###### **Security Testing**

Liz Hinz

CSD370-A339: Secure Software Development

Professor Nathan Braun

April 20, 2025

###### **Security Testing**

Attack Surface Analysis (1)

* Description
  + Checking for points in the system that have vulnerabilities.
* Tools available
  + One available tool is Microsoft’s Attack Surface Analyzer.
* Expected results
  + After the test runs, it should reveal potential vulnerabilities or points where attackers may target. It describes the risk associated with the vulnerabilities found and provides additional information.
* Test order reasoning
  + Identifying where the main points of vulnerability are helps to find issues in code and map out an attack plan.

Scanning (2)

* Description
  + Searches for specific weaknesses and vulnerabilities, looks for undesired results, and shows operating characteristics. It can determine the particular operating system and automatically enumerate specific attributes of an application or network.
* Tools available
  + Some resources for this are Snyk, SonarQube, and Checkmarx.
* Expected results
  + Provides the team with information on how the system will perform live. Identifies weaknesses and vulnerabilities
* Test order reasoning
  + Since scanning can provide a lot of information on vulnerabilities and finding them is key, this should be high on the priority list, early on the checklist.

Fuzzing (3)

* Description
  + It is a brute force method that can be applied anywhere to check for memory leaks, buffer overflows, & wide range of input validation issues. Data is exchanged and finds many errors with a single test method, addressing input validation issues and vulnerabilities.
  + The vast majority of browser errors are found via fuzzing
  + Works with white, black & gray box testing
* Tools available
  + Some tools available for fuzzing are LibFuzzer and Burp Suite.
* Expected results
  + It verifies input validation and checks for memory leaks and buffer overflows.
* Test order reasoning

Fuzzing catches a lot of browser errors, so catching input validation vulnerabilities is better to do sooner rather than later.

Simulation Testing (4)

* Description
  + Tests how a system behaves in action. Acts as a live environment without being live.
* Tools available
  + One tool accessible for simulation is the Service Creation Environment Simulator Interface within SCE.
* Expected results
  + Verify that the program is working correctly. Tests if the system can handle itself once deployed. Checks for unstable parts.
* Test order reasoning
  + I put simulation testing here since it mirrors how an application would work in production. This helps find errors before the program is deployed to users.

Testing for Failure (5)

* Description
  + Testing how far a program can be pushed before failing.
* Tools available
  + FMEA and Simulation can help test for failure
* Expected results
  + Exposes faulty or broken code and checks that it correctly handles this input.
* Test order reasoning
  + I put testing for failure here, following a simulation, since it seems like ensuring exceptions are handled is a good next step.

Penetration Testing (6)

* Description
  + Using the tools and techniques of an attacker to try to penetrate the system.
* Tools available
  + Tools available for penetration testing are Wireshark, Hashcat, and Invicti.
* Expected results
  + Provides a breakdown of vulnerabilities and detailed information on resolving these issues.
* Test order reasoning
  + I chose penetration testing for this position since it uses practices that attackers would use, which helps identify errors. I think it makes more sense to test for these attacks later, after other vulnerabilities have already been found.

Regression Testing (7)

* Description
  + Checks previous versions of code to verify updates don’t crash the system.
* Tools available
  + Tools for regression testing include Selenium, Appium, and Testsigma.
* Expected results
  + Shows where conflicts between versions may occur.
* Test order reasoning
  + Regression made the most sense in the end because it tests to see if updated codes work with existing versions, so the code already needs to be implemented and established before this step occurs. So, this is best suited to be the last testing phase.

**References**

Conklin, WM. A., & Shoemaker, D. P. (2022). *CSSLP Certified Secure Software Lifecycle Professional: Exam Guide.* McGraw-Hill Education.