EXPERIMENT 4

Aim -

Study of SAST 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 need a running system to perform a scan.

Importance of SAST

According to the Forrester report, a survey of security professionals showed that almost two-thirds of external attacks in 2022 were carried out either through a web application (32 percent) or by exploiting a software vulnerability (35 percent). SAST has become synonymous with application security testing tools, but if we really want to ensure our software is secure, it's important to know how the tools we use work.

SAST enables developers to detect security flaws or weaknesses in their custom source code. The objective is either to comply with a requirement or regulation (for example, PCI/DSS) or to achieve better understanding of one's software risk. Understanding security flaws is the first step toward remediating security flaws and thus reducing software risk.

Limitation

Some limitations of SAST include:

1. Being Language-Specific: SAST reads and analyzes an application's source code, meaning that it needs to understand the language that it is written in. This can be problematic if an organization uses many different languages or less common ones.
2. The Inability To Detect All Vulnerabilities: SAST solutions are designed to analyze source code, not a running application. This leaves it blind to configuration errors and runtime vulnerabilities.
3. High False Positive Rates: SAST solutions do not perform runtime analysis, meaning that they cannot determine whether a potential vulnerability is a real threat or a false positive. SAST results must be analyzed to determine whether they represent real security risks.
4. Frequent, Time-Consuming Tests: SAST scans take a long time to run, and the report analyzes a snapshot of the code so it becomes outdated quickly. This means that SAST scans must be run frequently to remain up-to-date.

Tools

1. Klocwork

Klocwork 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
   Forgive us for the self promotion here but 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. Checkmarx
   Checkmarx is a solid SAST tool that supports numerous languages right out of the box with no configuration. Not only does it identify security issues, but it also offers solutions. It can be a great tool to try out if you're unfamiliar with SAST.

   Although the UI is a bit lacking compared to more modern solutions, it is old, reliable, does what it says on the cover, and does well. As many SAST tools tend to be, it is vulnerable to a high number of false positives.

4. Veracode
   Veracode has many security-related software solutions. Their SAST Veracode Static Analysis has a low false-positive count and offers developers potential solutions to issues it finds.

   Being Software as a Service means low setup overhead and a quick turnaround from first acquiring access and getting results. However, Veracode does not offer a free version to try out. That said, reviewers are overall pleased with the product, particularly in maintaining security standards.

5. Github
   If you're using GitLab CI/CD, you can use Static Application Security Testing (SAST) to check your source code for known vulnerabilities. You can run SAST analyzers in any GitLab tier. The analyzers output JSON-formatted reports as job artifacts.

   With GitLab Ultimate, SAST results are also processed so you can:

   - See them in merge requests.
   - Use them in approval workflows.
   - Review them in the security dashboard.

   A pipeline consists of multiple jobs, including SAST and DAST scanning. If any job fails to finish for any reason, the security dashboard does not show SAST scanner output. For example, if the SAST job completes but the DAST job fails, the security dashboard does not show SAST results. On failure, the analyzer outputs an exit code.

Conclusion -

Thus we have successfully completed the experiment by studying the importance and identifying the various SAST tools.