Secure Coding with C#

Introduction to Secure Code

Alexander Tushinsky

Cybersecurity & Software Development Consultant

@ltmodcs alextushinsky.com

Secure Coding With OWASP in C# 10

Version Check

Version Check

This version was created by using:

- Visual Studio 2022 Community Edition
- C# version 10
- .NET 6

Version Check

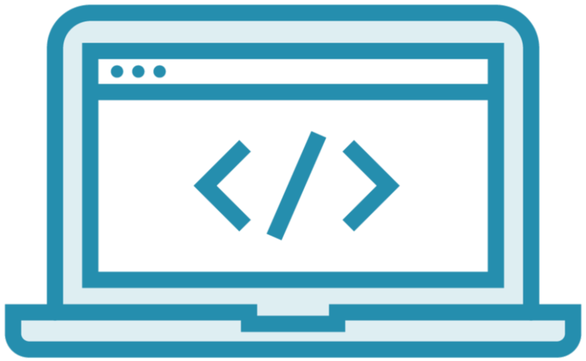
This version was created by using:

- OWASP Top 10 2021
- OWASP Proactive Controls 2018
- OWASP ASVS 4.0.3

Overview

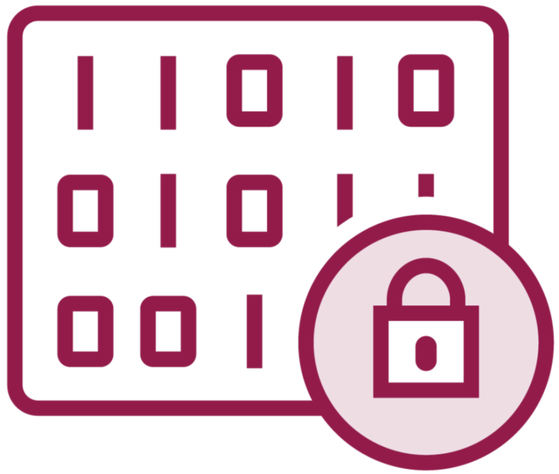
- What is secure coding?
- The Open Web Application Security Project
- How this course is organized

Why Is Software Vulnerable?



Software Development

- Developers are not exposed to secure coding training
- Security concerns addressed while writing code
- Developers may not be aware of potential issues
- “Roll-your-own” security



Software Compromises

- Over 8,000 vulnerabilities were identified in Q1 2022
- More than 50% are high-risk
- Remediation takes an average of 58 days
- 75% of attacks use vulnerabilities that are over two years old

Aimee O’Driscoll / Comparitech, 25+ Cyber Security Vulnerability Statistics and Facts of 2022, https://bit.ly/3chZ4Wv

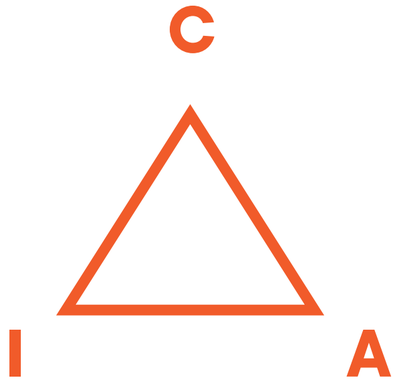
Path to Success

- Understanding of security concepts
- Finding a balanced design
- Secure SDLC
- Threat Modeling
- Programming considerations

Building Secure Software

Building Secure Software

What should we consider when building software?

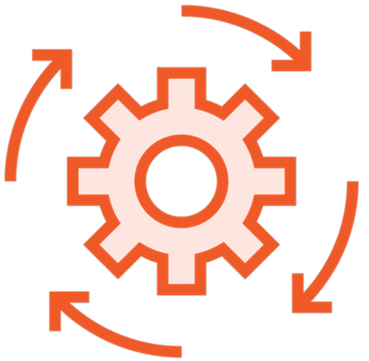


CIA Triad

Confidentiality, Integrity, and Availability

Balanced Design

Security, features, and complexity need to find a balance



Secure SDLC

Incorporate security before, during, and after the project

Confidentiality

– Protect against unauthorized access

Considerations

– How is the database protected?
– How is the data transmitted?
– Who can view or modify physical files?
– What are you logging?
– Are you using encryption?

