**Julio Pochet**  
**Module 5.2 – Security Testing**

**1. DAST (Dynamic Application Security Testing)**

DAST is like ethical hacking from the outside. It tests a running app by simulating attacks and looking for vulnerabilities—kind of like a hacker would. It doesn’t look at the source code, just how the app behaves while running.

**Tool Example**: OWASP ZAP (Zed Attack Proxy) is a free and powerful DAST tool. It scans live web apps for things like SQL injection or cross-site scripting.

**When Used**: This is done during or after development when the app is running in a test or staging environment.

**2. SAST (Static Application Security Testing)**

SAST checks the code before the app even runs. It looks for weaknesses right in the source code, like logic errors, insecure libraries, or hardcoded passwords.

**Tool Example**: SonarQube is a common tool that analyzes source code for bugs, vulnerabilities, and code smells without executing the app.

**When Used**: This is used early—during development or even right after writing the code.

**3. IAST (Interactive Application Security Testing)**

IAST is a mix of DAST and SAST. It runs while the app is being used (in testing), but it also checks the code and context behind the scenes.

**Tool Example**: Contrast Assess is a well-known IAST tool that runs in the background while testers interact with the app.

**When Used**: It’s used during testing stages, especially in QA environments, and provides real-time feedback.

**4. RASP (Runtime Application Self-Protection)**

RASP defends the app while it’s running in real time. It monitors behavior and blocks suspicious actions—kind of like giving your app its own security guard.

**Tool Example**: Imperva RASP is a strong tool that protects apps during runtime by blocking actual attacks automatically.

**When Used**: This is implemented in production environments when the app is live and needs to defend itself from real-world threats.