

Course 900: Solutions and Responsibilities in Software Security (4 days)

Program Overview

Software has become ubiquitous. It's in everything from the clock on our bedside to the control tower that monitors a plane as it lands. Software should be developed with the idea that it lives in a hostile environment. It should work, even when under attack. The challenges of software life cycle management are well understood; however, most methodologies were developed at a time when the need for security wasn't as necessary as today.

This course is designed to show developers and Information System Security Engineers (ISSEs), with at least one year of experience in their respective fields, how to integrate security into each phase of the software life cycle.

The course starts by reviewing the dimensions of security as applied to a software life cycle. A prime goal is to engage security thinking and practice from the start of any development cycle. Sometimes this is simply not possible. Instructors will work with students to demonstrate how they can re-engineer security into a system that has already been deployed. This same engineering procedure can be used for new development as well.

Subsequent sections of the course show students how to incorporate security in the requirements and architectural design processes. Additional topics include software metrics for security, testing and other methods of assurance, intelligent deployment, and how to handle post-deployment maintenance updates. Each piece of the course comes alive through hands-on workshops.

See next page for a detailed course outline...



Outline

1) Introduction

- a) Dimensions of Software Security
- b) Causes of the Problems
- c) Terminology
- d) The Legislative View
- e) An ethical View
- f) Course Design

2) System Review Techniques: Part I

- a) Attack Resistance Analysis
 - i) The Procedure: Check for Threats
 - ii) Minimum Documentation
 - iii) Workshop: Attack Resistance Analysis
 - iv) Build the Process
- b) Ambiguity Analysis
 - i) The Process: Reading and Questioning
 - ii) Workshop: Ambiguity Analysis
- c) Risk Analysis: Weakness Analysis
 - i) The concept of trust
 - ii) The process: Understanding the Environment
 - (1) Understanding the Framework for the Project
 - (2) Checking the Software's Pedigree
 - iii) Workshop: Weakness Analysis

3) System Review Techniques: Part II

- a) Risk Analysis and Ordering
 - i) Qualitative vs. Quantitative Risk Analysis
 - ii) Steps for a Qualitative Risk Analysis
 - iii) Workshop: Risk Analysis
 - iv) Methods of Calculating a Dollar Cost
- b) Mitigations
 - i) What can be done
 - ii) Mitigation Strategies
 - iii) Determining the Cost of Mitigation
 - iv) Deciding What To Do
 - v) Residual Risk Preparedness
 - vi) Workshop: Mitigations
- c) Security Assurance
 - i) Guidelines
 - ii) Methodology
 - (1) Documentation Needs
 - (2) Evaluation Methods
 - (3) Auditing Requirements



4) Designing Security In

- a) Work in Requirements
 - i) Assets/Harm: Early Requirements Analysis
 - ii) Workshop: Assets/Harm Analysis
 - iii) Misuse Cases: Late Requirements Analysis
 - iv) Workshop: Misuse Cases
- b) Some First Questions
 - i) A Framework? Which one?
 - ii) Which Programming Language?
- c) Security in Architectural Design
 - i) Guiding Principles
 - ii) Challenges of Specific Techniques
 - (1) Resisting Attack
 - (2) Detecting Attack
 - (3) Recover from Attack

5) An Overview of Formal Methods

- a) Why Formal Methods are Needed
- b) What are Formal Methods
- c) An Very Simple Example
- d) Some Tools

6) Software Metrics for Security

- a) Why Have Metrics?
- b) Estimating Size
 - i) Techniques
 - ii) Relating Size to Development Time
 - iii) Relating Size to Development Effort
 - iv) Workshop: Estimating Size
- c) Project Dynamic Measurements
 - i) Change Requests
 - ii) Baseline Code
 - iii) Growth of Code
 - iv) Defect Rate
- d) Process Characteristics
 - i) Technique Example: Processes to Defect Rate
- e) Workshop: Review of Software Metric Data
- f) Cost Tracking

7) Coding Practices

- a) Coding Practices and Style Sheet
- b) Managing Code Changes
- c) Code Review with a Tool
- d) Peer and Formal Code Review
- e) Workshop: Code Review



8) Test Plans and Test Results

- a) Test Plans
 - i) The Challenges of Feature Testing
 - ii) Why Security Testing is Different
 - iii) How to Write a Security Test Plan
 - iv) Workshop: Writing a Security Test Plan
 - v) Penetration Testing
 - (1) White and Black Hat Testing
 - (2) How to Obtain Greatest ROI on Penetration Testing
- b) Testing and Results
 - i) Security Testing is Not Absolute
 - ii) How to Interpret Security Testing Results
 - iii) Workshop: Test Results Interpretation

9) Installation and Operation

- a) Secure Implementation
 - i) Secure Defaults
 - ii) Using an Installation Tool
 - iii) Methods of Securing Installation
- b) Security Bug Processing
 - i) What Not To Do
 - ii) First Steps
 - iii) Finding and Fixing Problem
 - iv) Finding Similar Problems
 - v) Steps for Prevention
 - vi) Traditional Methods of Patching

10) Security and New Features

- a) Security and a Product Upgrade
 - i) The Challenge of Backwards Compatibility
 - ii) Security Checking in Design
 - iii) Security Testing – Regression Testing
- b) Traditional Upgrades
- c) Current Update Techniques
 - i) Secure Delivery
 - ii) The Challenge of Upgrading

- ◆ **GLOSSARY**
- ◆ **BIBLIOGRAPHY**
- ◆ **RECOMMENDED RESOURCE LIST**