Integrity

- Data is consistent and accurate
- Data is trustworthy

Considerations

- No unauthorized modifications
- Authentication
- Principle of least privilege
- Use of hashes
- Data submitted from public sources is not trusted
- User input is validated

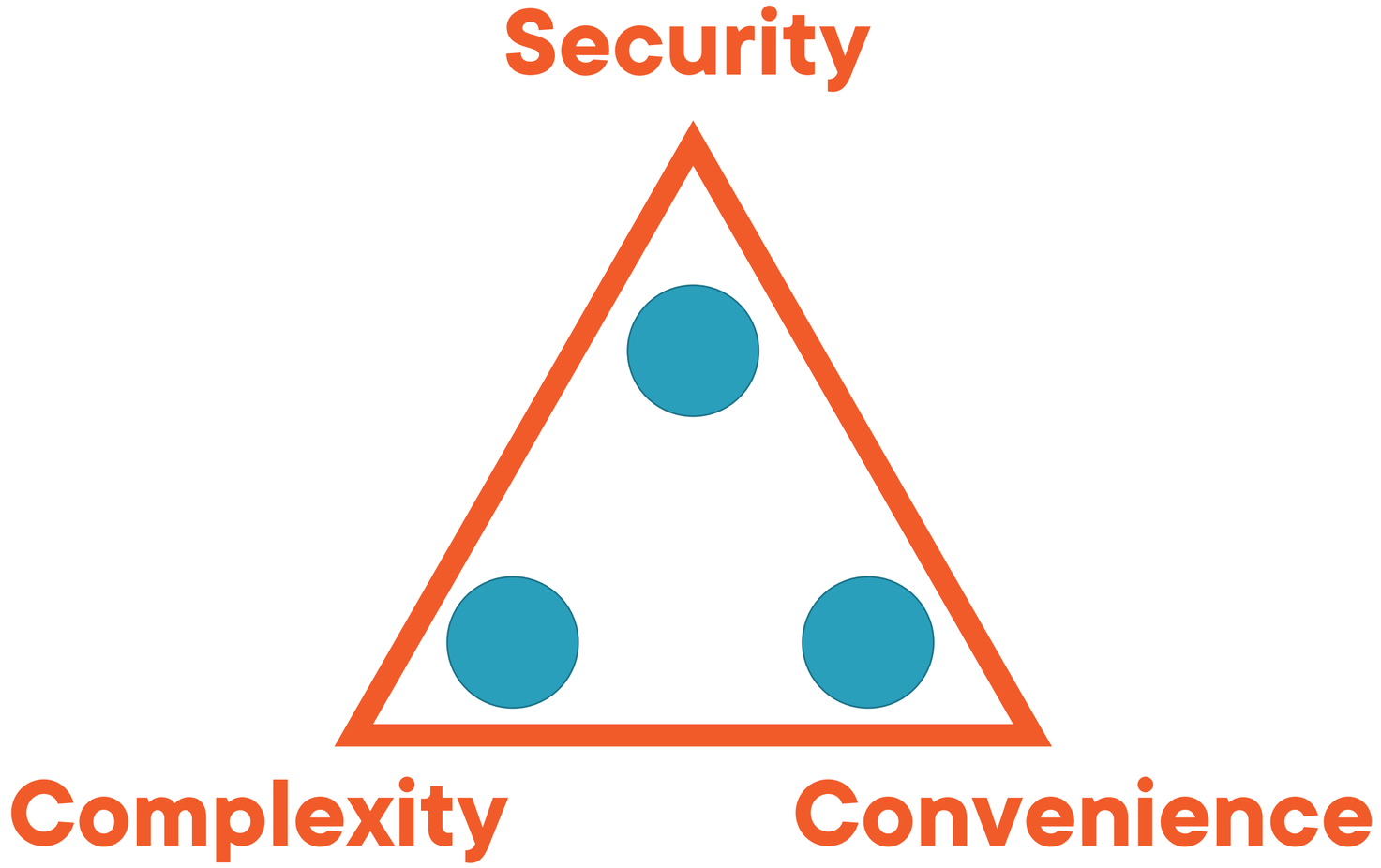
Availability

- Data is readily available to authorized users

Considerations

- Backups
- Load Balancing and High-Availability
- Request Throttling
- Firewall Use

