Training: SCADEMY

CL-NSM C# and Web application security master course

TRAINING GOALS:

As a developer, your duty is to write bulletproof code. However...

What if we told you that despite all of your efforts, the code you have been writing your entire career is full of weaknesses you never knew existed? What if, as you are reading this, hackers were trying to break into your code? How likely would they be to succeed?

This combined course will change the way you look at code. A hands-on training during which we will teach you all of the attackers' tricks and how to mitigate them, leaving you with no other feeling than the desire to know more.

It is your choice to be ahead of the pack, and be seen as a game changer in the fight against cybercrime.

Participants attending this course will:

- 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 client-side vulnerabilities and secure coding practices
- Learn about denial of service attacks and protections
- Understand security concepts of Web services
- Learn about JSON security
- Learn to use various security features of the .NET development environment
- Have a practical understanding of cryptography
- Understand essential security protocols
- Get information about some recent vulnerabilities in .NET and ASP.NET
- Learn about typical coding mistakes and how to avoid them
- Understand security testing approaches and methodologies
- Get practical knowledge in using security testing techniques and tools
- Get sources and further readings on secure coding practices

Audience:

Web developers using C#, software architects and testers

CONSPECT:

- 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
  - Classification of security flaws
    - Landwehr’s taxonomy
    - The Seven Pernicious Kingdoms
    - OWASP Top Ten 2017
- Web application security (OWASP Top Ten 2017)
  - A1 - Injection
    - Injection principles
    - SQL injection
      - Exercise – SQL injection
      - Typical SQL Injection attack methods
      - Blind and time-based SQL injection
      - SQL injection protection methods
      - Effect of data storage frameworks on SQL injection
    - Other injection flaws
      - Command injection
      - Command injection exercise – starting Netcat
      - Case study – ImageMagick
    - HTTP parameter pollution
      - Cookie injection / HTTP parameter pollution
      - Exercise – Value shadowing
  - A2 - Broken authentication
    - Session handling threats
    - Session fixation
    - Exercise – Session fixation
    - Session handling best practices
    - Setting cookie attributes – best practices

        - Cross site request forgery (CSRF)
            - CSRF prevention
    - A3 - Sensitive data exposure
        - Sensitive data exposure
        - Transport layer security
    - A4 - XML external entity (XXE)
        - XML Entity introduction
        - 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
    - A5 - Broken access control
        - Typical access control weaknesses
        - Insecure direct object reference (IDOR)
        - Exercise – Insecure direct object reference
        - Protection against IDOR
        - Case study – Facebook Notes
- Web application security (OWASP Top Ten 2017)
    - A6 - Security misconfiguration
        - ASP.NET components and environment overview
        - Insecure file uploads
        - Exercise – Uploading executable files
        - Filtering file uploads – validation and configuration
    - A7 - Cross-Site Scripting (XSS)
        - Persistent XSS
        - Reflected XSS
        - DOM-based XSS
        - Exercise – Cross Site Scripting
        - XSS prevention
        - Output encoding API in C#
        - XSS protection in ASP.NET – validateRequest
    - A8 - Insecure deserialization
        - Serialization and deserialization basics
        - Security challenges of deserialization

        - Deserialization in .NET
        - From deserialization to code execution
        - POP payload targeting MulticastDelegate (C#)
        - Real-world .NET deserialization vulnerabilities
        - Issues with deserialization – JSON
        - Best practices against deserialization vulnerabilities
    - A9 - Using components with known vulnerabilities
        - Vulnerability attributes
        - Common Vulnerability Scoring System – CVSS
    - A10 - Insufficient logging and monitoring
        - Detection and response
        - Logging and log analysis
- Client-side security
    - JavaScript security
    - Same Origin Policy
    - Simple requests
    - Preflight requests
    - Exercise – Client-side authentication
    - Client-side authentication and password management
    - Protecting JavaScript code
    - Clickjacking
        - Clickjacking
        - Exercise – IFrame, Where is My Car?
        - 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
        - Form tampering

