SAD LAB EXPERIMENT - 4

AIM: Study of SAST (Static Application Security Testing) Tools

TO DO:

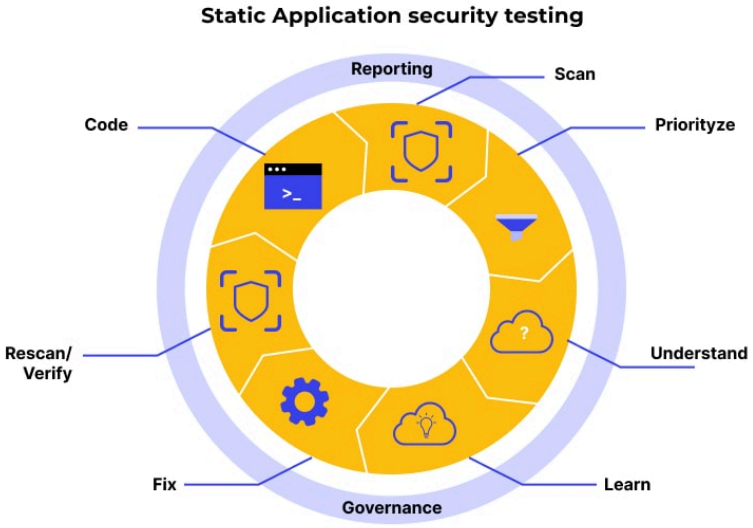
1. What is SAST?
2. Explain its Importance.
3. Write its limitations.
4. Explain any 5 tools

THEORY:

➔ **What is SAST?**

Static Application Security Testing (SAST) is a frequently used Application Security (AppSec) tool, which scans an application's source, binary, or byte code. A white-box testing tool, it identifies the root cause of vulnerabilities and helps remediate the underlying security flaws. SAST solutions analyze an application from the "inside out" and do not read a running system to perform a scan.

SAST reduces security risks in applications by providing immediate feedback to developers on issues introduced into code during development. It helps educate developers about security while they work, providing them with real-time access to recommendations and line-of-code navigation, which allows for faster vulnerability discovery and collaborative auditing. This enables developers to create more code that is less vulnerable to compromise, which leads to a more secure application, and less need for constant updates and modernization of apps and software.



SAST uses a Static Code Analysis tool, which can be thought of like a security guard for a building. Similar to a security guard checking for unlocked doors and open windows that could provide entry to an intruder, a Static Code Analyzer looks at the source code to check for coding and design flaws that could allow for malicious code injection.

- Scans source code to find weaknesses that lead to vulnerabilities
- Provides real-time reporting
- Cover languages that developers use

➔ **Importance**

- SAST is an essential step in the Software Development Life Cycle (SDLC) because it identifies critical vulnerabilities in an application before it's deployed to the public, while they're the least expensive to remediate.
- Secure coding is crucial for all software - whether you write code that runs on websites, computers, mobile devices, or embedded systems. Poorly coded software is an easy target for attackers and can be hacked to perform malicious activities.
- SAST tools can scan your code thoroughly and do it at a much faster pace than humans performing manual secure code reviews. We use SAST tools to scan millions of lines of code to automatically detect security vulnerabilities and mitigate them.
- SAST helps integrate security into the early stages of the software development lifecycle. This enables security testers to detect vulnerabilities in the proprietary code in the design stage or the coding stage when they are relatively easier to mitigate.
- It's in this stage of static code analysis that developers can code, test, revise, and test again to ensure that the final app functions as expected, without any vulnerabilities. When SAST is included as part of the Continuous Integration/Continuous Development (CI/CD) pipeline, this is referred to as "Secure DevOps," or "DevSecOps."

➔ **Limitations**

- SAST tools, however, are not capable of identifying vulnerabilities outside the code. For example, vulnerabilities found in a third-party API would not be detected by SAST and would require Dynamic Application Security Testing (DAST).
- Not capable of identifying vulnerabilities in dynamic environments
- High risk of reporting false positives
- Since the report is static, it becomes outdated quickly

- Limited Coverage: SAST programs analyze the application's static code and may miss vulnerabilities related to runtime behaviors, server configurations, or system-level vulnerabilities.
- Limited Testing Scope: SAST programs primarily focus on the application's code and may not cover other components, such as third-party libraries, frameworks, or external dependencies.
- False Positives and Negatives: SAST tools may generate false positive or false negative results. False positives occur when the tool flags code as vulnerable when it is not, leading to wasted time investigating non-existent issues. False negatives occur when the tool fails to identify actual vulnerabilities, providing a false sense of security.