Balanced Design

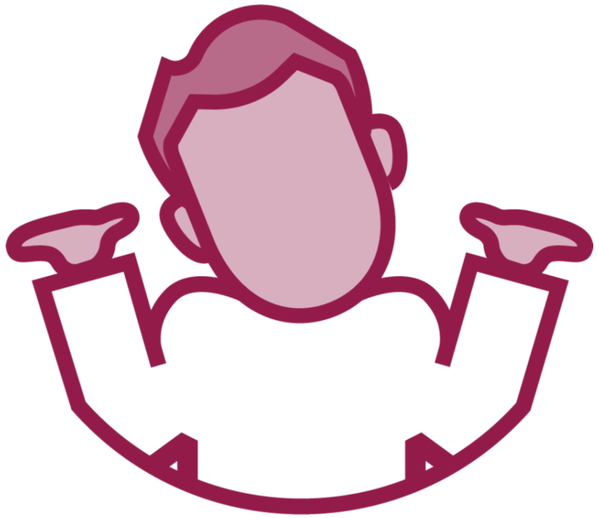


Security by Design

- Security is considered at all phases of the Software Development Life Cycle
- Operate under the assumption of a breach
- Follow principals of least privilege
- Avoid security through obscurity

Threat Modeling

- What are we building?
- What can go wrong?
- What are we going to do about it?
- Did we do a good enough job?



How do we achieve this?

- Are there guidelines?
- Is there a list of vulnerabilities for us to avoid?
- What can we do proactively?

The Open Web Application Security Project®

The Open Web Application Security Project®

- OWASP
- Nonprofit that works to improve security of software
- Community-led open-source projects
- Extensive documentation and cheat sheets
- https://owasp.org

OWASP Projects

- OWASP Top 10
- OWASP Proactive Controls
- OWASP Application Security Verification Standard
- Many, many other projects

OWASP Top 10 2021

https://owasp.org/www-project-top-ten

A01:2021 – Broken Access Control

A02:2021 - Cryptographic Failures

A03:2021 - Injection

A04:2021 – Insecure Design

A05:2021 – Security Misconfiguration

A06:2021 – Vulnerable and Outdated Components

A07:2021 – Identification and Authentication Failures

A08:2021 – Software and Data Integrity Failures

A09:2021 – Security Logging and Monitoring Failures

A10:2021 – Server-Side Request Forgery (SSRF)

OWASP Proactive Controls 2018

https://owasp.org/www-project-proactive-controls

C1: Define Security Requirements

C2: Leverage Security Frameworks & Libraries

C3: Secure Database Access

C4: Encode and Escape Data

C5: Validate All Input

C6: Implement Digital Identity

C7: Enforce Access Controls

C8: Protect Data Everywhere

C9: Implement Security Logging and Monitoring

C10: Handle All Errors and Exceptions

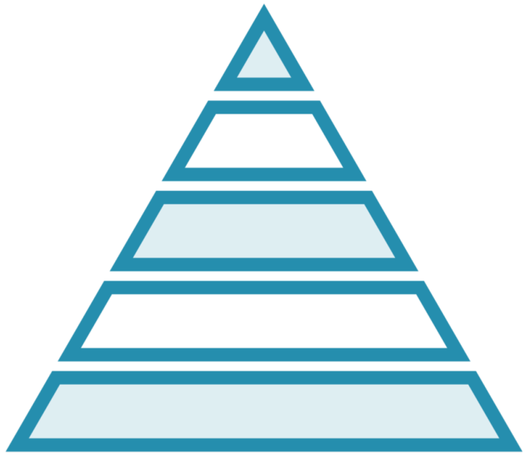
Application Security Verification Standard

- An actual standard
- Use as a metric
- Use as a guide
- Use during procurement

Application Security Verification Standard

ASVS Goals

- Help organizations develop and maintain secure applications
- Align security requirements and offerings



ASVS Severity Levels

- **Level 1** – For applications requiring minimum security
- **Level 2** – For applications that work with sensitive data. Recommended for most applications
- **Level 3** – For critical, high-performing applications that require the highest level of trust

Using ASVS

- Determine the best level for the organization
- Guide for security architecture
- Secure coding checklist
- Guide for automated unit and integration testing
- Secure development training
- Agile application security
- Framework for secure software purchases

ASVS Controls

V1 – Architecture, Design and Threat Modeling

V2 - Authentication

V3 – Session Management

V4 – Access Control

V5 – Validation, Sanitization and Encoding

V6 – Stored Cryptography

V7 – Error Handling and Logging

V8 – Data Protection

V9 - Communication

V10 – Malicious Code

V11 – Business Logic

V12 – Files and Resources

V13 – API and Web Services

V14 - Configuration

ASVS Documentation

- All versions are available on GitHub - https://github.com/OWASP/ASVS
- Extensive Cheat Sheets - https://cheatsheetseries.owasp.org

Demo

OWASP

- Project web documentation
- ASVS Git repo

How to Use This Course

- Each module is a different vulnerability
- Looked at from the perspective of OWASP Top 10, ASVS, and Proactive Controls
- Vulnerability → Impact → Remediation
- C# 10 example

Up Next:
Broken Access Control