        - Exercise – Form tampering
        - Cross-origin requests
        - HTML proxy with cross-origin request
        - Exercise – Client side include
- Denial of service
    - DoS introduction
    - Asymmetric DoS
    - Regular expression DoS (ReDoS)
        - Exercise ReDoS
        - ReDoS mitigation
        - Case study – ReDos in Stack Exchange
    - Hashtable collision attack
        - Using hashtables to store data
        - Hashtable collision
        - Hashtable collision in ASP.NET
- Data access security in .NET
    - Working with databases in .NET
    - XML security
        - Introduction
        - XML parsing
        - XML injection
            - (Ab)using CDATA to store XSS payload in XML
            - Exercise – XML injection
            - Protection through sanitization and XML validation
            - XML bomb
            - Exercise – XML bomb
    - JSON security
        - Embedding JSON server-side
        - JSON injection
        - JSON hijacking
        - Case study – XSS via spoofed JSON element
- .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
    - Role-based security
        - Principal-based authorization
        - Exercise – adding role-based authorization
        - Impersonation
- Practical cryptography
    - Rule #1 of implementing cryptography
    - Cryptosystems
        - Elements of a cryptosystem
        - .NET cryptographic architecture
    - Symmetric-key cryptography
        - Providing confidentiality with symmetric cryptography
        - Symmetric encryption algorithms
        - Modes of operation
        - Encrypting and decrypting (symmetric)
    - Other cryptographic algorithms
        - Hash or message digest
        - Hash algorithms
        - SHAttered
        - Hashing
        - Message Authentication Code (MAC)
        - Providing integrity and authenticity with a symmetric key
        - Random number generation
            - Random numbers and cryptography

            - Cryptographically-strong PRNGs
            - Weak PRNGs in .NET
            - Strong PRNGS in .NET
            - Hardware-based TRNGs
    - Asymmetric (public-key) cryptography
        - Providing confidentiality with public-key encryption
        - Rule of thumb – possession of private key
        - The RSA algorithm
            - Introduction to RSA algorithm
            - Encrypting with RSA
            - Combining symmetric and asymmetric algorithms
            - Digital signing with RSA
            - Asymmetric algorithms in .NET
            - Exercise Sign
            - Exercise – using .NET cryptographic classes
    - Public Key Infrastructure (PKI)
        - Man-in-the-Middle (MitM) attack
        - Digital certificates against MitM attack
        - Certificate Authorities in Public Key Infrastructure
        - X.509 digital certificate
- Security protocols
    - Secure network protocols
    - Specific vs. general solutions
    - The TLS protocol
        - SSL and TLS
        - Usage options
        - Security services of TLS
        - SSL/TLS handshake
        - SSL/TLS in .NET
- Security of Web services
    - Securing web services – two general approaches
    - SOAP - Simple Object Access Protocol
    - Security of RESTful web services
        - Authenticating users in RESTful web services
        - Authentication with JSON Web Tokens (JWT)
        - Authorization with REST

      - Vulnerabilities in connection with REST
    - Windows Communication Foundation security
      - Introduction to WCF
      - WCF architecture and security considerations
        - WCF architecture
        - Security considerations for the hosting environment
        - WCF security terminology
        - Transport layer security
        - Transport layer security – client authentication
        - Message level security
        - Authorization options
      - Common security issues with WCF applications
  - Desktop application security
    - Windows Forms
      - Introduction to Windows Forms
      - Using TextBoxes to ask for passwords
      - Best practices for password input
      - Exercise – TextBoxPassword
      - Other security considerations
    - Windows Presentation Foundation
      - Introduction to WPF
      - Extensible Application Markup Language (XAML)
      - WPF deployment models
    - Common security issues with desktop .NET applications
      - Resource hijacking in WPF applications
      - Exercise – LibHijack
    - Protecting .NET code
      - Protecting .NET code
      - Strong naming
      - Exercise – Using strong names
      - Authenticode
      - Exercise – Using Authenticode
    - Features and vulnerabilities
      - Accessing disabled and hidden controls
      - Control sequence attacks
      - Case study – Forms Authentication Bypass

            - NULL byte termination vulnerability
            - The Forms Authentication Bypass vulnerability in the code
            - Exploiting the Forms Authentication Bypass

- Common coding errors and vulnerabilities
    - Input validation
        - Input validation concepts
        - Integer problems
            - Representation of negative integers
            - Integer overflow
            - Exercise IntOverflow
            - What is the value of Math.Abs(int.MinValue)?
            - Integer problem – best practices
            - Case study – Integer overflow in .NET
        - Path traversal vulnerability
            - Path traversal – weak protections
            - Path traversal – best practices
            - Case study – Insufficient URL validation in LastPass
        - Unvalidated redirects and forwards
        - Unsafe native calls
            - Unsafe native calls
            - Exercise – Unsafe unmanaged code
        - Unsafe reflection
            - Implementation of a command dispatcher
            - Unsafe reflection – spot the bug!
            - Mitigation of unsafe reflection
        - Log forging
            - Some other typical problems with log files
    - Improper use of security features
        - Typical problems related to the use of security features
        - Password management
            - Exercise – Weakness of hashed passwords
            - Password management and storage
            - Brute forcing
            - Special purpose hash algorithms for password storage
            - Argon2 and PBKDF2 implementations in .NET
            - bcrypt and scrypt implementations in .NET

