

C# and Web Application Security

Duration: 3 Days

Description

Writing .NET web applications can be rather complex – reasons range from dealing with legacy technologies or underdocumented third-party components to sharp deadlines and code maintainability. Yet, beyond all that, what if we told you that attackers were trying to break into your code right now? How likely would they be to succeed?

This course will change the way you look at your C# code. We'll teach you the common weaknesses and their consequences that can allow hackers to attack your system, and – more importantly – best practices you can apply to protect yourself. We give you a holistic view on the security aspects of the .NET framework – such as making use of cryptography or Code Access Security – as well as common C# programming mistakes you need to be aware of. We also cover typical Web vulnerabilities with a focus on how they affect ASP.NET web apps on the entire stack – from the CLR to modern AJAX and HTML5-based frontends. We present the entire course through live practical exercises to keep it engaging and fun.

Delegates will learn how to

- Understand basic concepts of security, IT security and secure coding
- Learn Web vulnerabilities beyond OWASP Top Ten and know how to avoid them
- Learn about XML security
- Learn how to set up and operate the deployment environment securely
- Learn client-side vulnerabilities and secure coding practices
- Learn to use various security features of the .NET development environment
- Have a practical understanding of cryptography
- Learn about typical coding mistakes and how to avoid them
- Get sources and further readings on secure coding practices

Outline

IT security and secure coding

- Nature of security
- What is risk?
- IT security vs. secure coding

- From vulnerabilities to botnets and cybercrime
- Nature of security flaws
- Reasons of difficulty
- From an infected computer to targeted attacks
- The Seven Pernicious Kingdoms
- OWASP Top Ten 2017

Web application security

- Injection
- Injection principles
- SQL injection
- Exercise – SQL Injection
- Exercise – SQL injection
- Typical SQL Injection attack methods
- Blind and time-based SQL injection
- SQL injection protection methods
- Other injection flaws
- Command injection
- Command injection exercise – starting Netcat
- Case study – ImageMagick
- Cookie injection / HTTP parameter pollution
- Exercise – Value shadowing
- Broken authentication
- Session handling threats
- Session fixation
- Exercise – Session fixation
- Session handling best practices
- Setting cookie attributes – best practices
- Sensitive data exposure
- Transport layer security
- XML external entity (XXE)
- XML Entity introduction
- XML bomb
- Exercise – XML bomb
- XML external entity attack (XXE) – resource inclusion
- XML external entity attack – URL invocation
- XML external entity attack – parameter entities
- Exercise – XXE attack

- Preventing entity-related attacks
- Case study – XXE in Google Toolbar
- Broken access control
- Typical access control weaknesses
- Insecure direct object reference (IDOR)
- Exercise – Insecure direct object reference
- Protection against IDOR
- Case study – Facebook Notes
- Failure to restrict URL access
- Security misconfiguration
- Configuration management
- Hardening
- Patch management
- NET components and environment overview
- Insecure file uploads
- Exercise – Uploading executable files
- Filtering file uploads – validation and configuration
- Cross-Site Scripting (XSS)
- Persistent XSS
- Reflected XSS
- DOM-based XSS
- Exercise – Cross Site Scripting
- Exploitation: CSS injection
- Exploitation: injecting the <base> tag
- Exercise – HTML injection with base tag
- XSS prevention
- Output encoding API in C#
- XSS protection in ASP.NET – validateRequest
- Web Protection Library (WPL)

Web application security

- Insecure deserialization
- Deserialization basics
- Security challenges of deserialization
- Deserialization in .NET
- From deserialization to code execution
- POP payload targeting MulticastDelegate (C#)
- Real-world .NET examples of deserialization vulnerabilities

- Issues with deserialization – JSON
- Best practices against deserialization vulnerabilities
- Using components with known vulnerabilities
- Vulnerability attributes
- Common Vulnerability Scoring System – CVSS

Client-side security

- JavaScript security
- Same Origin Policy
- Cross Origin Resource Sharing (CORS)
- Exercise – Client-side authentication
- Client-side authentication and password management
- Protecting JavaScript code
- Exercise – JavaScript obfuscation
- Clickjacking
- Exercise – Do you Like me?
- Protection against Clickjacking
- Anti frame-busting – dismissing protection scripts
- Protection against busting frame busting
- AJAX security
- XSS in AJAX
- Script injection attack in AJAX
- Exercise – XSS in AJAX
- XSS protection in Ajax
- Exercise CSRF in AJAX – JavaScript hijacking
- CSRF protection in AJAX
- HTML5 security
- New XSS possibilities in HTML5
- HTML5 clickjacking attack – text field injection
- HTML5 clickjacking – content extraction
- Form tampering
- Exercise – Form tampering
- Cross-origin requests
- HTML proxy with cross-origin request
- Exercise – Client side include

.NET security architecture and services

- .NET architecture
- Code Access Security

- Full and partial trust
- Evidence classes
- Permissions
- Code access permission classes
- Deriving permissions from evidence
- Defining custom permissions
- .NET runtime permission checking
- The Stack Walk
- Effects of Assert()
- Class and method-level declarative permission
- Imperative (programmatic) permission checking
- Exercise – sandboxing .NET code
- Using transparency attributes
- Allow partially trusted callers
- Exercise – using transparency attributes

Practical cryptography

- Cryptosystems
- Elements of a cryptosystem
- Symmetric-key cryptography
- Providing confidentiality with symmetric cryptography
- Symmetric encryption algorithms
- Block ciphers – modes of operation
- Other cryptographic algorithms
- Hash or message digest

Prerequisites

C# programming experience.