➔ **Any 5 SAST Tools**

1. **Klocwork**

It works with C, C#, C++, and Java codebases and is designed to scale with any size project. The static analysis nature of Klocwork works on the fly along with your code linters and other IDE error checkers. It is especially good at finding div by zero, null pointer issues, array out of bounds, and the likes, without running the code to test it. Klocwork can help you adhere to several coding and security standards: CWE, OWASP, CERT, PCI DSS, DISA STIG, and ISO/IEC TS 17961. Users may also add custom checks, although some users found the lack of documentation around the area difficult to maneuver. Klocwork can do pre- and post-check-in analysis as part of your CI/CD pipeline to increase the overall quality of your code.

2. **SpectralOps**

SpectralOps is unique in the landscape since it scans the entire SDLC for hard coded secrets, keys, and misconfigured code, continuously. Spectral is a multi-language AI-driven SAST. The primary objective of Spectral is to prevent secrets (credentials, API keys, encryption keys, etc) from leaking. Secrets tend to be hard-coded at the early stages of development of every feature and then forgotten in the code, leaving them to be exposed to potential attackers. This is not restricted to code, but other file types are potential leaks.

Unlike most SAST, SpectralOps avoids false positives by using a sophisticated AI. Avoiding false positives is one of the most important aspects of any SAST, as a high volume of false positives is like your SAST crying wolf. Eventually, developers will ignore the warnings. Secret scanners are an essential part of any security stack you should not overlook.

3. Sonarqube

SonarQube puts developers in the position to write safer and cleaner code by automating code inspection. It additionally boosts the process of releasing quality code with the capacity to define static code analysis rules.SonarQube is versatile and expansive, with support for multi-language applications, which currently stands at 24 programming languages. Some of the vital languages it provides critical security for include C#, C++, Java, PHP, Python, and so on. Moreover, it provides code review feedback by analyzing branches of repositories during pull requests for GitHub, BitBucket, GitLab, and so on.SonarQube provides developers with multiple points of integration into the software development ecosystem. It offers integration with SonarLint while also providing integration into the development and CI/CD workflow. In terms of tools, SonarQube integrates with DevOps tools like Azure DevOps, GitHub, and Jenkins. SonarQube's Community edition is open source and free.

4. DeepSource

DeepSource is a sophisticated static analysis platform that provides enterprise-grade shift left security tools. DeepSource emphasis is on making life easier for DevSecOps and QA teams, with its continuous code quality checks. In addition to judiciously tracking the key metrics of code health, With DeepSource, you can jump right in and start analyzing code without minimal configurations. If automatically formatting your code wasn't enough, it goes a step further with its Autoflix feature that generates bug fixes so that vulnerabilities don't end up in production. DeepSource can be integrated with tools like BitBucket, GitLab, and GitHub. Moreover, DeepSource is flexible and versatile. It can be used as infrastructure-as-code and covers all the major programming languages. DeepSource uses a per-user pricing plan. However, it is free for small teams and personal accounts

5. Github

GitHub is a tool that provides significant code collaboration with the history of files in the code repository to be easily tracked. While GitHub still makes it possible to upload source code and share it with remote partners, it has evolved by adding robust security features. GitHub has recently strengthened its competencies in security by enabling developers to find and fix security problems in code as they write.

In essence, GitHub's application security allows teams to find and fix vulnerabilities before code is merged into the repositories. It facilitates the implementation of left-shift security by enabling the incorporation of security analysis into the development workflow. Thanks to CodeQL, GitHub implements real-time code scanning to provide feedback as you write while also integrating the result natively into the developer workflow.

In addition to its enabled-code scanning for repositories, GitHub also allows DevSecOps to schedule code scanning to run each time there is a pull or push request as part of code review.

GitHub provides personal, organizational, and enterprise account tiers. GitHub allows individuals and organizations to own and use an unlimited number of private and public repositories. Individuals and organizations can use either GitHub Free or GitHub Pro accounts. Likewise, organizations can use GitHub Free but to gain more control and features, they must upgrade to GitHub Team or GitHub Enterprise Cloud.

Generally, GitHub bills for advanced security features by requiring you to purchase a license for an enterprise account; specifically, either GitHub Enterprise Cloud or GitHub Enterprise Server. However, these advanced security features remain free for public repositories hosted on GitHub.com.

So, GitHub is free for individuals and organizations. GitHub Team is $44 per user/year for the first 12 months. GitHub Enterprise comes with a free trial but is billed at $231/user/year for the first 12 months. However, GitHub primarily uses per-user pricing models, so alternatively, you contact GitHub's sales team for GitHub Enterprise pricing quotes.

CONCLUSION: Thus, By detecting code flaws and vulnerabilities early in the development cycle, SAST tools are essential for improving software security since they lower the risk of potential threats and guarantee a more reliable product. The integrity and safety of software systems depend on their integration.