Desktop application security in C#

CYDCsDsk3d | 3 days | On-site or online | Hands-on

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 -2^31? 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 the .NET framework.

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

So that nothing unexpected happens.

Nothing.

Cyber security skills and drills

25 LABS

7 CASE STUDIES

Audience

C# developers working on desktop applications

Group size

12 participants

Preparedness

General C# development

Outline

- Cyber security basics
- Input validation
- Security features
- Errors
- Denial of service
- Cryptography for developers
- Common software security weaknesses
- Wrap up

Standards and references

CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Input validation approaches and principles
- Identify vulnerabilities and their consequences
- Learn the security best practices in C#
- Correctly implementing various security features
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly in C#

Table of contents

Day 1

- Signed / unsigned confusion
- *Case study – The Stockholm Stock Exchange*
- *Lab – Signed / unsigned confusion*
- Integer truncation
- Best practices
  - Upcasting
  - Precondition testing
  - Postcondition testing
  - Integer handling in C#
  - *Lab – Checked arithmetics*

Files and streams

- Path traversal
- *Lab – Path traversal*
- Path traversal best practices
- *Lab – Path canonicalization*

Unsafe reflection

- Reflection without validation
- *Lab – Unsafe reflection*

Day 2

› Security features

Authentication

- Authentication basics
- Multi-factor authentication (MFA)
- Time-based One Time Passwords (TOTP)
- *Case study – PayPal 2FA bypass*
- 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

Information exposure

- Exposure through extracted data and aggregation
- *Case study – Strava data exposure*

Platform security

- .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

› 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
- *Lab – Exception handling mess*

Denial of service

Flooding

Resource exhaustion

Algorithmic complexity issues

- Regular expression denial of service (ReDoS)
  - *Lab – ReDoS*
  - Dealing with ReDoS

Day 3

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#*

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
- Combining symmetric and asymmetric algorithms

Integrity protection

- Message Authentication Code (MAC)
  - Calculating HMAC in C#
  - *Lab – Calculating MAC in C#*
- Digital signature
  - Elliptic Curve Cryptography
    - ECC basics
    - Digital signature with ECC
  - Digital signature in C#
    - *Lab – Digital signature with ECDSA in C#*

Public Key Infrastructure (PKI)

- Some further key management challenges
- Certificates
  - Certificates and PKI
  - X.509 certificates
  - Chain of trust
  - PKI actors and procedures
  - Certificate revocation

› Common software security weaknesses

Code quality

- Code quality and security
- 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
  - Mutability
    - *Lab – Mutable object*
    - Readonly collections
- Serialization
  - Serialization and deserialization challenges
  - Confidentiality – Serializing sensitive data

- Confidentiality – serialization best practices
- 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*
  - Summary – POP best practices

› Wrap up

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