Security testing C# Web applications

CYDWebCsTst3d | 3 days | On-site or online | Hands-on

Your Web application written in C# is tested functionally, 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.

Testing for security needs a remarkable software security expertise and 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.

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.

A special focus is given to finding all discussed issues during testing, and an overview is provided on security testing methodology, techniques and tools.

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

So that nothing unexpected happens.

Nothing.

Cyber security skills and drills


29 LABS


7 CASE STUDIES

Audience

C# developers and testers working on Web applications

Group size

12 participants

Preparedness

General C# and Web development, testing and QA

Outline

- Cyber security basics
- The OWASP Top Ten 2021
- Security testing
- Wrap up

Standards and references

OWASP, CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- 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
- Input validation approaches and principles
- Understanding security testing methodology and approaches
- Getting familiar with security testing techniques and tools

Table of contents

- Injection attacks
- SQL injection
  - SQL injection basics
  - *Lab – SQL injection*
  - Attack techniques
  - Content-based blind SQL injection
  - Time-based blind SQL injection
  - Testing for SQL injection
    - SQL injection tools
  - SQL injection best practices
    - Input validation
    - Parameterized queries
  - *Lab – Using prepared statements*
- 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*
    - Testing for command injection

Day 2

› The OWASP Top Ten 2021

A03 - Injection (continued)

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

Case study – XSS in Fortnite accounts

- Testing for XSS
  - Testing for XSS with tools
- XSS protection best practices
  - Protection principles - escaping
  - XSS protection APIs
  - Further XSS protection techniques

Lab – XSS fix / stored

Lab – XSS fix / reflected

› Security testing

Security testing methodology

- Security testing – goals and methodologies
- Overview of security testing processes
- Identifying and rating assets
  - Preparation and scoping
  - Identifying assets
  - Identifying the attack surface
  - Assigning security requirements

Lab – Identifying and rating assets

- Threat modeling
  - Attacker profiling
  - SDL threat modeling
  - Mapping STRIDE to DFD
  - DFD example

Lab – SDL threat modelling with OWASP Threat Dragon

- Attack trees
- Attack tree example

Lab – Crafting an attack tree

- Misuse cases
- Misuse case examples
- Risk analysis

Lab – Risk analysis

- Accomplishing the tests

- Reporting, recommendations, and review

› The OWASP Top Ten 2021

A04 - Insecure Design

- Client-side security
  - Frame sandboxing
    - Cross-Frame Scripting (XFS) attacks
    - *Lab - Clickjacking*
    - Clickjacking protection best practices
    - *Lab – Using CSP to prevent clickjacking*
  - Testing for client-side security weaknesses

Day 3

› The OWASP Top Ten 2021

A05 - Security Misconfiguration

- Configuration principles
- Testing for misconfiguration issues
- 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*
  - Testing for XXE and XML entity-related vulnerabilities

A07 - Identification and Authentication Failures

- Session management
  - Session management essentials
  - Why do we protect session IDs – Session hijacking
  - Session fixation
  - Session ID best practices
  - Testing for session management issues
- 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
- Testing for password management issues
- Using password cracking tools
- Password cracking in Windows

Lab – Password audit with l0phtcrack

A08 - Software and Data Integrity Failures

- 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
  - Testing for insecure deserialization
  - Property Oriented Programming (POP)
    - Creating a POP payload

Lab – Creating a POP payload

Lab – Using the POP payload

A10 - Server-side Request Forgery (SSRF)

- Server-side Request Forgery (SSRF)

Case study – SSRF and the Capital One breach

- Testing for SSRF

› Security testing

Security testing techniques and tools

- Code analysis
  - Static Application Security Testing (SAST)

- *Lab – Using static analysis tools*
- Dynamic analysis
  - Security testing at runtime
  - Penetration testing
  - Stress testing
  - Dynamic analysis tools
    - Dynamic Application Security Testing (DAST)
    - Web vulnerability scanners
    - *Lab – Using web vulnerability scanners*
    - Fuzzing

› 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
- Security testing resources