fortnite accounts*
    - SQL injection protection and ORM
    - Entity Framework and SQL injection
- NoSQL injection
  - NoSQL injection basics
  - NoSQL injection in MongoDB
  - NoSQL injection in CosmosDB
- Parameter manipulation
  - CRLF injection
  - HTTP header manipulation
    - HTTP response splitting

    - Header checking in ASP.NET
  - HTTP parameter manipulation
    - HTTP parameter pollution
    - Value shadowing
- 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*

Day 3

› The OWASP Top Ten 2021

A03 - Injection (continued)

- HTML injection - 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
  - *Lab – Stored XSS*
  - *Lab – Reflected XSS*
  - *Case study – XSS in Fortnite accounts*
  - XSS protection best practices
    - Protection principles - escaping
    - XSS protection APIs
    - Further XSS protection techniques
    - *Lab – XSS fix / stored*
    - *Lab – XSS fix / reflected*
    - Additional protection layers – defense in depth

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
        - Cross-Frame Scripting (XFS) attacks
        - *Lab - Clickjacking*
        - Clickjacking beyond hijacking a click
        - Clickjacking protection best practices
        - *Lab – Using CSP to prevent clickjacking*

A05 - Security Misconfiguration

- Configuration principles
- Server misconfiguration
- ASP.NET and IIS configuration best practices
    - ASP.NET configuration
    - IIS configuration
- Cookie security
    - Cookie attributes
- XML entities
    - 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 vulnerability in SAP Store*
        - Preventing XXE
        - *Lab – Prohibiting DTD*

A06 - Vulnerable and Outdated Components

- Using vulnerable components
- Assessing the environment
- Hardening
- Untrusted functionality import
- Vulnerability management
    - Patch management
    - Vulnerability management
    - Vulnerability databases
    - *Lab – Finding vulnerabilities in third-party components*
    - DevOps, the CI / CD build process and Software Composition Analysis
    - Dependency checking in C#

- *Lab – Detecting vulnerable components*

A07 - Identification and Authentication Failures

- Authentication
  - Authentication basics
  - Multi-factor authentication (MFA)
  - *Case study – PayPal 2FA bypass*
- Identity and access management (IAM)
  - Identity and access management in Azure
  - Azure Active Directory
  - Multi-factor authentication and password management with Azure
  - The Azure AD hybrid identity model
  - Azure RBAC
  - Azure Shared Access Signatures (SAS)

Day 4

› The OWASP Top Ten 2021

A07 - Identification and Authentication Failures (continued)

- 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
    - *Lab – Using adaptive hash functions in C#*
    - Password policy
    - NIST authenticator requirements for memorized secrets
      - *Case study – The Ashley Madison data breach*
      - *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

A08 - Software and Data Integrity Failures

- Integrity protection
  - Message Authentication Code (MAC)
  - Digital signature
    - Elliptic Curve Cryptography
    - ECC basics
    - Digital signature with ECC
- Subresource integrity
  - Importing JavaScript
  - *Lab – Importing JavaScript*
  - *Case study – The British Airways data breach*
- Insecure deserialization
  - Serialization and deserialization challenges
  - Integrity – deserializing untrusted streams
  - Integrity – deserialization best practices
  - Look ahead deserialization
  - Property Oriented Programming (POP)
    - Creating a POP payload
    - *Lab – Creating a POP payload*
    - *Lab – Using the POP payload*

A09 - Security Logging and Monitoring Failures

- Logging and monitoring principles
- Logging best practices
- Detection and monitoring in Azure
  - Utilizing Azure monitoring for security
  - The Azure Security Center
  - The Azure Application Gateway WAF

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
  - Azure and security
  - Azure security features
  - The Azure shared responsibility model
  - Azure cloud compliance
  - Azure hardening

  - Security tools for Azure

Container security

- Container security concerns
- Containerization, virtualization and security
- The attack surface
- Docker security
  - Docker and security
  - Docker security best practices
  - *Lab – Static analysis of Docker images*
  - Hardening Docker
- Kubernetes security
  - The Kubernetes architecture and security
  - Securing Kubernetes hosts
  - Best practices for Kubernetes access control
  - Protecting Kubernetes deployments at runtime
  - *Case study – Azurescape*

Day 5

› Cloud security

Data confidentiality and integrity in the cloud
Data privacy in the cloud
Compliance considerations
Storing secrets in Azure Key Vault
Key Vault types and Hardware Security Modules
Azure Key Vault security
Access control to Azure Key Vault
Protecting data at rest
Protecting data in transit
Protecting data in use

JSON security

- Best practices

Case study – ReactJS vulnerability in HackerOne

› The OWASP Top Ten 2021

Web application security beyond the Top Ten

- Code quality
  - Data handling
  - Initialization and cleanup
    - Class initialization cycles
    - *Lab – Initialization cycles*
  - Object oriented programming pitfalls
    - Mutability
    - *Lab – Mutable object*
- Denial of service
  - Flooding
  - Resource exhaustion
  - Algorithmic complexity issues
    - Regular expression denial of service (ReDoS)
    - *Lab – ReDoS*
    - Dealing with ReDoS

› 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
- *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
- *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
- *Lab – Unsafe native code*
- Best practices for dealing with native code

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop

And now what?

- Software security sources and further reading
- .NET and C# resources