            - Case study – the Ashley Madison data breach
            - Typical mistakes in password management
            - Exercise – Hard coded passwords
        - Insufficient anti-automation
            - Captcha
            - Captcha weaknesses
        - Accessibility modifiers
            - Accessing private fields with reflection in .NET
            - Exercise Reflection – Accessing private fields with reflection

- Common coding errors and vulnerabilities
    - Improper error and exception handling
        - Typical problems with error and exception handling
        - Empty catch block
        - Overly broad catch
        - Using multi-catch
        - Catching NullReferenceException
        - Exception handling – spot the bug!
        - Exercise – Error handling
    - Time and state problems
        - Concurrency and threading
        - Concurrency in .NET
        - Omitted synchronization – spot the bug!
        - Exercise – Omitted synchronization
        - Incorrect granularity – spot the bug!
        - Exercise – Incorrect granularity
        - Deadlocks
        - Avoiding deadlocks
        - Exercise – Avoiding deadlocks
        - Lock statement
        - Serialization errors (TOCTTOU)
        - TOCTTOU example
        - Exercise – Race condition
        - Exercise – Exploiting the race condition
    - Code quality problems
        - Dangers arising from poor code quality
        - Poor code quality – spot the bug!

        - Unreleased resources
        - Serialization – spot the bug!
        - Exercise – Serializable sensitive
        - Private arrays – spot the bug!
        - Private arrays – typed field returned from a public method
        - Class not sealed – object hijacking
        - Exercise – Object hijacking
        - Immutable string – spot the bug!
        - Exercise – Immutable strings
        - Using SecureString

- Security testing
    - Functional testing vs. security testing
    - Security vulnerabilities
    - Prioritization – risk analysis
    - Security assessments in various SDLC phases
    - Security testing methodology
        - Steps of test planning (risk analysis)
        - Scoping and information gathering
            - Stakeholders
            - Assets
            - Security objectives for testing
        - Threat modeling
            - Attacker profiles
            - Threat modeling
            - Threat modeling based on attack trees
            - Threat modeling based on misuse/abuse cases
            - Misuse/abuse cases – a simple example
            - SDL threat modeling
            - The STRIDE threat categories
            - Diagramming – elements of a DFD
            - Data flow diagram – example
            - Threat enumeration – mapping STRIDE to DFD elements
            - Risk analysis – classification of threats
            - The DREAD risk assessment model
        - Testing steps
            - Deriving test cases

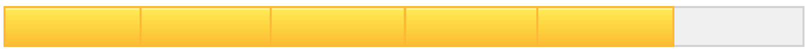
- Accomplishing the tests
- Processing test results
- Mitigation concepts
- Standard mitigation techniques of MS SDL
- Review phase

- Security testing techniques and tools
  - General testing approaches
  - Testing the implementation
    - Manual vs. automated security testing
    - Penetration testing
    - Stress tests
    - Proxy servers and sniffers
      - Testing with proxies and sniffers
      - Packet analyzers and proxies
      - Exercise – Testing with proxy
    - Web vulnerability scanners
      - Exercise – Using a vulnerability scanner
      - SQL injection tools
      - Exercise – Using SQL injection tools
- Principles of security and secure coding
  - Matt Bishop’s principles of robust programming
  - The security principles of Saltzer and Schroeder
- Knowledge sources
  - Secure coding sources – a starter kit
  - Vulnerability databases
  - .NET secure coding guidelines at MSDN
  - .NET secure coding cheat sheets
  - Recommended books – .NET and ASP.NET

REQUIREMENTS:

Advanced C# and Web application development

Difficulty level



CERTIFICATE:

The participants will obtain certificates signed by SCADEMY (course completion).

TRAINER:

Authorized SCADEMY Trainer

ADDITIONAL INFORMATION:

Training come with a number of easy-to-understand exercises providing live hacking fun. By accomplishing these exercises with the lead of the trainer, participants can analyze vulnerable code snippets and commit attacks against them in order to fully understand the root causes of certain security problems. All exercises are prepared in a plug-and-play manner by using a pre-set desktop virtual machine, which provides a uniform development environment.

SCADEMY together with online application security educational platform AVATAO (more about AVATAO www.avatao.com) for each of participant SCADEMYs authorized training adds the 30 days business AVATAO trial holds the following package:

- 30-day customized free trial