CSP105 - SECURE SOFTWARE TESTING

Course Learning Objectives

In Secure Software Testing, you'll start by taking a big-picture look at modern software testing practices. Then you'll learn how to test your source code during development. Following that, you'll see how you can incorporate secure testing strategies in the later stages of testing. And finally, you'll look at secure testing strategies you can use later in the software development lifecycle.

Description

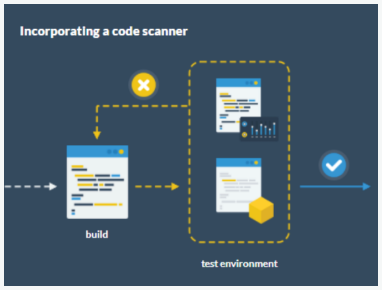
Security Software Testing is designed to improve the communication between junior software developers, managers, and application security professionals. This course addresses issues related to proper testing of software for security, including the overall strategies and plans. Learners will gain an understanding of the different types of functional and security testing that should be performed, the criteria for testing, automated tools for testing, and disaster recovery.

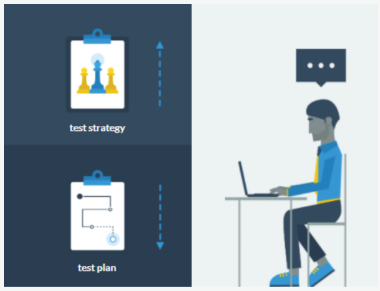
Audience

Junior Developers
Managers

Time Required

Tailored learning - 75 minutes total (approx.)





CSP105 - SECURE SOFTWARE TESTING

Course Outline

1. Planning for Security Testing

- The goals of security testing
- Testing for failure
- Testing that mitigations work
- Testing known attack types
- More testing for high-value code
- Make a test plan
- Test strategy
- Test plan
- Create a security-forward test plan
- Include security test cases
- Describe how to record issues in the test report
- Pinpoint your focus with threat modeling
- Implement testing throughout the SDLC
- Background
- Role of automation
- Automation goal

2. Testing Code

- The evolution of testing
- The patch-and-penetrate model weaknesses
- Testing early
- Understand testing approaches
- Opaque-box testing
- Clear-box testing
- Opaque-box vs. Clear-box testing
- Translucent-box testing
- Knowledge check
- Use static analysis
- Choosing a code scanner
- Incorporating a code scanner
- Use unit tests
- When to run unit tests
- How to choose good test cases
- Consider test-driven development
- Set a test coverage goal
- Fuzz testing
- Security regression tests
- Newsflash: Apple reintroduces vulnerability

3. Testing Applications

- What comes after unit testing
- Integration testing
- System testing
- Penetration testing
- Use negative scenarios
- Negative scenario examples
- Test for vulnerabilities
- Assess and document all failures
- Knowledge check
- Use automated HTTP testing for web apps
- Advantages
- Tools for automating web testing
- Automated UI testing
- Automated UI testing example

4. Penetration and System Testing

- Hardening your security posture
- Whole system view
- Configuration
- Penetration testing
- Availability testing
- Considerations of late-stage testing
- Use pen tests to find real-world vulnerabilities
- The stages of pen testing
- Knowledge check
- Supplement testing with a web security scanner
- Passive scan
- Manual scanning
- Use stress tests to improve resiliency
- Application-level DoS attacks
- Outside resources
- Nonlinear performance degradation
- Unbounded code
- Use fault injection to test mitigations
- Use disaster recovery tests to guarantee safety
- Disaster recovery test metrics

SecurityCompass